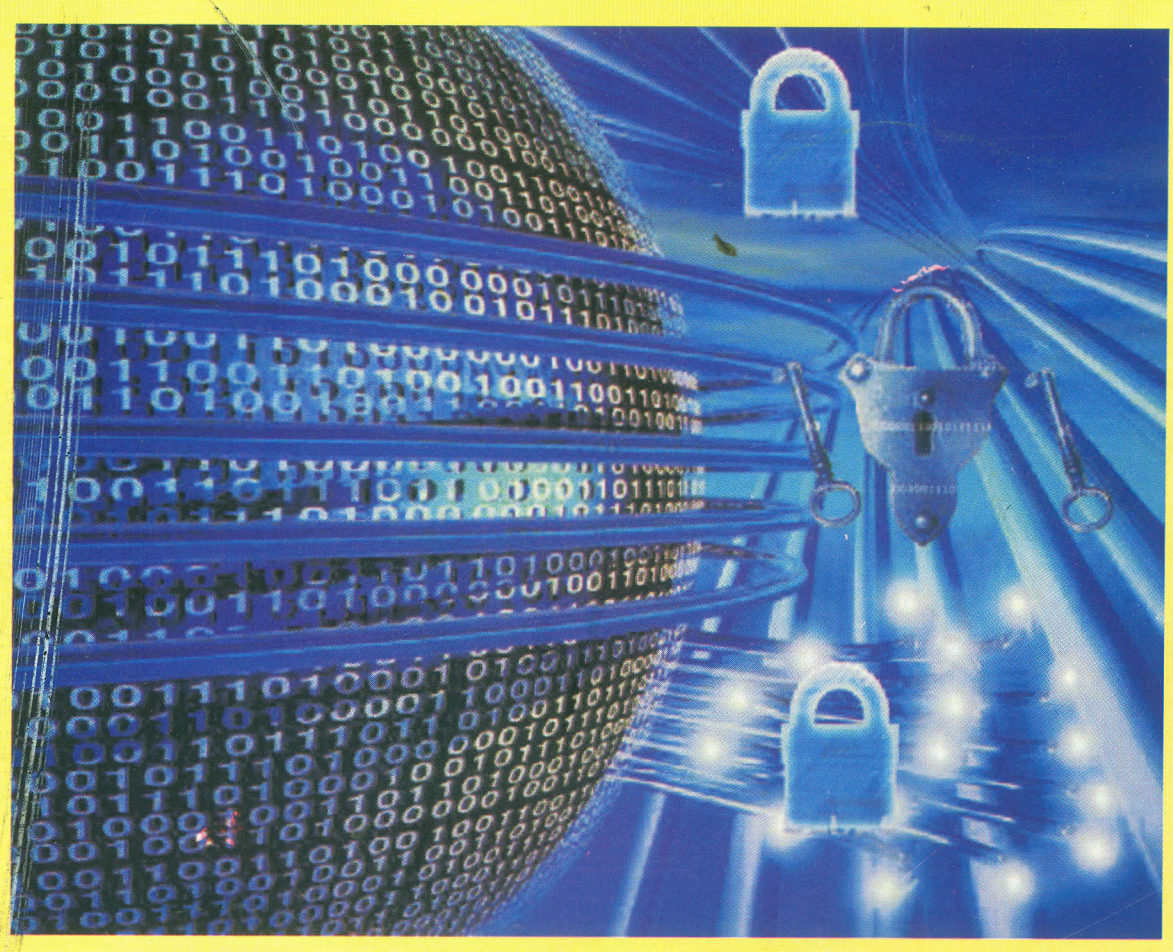
ignou
THE PEOPLE'S
UNIVERSITY
Indira Gandhi National Open University
School of Vocational Education and Training

MSEI-025
Application and Business Security Developments



Secure Application Development-I

2

"शिक्षा मानव को बन्धनों से मुक्त करती है और आज के युग में तो यह लोकतंत्र की भावना का आधार भी है। जन्म तथा अन्य कारणों से उत्पन्न जाति एवं वर्गगत विषमताओं को दूर करते हुए मनुष्य को इन सबसे ऊपर उठाती है।"

– इन्दिरा गांधी

"Education is a liberating force, and in our age it is also a democratising force, cutting across the barriers of caste and class, smoothing out inequalities imposed by birth and other circumstances."

—Indira Gandhi

Indira Gandhi National Open University
School of Vocational Education and Training

MSEI-025
Application and Business Security Developments

Block

2

SECURE APPLICATION DEVELOPMENT -I

UNIT 1

Critical Application Security Concepts 5

UNIT 2

Input Validation and Encoding 20

UNIT 3

Authentication, Authorization and Session Management 61

UNIT 4

Encryption, Confidentiality and Data Protection 91

Programme Expert/ Design Committee of Post Graduate Diploma in Information Security (PGDIS)

Prof. K.R. Srivathsan
Pro Vice-Chancellor, IGNOU

Mr. B.J. Srinath, Sr. Director & Scientist 'G', CERT-In, Department of Information Technology, Ministry of Communication and Information Technology Govt of India

Mr. A.S.A Krishnan, Director, Department of Information Technology, Cyber-Laws and E-Security Group, Ministry of Communication and Information Technology, Govt of India

Mr. S. Balasubramony, Dy. Superintendent of Police, CBI, Cyber Crime Investigation Cell, Delhi

Mr. B.V.C. Rao, Technical Director, National Informatics Centre, Ministry of Communication and Information Technology

Prof. M.N. Doja, Professor, Department of Computer Engineering, Jamia Milia Islamia
New Delhi

Dr. D.K. Lobiyal, Associate Professor, School of Computer and Systems Sciences, JNU
New Delhi

Mr. Omveer Singh, Scientist, CERT-In, Department of Information Technology, Cyber-Laws and E-Security Group, Ministry of Communication and Information Technology, Govt of India

Dr. Vivek Mudgil, Director, Eninov Systems
Noida

Mr. V.V.Subrahmanyam, Assistant Professor
School of Computer and Information Science
IGNOU

Mr. Anup Girdhar, CEO, Sedulity Solutions & Technologies, New Delhi

Prof. A.K. Saini, Professor, University School of Management Studies, Guru Gobind Singh Indraprastha University, Delhi

Mr. C.S. Rao, Technical Director in Cyber Security Division, National Informatics Centre, Ministry of Communication and Information Technology

Prof. C.G. Naidu, Director, School of Vocational Education & Training, IGNOU

Prof. Manohar Lal, Director, School of Computer and Information Science, IGNOU

Prof. K. Subramanian, Director, ACIIL, IGNOU
Former Deputy Director General, National Informatics Centre, Ministry of Communication and Information Technology, Govt. of India

Prof. K. Elumalai, Director, School of Law
IGNOU

Dr. A. Murali M Rao, Joint Director, Computer Division, IGNOU

Mr. P.V. Suresh, Sr. Assistant Professor, School of Computer and Information Science, IGNOU

Ms. Mansi Sharma, Assistant Professor, School of Law, IGNOU

Ms. Urshla Kant
Assistant Professor, School of Vocational Education & Training, IGNOU
Programme Coordinator

Block Preparation

Unit Writers

Ms. Meenu Chopra
Assistant Professor
Vivekananda Institute of Professional Studies
Pitampura, Delhi (Unit 1)

Mr. Deepak Sharma
Assistant Professor (Computer Application), Jagannath International Management School
Vasant Kunj, New Delhi
(Unit 2)

Ms. Surabhi Deshpande
Assistant Professor
Rukmini Devi Institute of Advanced Studies, Madhuban Chowk, Rohini, Delhi
(Unit 3)

Mr. Arun Bakshi
Sr. Assistant Professor (Information Technology)
Gitarattan International Business School (giBS)
Madhuban Chowk
Delhi. (Unit 4)

Block Editor

Mr. P.V. Suresh
Sr. Assistant Professor
School of Computer and Information Science, IGNOU

Ms. Urshla Kant
Assistant Professor, School of Vocational Education & Training, IGNOU

Proof Reading and Format Editing

Ms. Urshla Kant
Assistant Professor, School of Vocational Education & Training IGNOU

PRODUCTION

Mr. B. Natrajan Dy. Registrar (Pub.) MPDD, IGNOU	Mr. Jitender Sethi Asstt. Registrar (Pub.) MPDD, IGNOU	Mr. Hemant Parida Proof Reader MPDD, IGNOU

August 2011

ISBN: 978-81-266-5890-9

Further information on the Indira Gandhi National Open University courses may be obtained from the University's office at Maidan Garhi, New Delhi-110 068 or the website of IGNOU www.ignou.ac.in

Printed and Published on behalf of the Indira Gandhi National Open University, New Delhi, by the Registrar, MPDD.

Printed at: Berry Art Press A-9, Mayapuri, Phase-I New Delhi-64

BLOCK INTRODUCTION

This block deals with the secure application development-I. Organizations consist of large amount information which is asset to the organization. And if a company wants to sustain in this competitive world it should intelligently share this information with its customers, business partners and employees. This asset should be well protected from threats which may lead to financial loss and other harm to the company. Examples of such loss can be disclosure of trade secrets, damaged reputation, decreased customer confidence, etc. The aim of computer security is to find ways to protect this asset by the selection and application of appropriate safeguards. Successful companies arrange a computer security strategy known as defense-in-depth, which is a layered approach that relies on people, operations and intelligent application of multiple techniques and technologies to achieve the desired level of information assurance by arranging the effective safeguards intelligently, companies are able to manage the risk factors by decreasing the vulnerabilities to threats, which results in less possibility of compromise and financial consequences. Companies depend on computing services and applications to share information assets. These services and applications consist of a combination of software and custom applications that provide solutions to meet business goals. This block comprises of four units and is designed in the following way;

The **Unit one** covers Critical Application Security Concepts. It presents an overview of the current trends in the application security market, important drivers and inhibitors of application security, the evolving market landscape, and service provider delivery models for application security solutions. Additionally, it examines the relevance of the network for these solutions, and offers considerations for security professionals involved in solution implementations.

The **Unit two** is an effort towards answering some of the fundamental queries about input validation, its necessity and effects of ignoring input validation. It covers various input validation approaches like No input validation, Blacklisting, whitelisting and sanitizing etc. These input validation approaches are very useful to validate the inputs supplied by the outside world. Mostly, Integrated development environments which you are using are very useful in validating the inputs as the case with STRUTS, ASP.NET and Environmental variable.

Unit three is an effort towards answering some of the fundamental queries about authentication, authorization and session management. Authentication is mainly required to tie the other two principles together. We have tried to give an overview of how to authentication is done and what are the various modes of authentication. Session management which is used to manage the authorization rights for a specific session is also covered. Depending upon the confidentiality, privacy etc you can identify which scheme to be used for you application.

Security is really about risk management. Risk can be defined as the probability of events which may occur and produce negative consequences. This **unit four** covers encryption techniques, confidentiality and data protection. Type cryptography is also explained with its different types. The role of algorithm is also discussed.

Hope you benefit from this block.

ACKNOWLEDGEMENT

The material we have used is purely for educational purposes. Every effort has been made to trace the copyright holders of material reproduced in this book. Should any infringement have occurred, the publishers and editors apologize and will be pleased to make the necessary corrections in future editions of this book.

UNIT 1 CRITICAL APPLICATION SECURITY CONCEPTS

Structure

1.0 Introduction
1.1 Objectives
1.2 General Approach and Framework Definition
1.3 The Evolving Threats
1.4 Primary Drivers and Inhibitors
1.5 Market Landscape History
1.6 Service Delivery Models
1.7 Nexus Markets and Service Packing
1.8 Network Relevance for Application Security
1.9 Let Us Sum Up
1.10 Check Your Progress: The Key
1.11 Suggested Readings

1.0 INTRODUCTION

The protection of the applications and data that drive business processes and transactions is critical for ensuring business availability, employee productivity, revenue loss avoidance, and brand and corporate reputation protection. As the miscreant economy spreads across all sectors of the economy and threats increase in sophistication and complexity, the protection of mission-critical business assets, applications, data, and processes has become ever more essential.

1.1 OBJECTIVES

After studying this unit, you should be able to:

- explain the General Approach and Framework Definition;
- describe the Evolving Threats;
- define Primary Drivers and Inhibitors;
- explain Market Landscape;
- explain Service Delivery Models;
- explain Nexus Markets and Service Packing; and
- explain Network Relevance for Application Security

1.2 GENERAL APPROACH AND FRAMEWORK DEFINITION

There are many definitions of what constitutes application security. Many industry sources utilize the application lifecycle approach that emphasizes application fortification throughout the design, development, deployment, upgrade, and maintenance phases. Because providers of managed services are primarily involved at the front end of the application lifecycle (deployment and upgrade), this paper focuses on these stages. Within this scope, the definition of application security will be regarded as encompassing the use of software, hardware, policies, and procedural methods to protect business applications and data, either in static or dynamic form, from internal and external threats.

Applications are regarded as data or software programs running within and over business networks to enable business processes and services. This includes both systems software (for managing IT operating systems and IT resources) and applications software (for managing end-user requirements and business processes). As such, the protection of systems, end-user applications, and other business-related data, whether stored or in transit, is critical for business availability, employee productivity, revenue protection, and brand and reputation protection.

This requires that IT managers tasked with deploying application security should define clear business objectives, as well as identify business-critical applications, assets, and processes to be secured before proceeding. In addition, points of protection and their capabilities should also be defined and taken into consideration when evaluating the various security solutions to be deployed. As such, application security is ultimately concerned with the mitigation of business risk and enablement of internal and external compliance requirements by assuring the integrity and availability of applications that support business operations.

1.3 THE EVOLVING THREATS

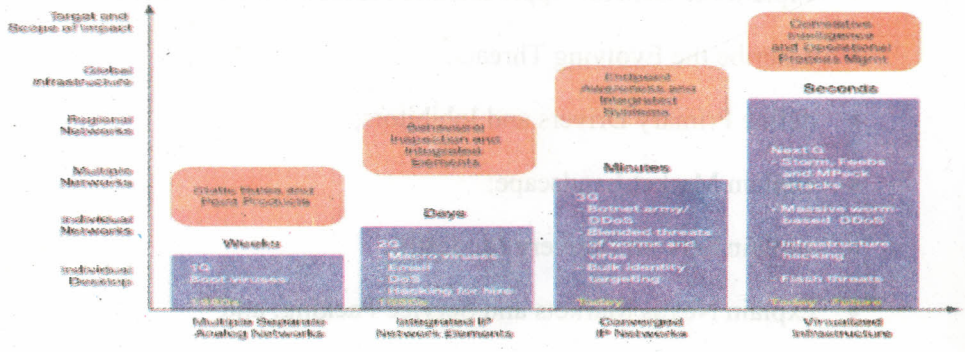


Fig. 1

Over the last few years, new threats have continued to emerge (see Fig. 2), taking advantage of the significant increase in system vulnerabilities. The two largest vulnerability categories are web applications and PC software. Web application vulnerabilities, which are typically remotely exploitable in nature, constitute the fastest growing and largest segment of the overall vulnerability count. In 2008, remotely exploitable vulnerabilities represented 90.2 percent of all vulnerabilities, up from about 85 percent in 2005. The most common attack techniques were SQL injection, cross-site scripting, and file-include.

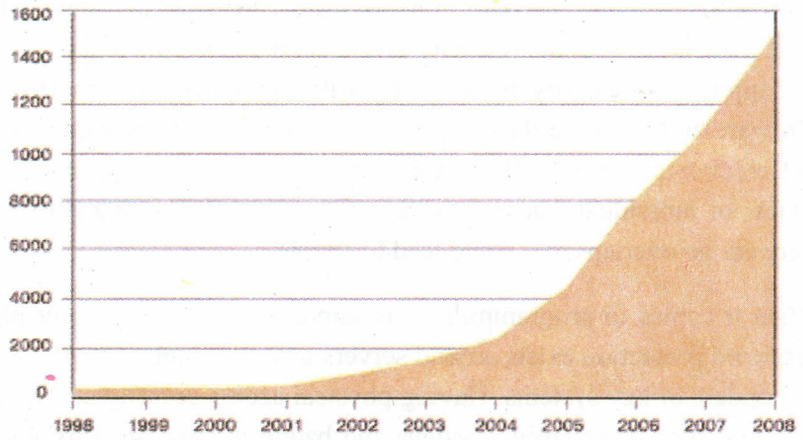


Fig. 2: Web Application Vulnerabilities – Cumulative Count, 1998 – 2008

Application security has become a pervasive requirement across many markets, and the increasing trend and adoption of web services and business process collaboration are behind its growing relevance. Also accelerating demand is the rapidly evolving, highly sophisticated miscreant economy, which poses a direct threat to business availability and profitability.

Application security is not a standalone security requirement. Rather, it must be addressed across business networks and, more importantly, across business processes. It should be regarded as part of the broader set of business requirements for minimizing business risk, maximizing employee productivity, and protecting company brands and corporate reputations.

Consideration must also be made for application security, in terms of the multiplicity of service delivery models available. These include software only, premises-based, hosted, fully managed, hybrid, and Software-as-a-Service delivery options. The delivery of services over virtualized network-based infrastructures is also gaining momentum. Cisco is well positioned to provide flexible architectures and cost-effective solutions that support variable service provider requirements, and can support providers regardless of their existing infrastructure or delivery models.

The above market trends underscore the need for service providers to address application security. However, the growing revenue opportunity provides the

most compelling reason to pay attention. Service providers can increase and diversify revenues with application security services, add value to their portfolios by bundling in application security solutions, and create a foundation for other security-sensitive managed services such as unified communications, collaboration, and rich media services. The direct and indirect revenue opportunities now place application security in the category of a fundamental offering for forward-looking service providers.

Application Security centers around three main functions: Programming, Processing, Access. By and large the two concepts of application security and segregation of duties are both in many ways connected and they both have the same goal, to protect the integrity of the companies' data and to prevent fraud. For application security it has to do with preventing unauthorized access to hardware and software through having proper security measures both physical and electronic in place. With segregation of duties it is primarily a physical review of individuals' access to the systems and processing and ensuring that there are no overlaps that could lead to fraud.

When it comes to programming it is important to ensure proper physical and password protection exists around servers and mainframes for the development and update of key systems. Having physical access security at your data center or office such as electronic badges and badge readers, security guards, choke points, and security cameras is vitally important to ensuring the security of your applications and data. Then you need to have security around changes to the system. Those usually have to do with proper security access to make the changes and having proper authorization procedures in place for pulling through programming changes from development through test and finally into production.

With processing it is important that procedures and monitoring of a few different aspects such as the input of falsified or erroneous data, incomplete processing, duplicate transactions and untimely processing are in place. Making sure that input is randomly reviewed or that all processing has proper approval is a way to ensure this. It is important to be able to identify incomplete processing and ensure that proper procedures are in place for either completing it, or deleting it from the system if it was in error. There should also be procedures to identify and correct duplicate entries. Finally when it comes to processing that is not being done on a timely basis you should back-track the associated data to see where the delay is coming from and identify whether or not this delay creates any control concerns.

Finally, access, it is important to realize that maintaining network security against unauthorized access is one of the major focuses for companies as threats can come from a few sources. First you have internal unauthorized access. It is very important to have system access passwords that must be changed regularly and that there is a way to track access and changes so you

are able to identify who made what changes. All activity should be logged. The second arena to be concerned with is remote access, people accessing your system from the outside through the internet. Setting up firewalls and password protection to on-line data changes are key to protecting against unauthorized remote access. One way to identify weaknesses in access controls is to bring in a hacker to try and crack your system by either gaining entry to the building and using an internal terminal or hacking in from the outside through remote access.

Check Your Progress 1

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) What is security? What are its evolving threats?

.....

.....

.....

.....

2) What are web based Vulnerabilities?

.....

.....

.....

.....

.....

.....

3) Discuss general approach to security and its framework in detail.

.....

.....

.....

.....

1.4 PRIMARY DRIVERS AND INHIBITORS

There are many forces at work that both encourage and discourage the adoption of application security solutions. Forces driving the need for application security include:

- **Aggressive cyber criminals.** Hackers continually change strategies to exploit security loopholes. Although businesses have invested in network security, they are not well equipped to identify or block threats that target application-based vulnerabilities.

- **Permeation of web services, Web 2.0, and applications.** The numbers and types of non-IT-controlled applications continue to grow on corporate networks. The number of endpoints, including mobile workers' laptops and smart devices, has also skyrocketed. Applications are more exposed than ever, particularly where public and private networks interface.

- **Outsourced IT models.** As businesses rely on service providers to manage or host some or all of their critical assets (such as data centers, storage servers, and networks), both customers and service providers must share security responsibilities. Outsourcing must not compromise the security of enterprise resource planning (ERP), customer relationship management (CRM), collaboration, video, and other applications.

- **Cost pressures.** Consolidation, virtualization, and standardization initiatives are being employed to strip down infrastructures and operating costs. Ensuring application security becomes a top priority within a virtualized environment, since applications are no longer tied to a specific server and can be shared across groups and teams.

- **Compliance and regulatory requirements.** Governance and compliance requirements have put additional burdens on the IT groups within retailers, healthcare providers, financial services companies, and numerous other vertical industries. These evolving regulations are driving a sub-market relating to application security testing, and this is just the beginning. For example, it is expected that the Federal Information Security Management Act (FISMA) will soon be revised to include stiffer regulations and enforcement.

- **Network-based computing.** Enterprise customers want to be able to take advantage of virtualized, scalable computing and storage resources offered by service providers, but only if they can be assured that their business data and applications are secure within the virtualized environment.

Factors that currently inhibit application security include:

- **Lack of application security expertise.** Service providers and enterprise IT teams are well versed when it comes to basic understanding and relevance of mature network security tools such as premises-based firewalls, VPNs, intrusion prevention and detection solutions, and Network Access Controls (NACs). However, many of these same professionals lack experience and skill sets relating to web application firewalls or application security testing.

- **Inertia and confusion around need for web application security.** Many businesses feel that endpoint solutions are adequate and are not aware of the emerging blended threats that employ a combination of email and web vectors for attack. Analysts such as IDC recommend a hybrid approach for web and email security, which offers a higher degree of protection by layering network-based services with traditional endpoint and perimeter customer-premises solutions.

- **Lack of push from leading security services players.** Symantec, Check Point, IBM ISS, and Microsoft are protecting large installed bases of software-only or premises-based solutions. They also currently enjoy pull-through revenues for integration services surrounding these products. Application security, in comparison to endpoint security, calls for a more flexible arsenal of solutions and delivery models.

1.5 MARKET LANDSCAPE HISTORY

Because the driving forces are greater than the inhibitors, demand is growing for application security services. Even so, the market remains embryonic and fragmented with a prevalence of niche providers. Current solution providers fall into six broad categories, in terms of the products and services they provide:

- **Application security vendors.** Include a mix of small private players such as F5, Check Point, startups, and a few more-established solution providers such as Symantec and Microsoft that give weight to this category.

- **Systems and network integrators.** Include providers such as HP and IBM, with extensive application development, integration, network aggregation, and data center capabilities, as well as extensive reach into enterprise data centers.

- **Hosting and data center-centric providers.** These providers, such as Savvis, leverage their existing data center infrastructure to offer hosted and virtual security solutions.

- **Network service providers.** This category is dominated by large traditional network providers like Verizon and BT that are looking to extend the value of their network to businesses, in part by offering security solutions.

- **Emerging virtual infrastructure providers.** Companies such as Google and Amazon offer a combination of virtual development platforms and network-based solutions for end users, and are increasingly targeting enterprise customers with enterprise-grade solutions and service-level agreements.

- **Application security testing providers.** This category is essentially a sub-market of the larger application security market. Services include static and dynamic application security testing, and providers range from small specialist providers like Cigital to large software providers like Oracle that bundle security testing solutions alongside other application services.

Target customers in this market include both enterprises and small and medium-sized businesses (SMBs) although they have different security requirements. Enterprises want to gain efficiencies for meeting compliance and audit requirements, and need to lower the total cost of ownership while simultaneously securing business assets. SMBs are struggling to keep up with the rapidly evolving security threats, because they operate with much smaller in-house IT resources and budgets but still face the same threats as much-larger enterprises. However, SMB solutions pose additional challenges for security vendors because they must defend against the same threats types but provide simplified interfaces for ease of use and management.

Check Your Progress 2

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.:

1) What are Forces driving the need for application security.

.....

.....

.....

.....

2) What are factors inhibiting the application.

.....

.....

.....

.....

1.6 SERVICE DELIVERY MODELS

Service delivery models greatly impact the overall value and effectiveness of the application security solution. Like many other business solutions, application security is currently delivered using one or a combination of the following models:

Customer-premises-based, customer-managed

• Hosted service

• Managed service

• Software-as-a-Service (SaaS)

Network-based services and managed services, in general introduce more stringent requirements for application security, thereby creating an internal demand for the customer-facing service. For securing data and applications in a virtualized or multi-tenant environment, service providers must rely on federated identity capabilities. Consider the following services, capabilities, and deployment models that require application security built into the service delivery model itself:

- Collaboration services are often purchased by businesses that need to provide access to partners, suppliers, customers, and other external parties (for example, an ecosystem).
- Document sharing, ERP/CRM/BI access, and hosted content introduce additional application and related content security requirements for the broader user community.
- New rich media applications such as high- definition (HD) video, TelePresence, and 2D and 3D computer aided design (CAD) are increasingly used to support business collaboration and are delivered over IP networks. Shared with customers and partners, these applications introduce additional security concerns regarding access, media application encryption, white lists/black lists, user groups, and inter-provider security policies.
- SaaS application models, in which the service provider may host or store application content in disparate geographies, exposes business data to local regulatory environments. This imposes the need for greater service flexibility and policy options for hosted or virtualized resources.

Traditional service delivery models have been based on either software licenses only, or a combination of on-premises equipment (leased or purchased) and software licenses. Both of these models typically include monthly recurring charges for updates and/or maintenance. Today, providers of all types, from pure software providers to network service providers and virtual infrastructure providers, are optimizing their infrastructures and assets for volume-based multi-tenanted capability and service delivery.

SaaS and on-demand are emerging as the preferred delivery and consumption models in many cases. For SMBs, network-based service models are affordable and can help them overcome skill gaps and operate with limited backup and storage resources. Enterprises, while not as resource-constrained as SMBs, have security requirements around large distributed knowledge workers accessing the corporate network anywhere, anytime, in any format (multiple

endpoint types). They also have multiple security and IT compliance requirements around data leakage, and must often secure extended supply chains into, and on top of, their networks – mandating hardened, private WAN-public network interfaces.

1.7 NEXUS MARKETS AND SERVICE PACKING

At a vertical level, application security is emerging as a standalone service category in its own right. It can be bundled with other security services, particularly alongside data loss prevention and access control. For customers that do not have dedicated in-house security experts, this type of bundling is preferred. Another vertical option is to sell it as part of an enhanced unified threat management (UTM) service.

At a horizontal level, application security is also offered with other managed services that are commonly required within a particular market. Some providers also offer bundles that enrich broader service portfolios by combining application security with collaboration, unified communications, data center, virtualized servers, or other managed services. In doing so, the application-security-embedded service bundle enables service providers to offer a more complete service portfolio, increasing their value and relevance to their target customers.

Increasing demand for application security is also driving the emerging Secure Web Gateway market. Secure Web Gateway solutions provide URL filtering, malware blocking, and controls for web-based applications such as voice-over-IP services or instant messaging content. These capabilities give users the ability to safely use the Internet. Businesses can use Secure Web Gateways to enforce company and compliance policies. An increasing number of service providers are offering Secure Web Gateways alongside other security services.

1.8 NETWORK RELEVANCE FOR APPLICATION SECURITY

A minority of the overall security market still exhibits limited understanding of the need for dedicated application security solutions. However, widespread acceptance is driving security origination, operations, and delivery further back into the network. The network has become a critical platform on which to deliver, manage, and monetize security services. This creates an opportunity for value-added service (VAS) offerings from different market players, while posing challenges too. Many service provider networks are not adequately configured to block next-generation web threats.

Migration to a flexible network-based service delivery model can help service providers overcome these limitations. Cisco is uniquely positioned to help service providers evolve existing infrastructures and build security into every layer of their service delivery infrastructure (see Figure 3).

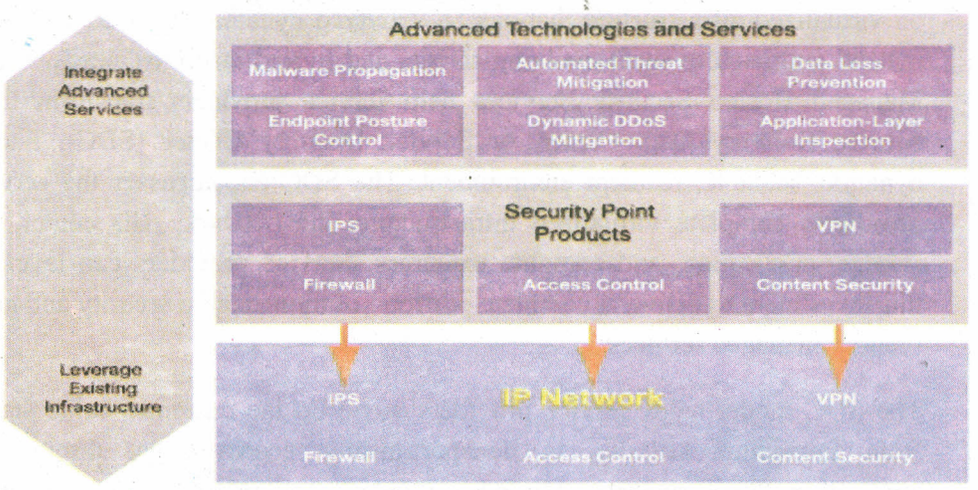


Fig. 3: Virtualized Network-Based Security as the Platform for Application Security

Cisco® solutions enable service providers to deliver relevant application security solutions in the short term, and also evolve to a new Web 2.0 service model for the medium to long term. The Cisco network-based approach for application security takes advantage of the emerging virtualized, service-oriented model where applications are run and delivered from virtual machines rather than standalone, dedicated systems. As earlier stated, this allows the service provider to extend and cross-sell the benefits of virtualized network security across other managed services such as data center solutions, application performance management solutions, fixed-mobile convergence, and unified collaboration and communication solutions.

Fig. 4: Extending the Benefits of a Flexible Architecture to Horizontal Bundling

Cisco Managed Hosted Security Services address application security within the broader overall security context, and include both a robust architectural approach as well as a range of service delivery solutions. The Cisco offering spans a continuum of cost-effective and flexible Cisco security solutions built on virtualized infrastructure that can be delivered dynamically in a "mix-and-match" approach to meet evolving threats and hybrid customer needs. Cisco Managed Hosted Security Services help service providers transform their Service Delivery Data Center, or Service Delivery Center (SDC), into a dynamic, scalable, resilient environment. The SDC encompasses the service provider's complete virtualized infrastructure: the network, data center, IT, storage, application, and compute resources. Service providers can leverage this investment to deliver a complete portfolio of monetizable security and non-security managed services.

The SDC model helps service providers move up the customer value stack: from foundation security, to transport-aware security, right through to application-aware security. Service providers can adopt service delivery models – premises-based, network-based, data-center-based, or hybrid – that suit their target markets and customers.

Check Your Progress 3

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) Discuss Service Delivery Model.

.....

.....

.....

.....

2) What do you mean by Nexus in market and Service Packing?

.....

.....

.....

.....

1.9 LET US SUM UP

This unit helps service provider security-portfolio managers to address Critical application security requirements within their customer base. It also presents an

overview of the current trends in the application security market, important drivers and inhibitors of application security, the evolving market landscape, and service provider delivery models for application security solutions. Additionally, it examines the relevance of the network for these solutions, and offers considerations for security professionals involved in solution implementations.

1.10 CHECK YOUR PROGRESS: THE KEY

Check Your Progress 1

1) Security is defined as "the quality or state of being secure—to be free from danger." Security is often achieved by means of several strategies usually undertaken simultaneously or used in combination with one another. Specialized areas of security are Physical security, Personal security, Operations security, Communications security, Network security, Information security. The components are Confidentiality, Integrity and Availability. The two largest vulnerability categories are web applications and PC software. Web application vulnerabilities, which are typically remotely exploitable in nature, constitute the fastest growing and largest segment of the overall vulnerability count.

2) Web application vulnerabilities, which are typically remotely exploitable in nature, constitute the fastest growing and largest segment of the overall vulnerability count. In 2008, remotely exploitable vulnerabilities represented 90.2 percent of all vulnerabilities, up from about 85 percent in 2005. The most common attack techniques were SQL injection, cross-site scripting, and file-include.

3) There are many definitions of what constitutes application security. Many industry sources utilize the application lifecycle approach that emphasizes application fortification throughout the design, development, deployment, upgrade, and maintenance phases. Because providers of managed services are primarily involved at the front end of the application lifecycle (deployment and upgrade), this paper focuses on these stages. Within this scope, the definition of application security will be regarded as encompassing the use of software, hardware, policies, and procedural methods to protect business applications and data, either in static or dynamic form, from internal and external threats.

Check Your Progress 2

1) Forces are:-

   - Aggressive cyber criminals
   - Permeation of web services, Web 2.0, and applications

- Outsourced IT models
- Cost pressures
- Compliance and regulatory requirements
- Network-based computing

2) Factor are:

- Lack of application security expertise
- Inertia and confusion around need for web application security
- Lack of push from leading security services players

Check Your Progress 3

1) Service delivery models greatly impact the overall value and effectiveness of the application security solution. Like many other business solutions, application security is currently delivered using one or a combination of the following models:

   Customer-premises-based, customer-managed

   • Hosted service

   • Managed service

   • Software-as-a-Service (SaaS)

2) Application security is also offered with other managed services that are commonly required within a particular market. Some providers also offer bundles that enrich broader service portfolios by combining application security with collaboration, unified communications, data center, virtualized servers, or other managed services. In doing so, the application-security-embedded service bundle enables service providers to offer a more complete service portfolio, increasing their value and relevance to their target customers.

1.11 SUGGESTED READINGS

- Application security: Find web application security vulnerabilities during every phase of the software development lifecycle, HP center
- Application Security Framework, Open Mobile Terminal Platform
- Improving Web Application Security: Threats and Countermeasures, published by Microsoft Corporation.

- http://www.parasoft.com/parasoft_security Parasoft Application Security Solution
- http://www.preemptive.com/application-protection.html Application Protection
- http://www.veracode.com/solutions Veracode Security Static Analysis Solutions
- Open Web Application Security Project
- Simon Higginson, Platform Security Concepts.
- Security Solutions patterns & practices Application Security Methodology.
- The Microsoft Security Development Lifecycle (SDL) patterns & practices Security Guidance for Applications
- The Web Application Security Consortium

UNIT 2 INPUT VALIDATION AND ENCODING

Structure

2.0 Introduction
2.1 Objectives
2.2 Basic Input Validation Approaches
  2.2.1 No Input Validation
  2.2.2 Blacklisting
  2.2.3 Whitelisting
  2.2.4 Sanitizing
  2.2.5 Mixed Validator Strategies
  2.2.6 Comparison and Suitability of Input Validation Approaches
2.3 Current Architectures for Input Validation
  2.3.1 Centralized Input and Data Validation
2.4 Guidelines for Input Validation
  2.4.1 Validate all Input
  2.4.2 Use your IDE for the Code
    2.4.2.1 STRUTS Input Validation
    2.4.2.2 ASP.NET Validator Controls
    2.4.2.3 Environmental Variables (Hidden Inputs)
    2.4.2.4 Process Attributes
  2.4.3 Test the Input Validation
2.5 Avoiding Input Validation
  2.5.1 Buffer Overflows
  2.5.2 Cross-site Scripting
  2.5.3 SQL Injection
  2.5.4 Canonicalization
2.6 Encoding
  2.6.1 The Basics
  2.6.2 Examples of Encoding
    2.6.2.1 ASCII
    2.6.2.2 Unicode
2.7 Output Encoding
2.8 Let Us Sum Up
2.9 Check Your Progress: The Key
2.10 Suggested Readings

2.0 INTRODUCTION

Increasingly, computer attackers are exploiting flaws in Web applications, exposing enterprises to significant threats, including Personally Identifiable

Information breaches and uploads of malware onto vulnerable corporate Websites for distribution to customer browsers. Many of these Web application vulnerabilities are a direct result of improper input validation which leads to numerous kinds of attacks, including cross-site scripting (XSS), SQL injection, command injection, buffer overflows and many others. This chapter describes some of the best defenses against such attacks, which every Web application developer should master.

Input validation is no new invention of security experts. It has been a best practice from the beginnings of programming to verify all data syntactically. Unfortunately in the times of "rapid prototyping" and "time to market" pressure this guideline has been lost widely. Even some modern development environments does not support input validation in a convincing way. Thus programmers have to find their own way of input validation.

Whenever an application needs to gather information from a user or a browser, that information must be validated carefully to remove potential attack strings. Likewise, data sent back from the Web server to a browser should be filtered to make sure that exploits that an attacker has managed to sneak onto a Web server aren't served back to unwitting consumers who visit the site.

As an example, consider a Web site that needs to gather from a user a specific name and age for processing. One alpha field and one numeric field would likely suffice to gather this information. Without proper validation, however, attackers might be able to use other characters, including semicolons, greater-than or less-than symbols, and quotation marks to exploit the application. If a software developer does not properly screen all forms of user input to prevent certain characters from being entered, the software may be exploitable in numerous different ways. Input validation acts as a shield by deflecting potentially malicious characters. Likewise, output filtering prevents the Website from shooting back an exploit to browsers. Protection in both directions is needed, and Web application developers are a crucial line of such defense.

Unfortunately, some software developers either leave out input-validation and output filtering code altogether, or they implement such code poorly so that it does not filter out a truly comprehensive set of potential attack characters. Also, there are dozens of ways to alter or encode data to dodge validation filters: UTF-8, Hex, Unicode, mixed case and many more. Noted Web app security guru RSnake has written a great Web page that shows some different encoding tricks designed to slip cross-site scripting attacks input-validation code http://ha.ckers.org/xss.html. With that overview in mind, let's look carefully at the methods that developers should employ to prevent input-validation attacks.

2.1 OBJECTIVES

After studying this unit, you should be able to explain:

- what is input validation;
- different basic approaches of input validation;
- impacts of ignoring input validation;
- architecture of input validation;
- use of programming environment for input validation;
- encoding for secure data; and
- learning ASCII and unicode.

2.2 BASIC INPUT VALIDATION APPROACHES

Input validation is an important task in Web application development. It includes checking if there is a value in a mandatory field, whether a date or a number was entered in the correct format, etc. There are various basic approaches of input validation and some of the approaches are discussed in this section:

2.2.1 No Input Validation

In general it is not recommended to skip validation. There is an exception of this rule. If the design ensures that the input has been successfully validated before then it is acceptable to skip the input validation. However as a basic principle each component should be able to defend itself. A component may skip input validation only if it has a trust in the correctness of the input validation of the delivering component. This is a classical example of the design by contract methodology.

Each component has preconditions which must be satisfied by the caller. In this case the precondition would be "all input delivered to the called function has been successfully validated by the calling function". This assumption must hold during over the complete software (component) lifecycle. In particular reuse of components is susceptible for the violation of mutual contracts.

1. The change management in the software (component) lifecycle must guarantee that all changes of the contract must be inherited to all contracts with dependable modules.

2. New callable functions in the backend can require changes in the input validation of the calling functions.

3. The validating function must "know" what malicious input is for all subsequent callable functions.

These are three strong requirements which are difficult to assure over long times. Furthermore such strong requirements between functional unrelated modules are undesirably because they make difficult the decoupling of components.

2.2.2 Blacklisting

The term "blacklisting" means to filter evil input only. Unfortunately "evil input" can not be exactly defined. Usually blacklisting is based on the detection of possibly or probably or certain malicious input. The most common example is filtering of the JavaScript tag <script> to avoid Cross Site Scripting. This inhibits code injections like

<script>alert("You have been hacked!"</script>.

Blacklisting does not enjoy a good reputation. There are some inevitable problems which have no satisfying solution.

1. One can not define all evil input. Only known attacks can be described in patterns. New attacks must be detected, analyzed, described in corresponding patterns, and afterwards implemented. This is the same awkward position of the antivirus industry: "You can only defeat what you know."

2. A famous motto of Perl programmers is "There's more than one way to do it." This holds for the bad guys too. They a very clever to develop circumventions of blacklists. For example an ASP.NET page should throw an exception if an input form field is filled with "<script>". In ASP.NET 1.1 this doesn't hold for "<%00script>" but browser render both strings in the same matter such that an attacker can use <%00script> to perform a XSS attack.

Nevertheless, blacklisting is very popular. It can partly defend against specific known threats. Sometimes it is the only possibility. If your application server has a well known but not yet patched vulnerability then blacklisting the attack at your bastion host can be an appropriate countermeasure.

Some vendors have implemented default blacklists. For example in ASP.NET all requests are checked with a hard coded list of malicious strings. Obviously this concept is deficient by design but it could be the starting point of a "poor man's blacklist".

Sometimes Blacklisting is an appropriate approach. It can be used to fill the gap between the release of a so called "Zero Day Exploit" and the release of a well tested patch. Often such an exploit is identified by a characteristic attack

pattern which can filter. This is not a solution for ages because later exploits can and will vary.

However blacklisting can be a countermeasure to gain time for your developers fixing the real problem.

A similar motivation for blacklisting is defer patching to a regular maintenance slot.

Sometimes an implemented blacklisting will be used as a replacement for a whitelisting. Only for the sake of completeness we remark that you should consider blacklisting only as a workaround of inferior quality. Avoid Blacklisting if you can do better. Whitelisting is preferable from the viewpoint of security. However blacklisting is much better than no input validation.

2.2.3 Whitelisting

Whitelisting is the recommended best practice for input validation. All input may only pass if it is validated as known good input. This is also called positive input validation in contrary to the negative validation of the blacklisting. This is strong requirement and in real world application not easy to implemented. The simplistic standard example is the positive validation of a ZIP with a plain regular expression.

Example: the regular expression

^\d{5}$

can be used to validate a German zip code consisting of five digits.

The basic idea of whitelisting is that validated input is classified as non-malicious. There is a consensus in the community to consider it as the most secure approach. On the other hand the experiences from security audits show that sufficient implementations are not very common.

There are some reasons for that.

1. A successful whitelisting requires a very exact knowledge and rigid formal definition of the business data. In the best case there is a formal specification of the expected input. Otherwise they must to be provided in the business requirements of the application. Such detailed documents can be an expensive and require time consuming discussions. Changes in the requirements cause usually changes in the input validation and test cases.

2. The description patterns for whitelist entries are usually regular expression. It is not efficient to code regular expression one-to-one to input fields. Thus there is a need for an input validation design pattern or framework.

3. There is no classical design pattern for input validation. The design of a strict and comprehensive whitelisting is difficult. For example there are

software components which are only used to transmit data. They can not decide for themselves what is valid or not with respect to following components. A validation in front of these components mirrors some parts of the business logic in the validation logic. This is a great effort, difficult to maintain and collides possibly with the separation of tiers in multi tier architectures. The responsibility for input validation shifts from business logic to gateway components. Errors and false positives can be hard to track in complex software projects.

4. It is easy to test blacklisting with known attacks. It is more difficult to test whitelisting against false rejections and against false acceptance in a rigid way.

5. Check the input length every time and restrict it to the shortest possible value.

2.2.4 Sanitizing

Sanitizing is the approach to defuse a possibly malicious input and to replace it by non malicious input. Example: you have a web application which is vulnerable to SQL code injection but there is some reason that you can not fix the problem at the data access layer. A simple and strict whitelisting excludes all SQL meta characters is not favored because (for example) the name "O'Relly" would be rejected. Obviously "O'Relly" is a legitimate and non malicious input. A sanitize filter could omit or replace the apostrophe with a description. Example

"O'Relly" will be transformed to "ORelly" or

"O'Relly" will be transformed to "O[apostroph]Relly".

Both possibilities damage the integrity of the input but that is the purpose of sanitizing.

Both possibilities are human readable thus they could be completely sufficient for a customer care agent for example. The second one is reversible which can be helpful for some purposes.

Please keep in mind the followings:-

1. Avoid sanitizing if possible.

2. If you can not avoid sanitizing do it first and whitelist it second.

2.2.5 Mixed Validation Strategies

In real life the situation is often more complex. Usually a real life application comprises several components, some of them are self developed products, and other components are black box components by miscellaneous vendors. In such

cases one can use a mix of the approaches above. Here we recommend to ask an experienced web application security consultant.

2.2.6 Comparison and Suitability of Input Validation Approaches

This principle is certainly not a silver bullet but if you ensure that all of the data received and processed by your application is sufficiently validated you can go along way towards preventing many of the common vulnerabilities being actively exploited by malicious users. It is important for you to understand what data your application should accept, what its syntax should be and its minimum and maximum lengths. This information will allow you to define a set of "known good" values for every entry point that externally supplied data could exist.

Two main approaches exist for input validation called whitelisting and blacklisting respectively. It would be wrong to suggest either of these approaches is always the right answer but it is largely accepted that validating inputs against whitelists will be the most secure option. A whitelist will allow you to define what data should be accepted by your application for a given input point, in short you define a set of "known good inputs". The blacklist approach will attempt to do the opposite by defining a set of "known bad inputs" which requires the developer to understand a wide range of potentially malicious inputs.

A simple regular expression used for whitelisting a credit card number input is shown below:

^\\d{12,16}$

This will ensure that any data received in this input point is a number (\\d = 0-9) with a minimum length of 12 and a maximum of 16 ({12,16}). Although this is a simple example it clearly demonstrates the power of whitelist validation techniques because this input point will now prevent many common attacks.

The blacklisting approach will try to identify potentially malicious inputs and then replace or remove them. The example shown below will search the data received through an input point and replace any single quotes with a double quote.

s.replaceAll(Pattern.quote(" "), Matcher.quoteReplacement(" " "));

The blacklisting approach is often avoided where possible because it only protects against threats the developer could think of at the time of its creation. This means the blacklist might miss new attack vectors and have higher maintenance costs when compared to a whitelist.

Input Validation best practices:

- Apply whitelists (known good values) where possible.
- Canonicalise all inputs. This means reducing the data received to its simplest form, if the validation functions only searches for UTF-8 input an attacker could use another encoding method, like UTF-16, to encode the malicious characters and bypass the validation function.
- Check for content (i.e. 0-9), minimum and maximum lengths and correct syntax of all inputs.

2.3 CURRENT ARCHITECTURES FOR INPUT VALIDATION

Input validation can be done with many different approaches. The most basic way to validate input is to call explicitly a regular expression matching routine. This simple implementation can be sufficient in small projects but it lacks of checking for completeness and maintainability.

A better solution is centralized validation function or module. But both approaches require the possession of the sources. Another method is the filtering of the data stream before the reaches the application.

This can be done inside the hosting web server or by dedicated reverse proxy web servers or appliances. The major advantage of these external validations is the "black box" capability. It can be implemented afterwards and for closed source software. However, the training of such an adaptive filter is challenging with respect to the false positive and false rejection rates.

In the following table we summarize the most important characteristics.

Table 1: Characteristics of popular input validation approaches

Validation method or component / Evaluation Criteria	Validation in front of the application and before the input data will arrive the web server	Inside the web server but before the "coded" part of the application	Inside the application with central modules	Within the code "form by form"
Examples	Web application firewalls (WAF)	WAF web-server plugins Apache's mod_security	Using the validation routines in Struts	Using validation controls in ASP.NET
Typical purpose	Protection of Common of the Shelf (COTS) web applications	Protection of COTS web application but one has control about the webserver	Self developed applications based on a framework or with validation services	Self developed applications based but not based on a framework

Individual advantages of such a solution	A web application firewall (WAF) can implemented independently of the underlying business application. There are no (or only few) changed necessary.	Similar to the WAF scenario but one can spare an additional proxy. This improves performance, maintaining and costs.	Central validation control point. It is easy to check that all input will be validated. Changes in the whitelist can implemented at a single point.	Easy to implement. Sufficient and rigid approach for small and middle size applications.
Individual disadvantages of such a solution	· Requires additional hardware and software · Reduces performance and stability · Needs extensive training of the WAF No programmatic Interaction with the application.	· Increases slightly the complexity of the workflow Reduces slightly performance and stability Needs extensive training of the filter. No programmatic Interaction with the application.	· Only applicable when such frameworks are used. There is no well accepted design pattern for input validation.	It is difficult to verify the completeness and rigidity of the validation. Changes of business data specification can result in complex refactoring.
Validation method or component / Evaluation Criteria	Validation in front of the application and before the input data will arrive the web server	Inside the web server but before the "coded" part of the application	Inside the application with central modules	Within the code "form by form"
Occurs for the first time in which phase of software lifecycle	· Testing or · Production	· Testing or · Production	· Design	· Implementation
Performance Issues	Moderate: "yet another proxy"	Good	Good	Good
False positive Rate	Annoying	Annoying	Low	Low
Request specific error handling	None	None	Possible	Possible
Protected components	· Application · Web server	Application	Application	Application

i. Centralized input and data validation

Centralizing input validation helps ensure that data is validated in a consistent way throughout the application and provides a single point of maintenance. Perform the following steps to assure that all input is validated:

1. **Centralize validation.** When you develop an input- and data-validation architecture for your application, consider developing a library of validation routines in all but the smallest applications. This will help ensure that data is validated in a consistent way throughout the application and provide a single point of maintenance. You need to trace data from entry point to exit point to know how it should be validated. A good library includes routines for all of the different types of validation you need to apply, and these can be used in combination if necessary.

2. **Constrain, reject, and sanitize input.** Constrain what you allow from the beginning. It is much easier to validate data for known valid types, patterns, and ranges (using a white list) than it is to validate data by looking for known bad characters (using a black list). When you design your application, you know what your application expects. The range of valid data is generally a more finite set than the range of potentially malicious input. However, for added defense you might want to reject known bad input and then sanitize the input. Constrain input for type, length, format, and range. Use regular expressions to help constrain text input. Use strong data typing where possible.

3. **Identify trust boundaries.** Ensure that entry points between trust boundaries validate all input data explicitly. Make no assumptions about the data. The only exception is inside a routine that you know can only be called by other routines within the same trust boundary.

Check Your Progress 1

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) Explain advantage of Blacklisting over No input validation.

.....

.....

.....

.....

2) Explain the advantages of Whitelistng.

.....

.....

.....

.....

3) Discuss any two evaluation criterias.

.....

.....

.....

.....

4) Which strategy should be used for complex real life applications?

.....

.....

.....

.....

2.4 GUIDELINES FOR INPUT VALIDATION

Followings are the guidelines for validation of inputs:

2.4.1 Validate All Input

This is the most important rule. There are many input types which can be used to attack an application:

- GET parameters
- Forms fields (hidden fields)
- Selection lists and drop down lists
- File Upload (file name and file content)
- Cookies
- HTTP-headers
- Java applet communication
- Flat file import

GET parameters and form field are the most obvious inputs for web applications. Also back posts for selections lists and drop down lists must be validated to avoid SQL injections for example.

It is not obvious how to validate file uploads in a perfect manner. At least the pathname must be handled with special care. The file extension should be restricted to a reasonable and short list of acceptable file types. The validation of the path component of the file name must prevent arbitrary access to the file system (example "c:\windows\system32\trojan_horse.exe") and path climbing (example "..\..\..\system32\evil_driver.dll"). Usually the correctness of the

content of a file is very difficult to verify because the specification of file formats are complex or even unknown (for example .pdf or .ppt).

In general an application should avoid writing and reading of own specified cookies. The use of cookies for session management is sound but this is a task for the application server framework. Usually these frameworks support the saving of session parameters as key value pairs in a session object in an easily manner.

This object is stored internally at the server and not at the client. Thus there is no need to validate keys and values at every request. In contrast a validation has to be performed if session parameters are stored in cookies or hidden field. Transmitting session data to the client increases the threat of data spoofing.

There a two important exceptions for the strict rule of input validation. If a module hands over input one to one to another module then it can skip input validation. However the module must protect itself.

The second exception is input from trusted sources. The notion of a "trusted source" is a controversial issue. It requires extensive control about the source. The trust in a source can possibly change in the life cycle of the source or by reuse of the component. In general we do not recommend this concept. In any case a trusted source requires a diligent threat modeling.

2.4.2 Use Your IDE for the Code

A developer should not spend too much time with input validation coding issues. Frameworks like STRUTS or ASP.NET support developer friendly input validation. We present some examples here:

2.4.2.1 STRUTS Input Validation

In the following example (adaptation of the original documentation10) you can see how input validation in STRUTS works. The code is nearly self explanatory.

```
<field property="lastName" depends="required,mask">

<msg name="mask"

key="registrationForm.lastname.maskmsg"/>

<arg position="0" key="registrationForm.lastname.displayname"/>

<var>

<var-name>mask</var-name>

<var-value>^[a-zA-Z]*$</var-value>
```

</var>

</field>

Example 2: Using a variable for an regular expression

The usage of variables for regular expressions enables very efficient validation solutions. For example a business data dictionary could contain all business data types with their regular expression and the name of their variable. A developer can reference to these variables names for validation purposes.

2.4.2.2 ASP.NET Validator Controls

Microsoft's ASP.NET contains very powerful validators for input validation which can be easily integrated in their development environment Visual Studio 2005. In particular, the Regular Expression Validator is a useful tool from the viewpoint of security.

<asp:RegularExpressionValidator

ID="RegularExpressionValidator1"

runat="server"

ControlToValidate="txtUserName"

ErrorMessage="Username is not valid."

ValidationExpression="\w{1,10}">

</asp:RegularExpressionValidator>

Example 3: Input validation in ASP.NET using the Regular Expression Validator control

The validation expression in this example allows all usernames with one up to ten word characters ("\w"). There is a pitfall while using ASP.NET validators. The binding of a validator control to a form control does not change the execution flow of the page at the server side. The only effect is the setting of the "IsValid" property of the respective validation control. The developer must query this property to control the execution of the page. Additionally the validation of the client side JavaScript code can pretend a secure validation while the server side validation is still missing. ASP.NET validation controls validate input mandatory at the server side but they can generate additional client side JavaScript validation code. Unfortunately, the interpretation of regular expressions in JavaScript is slightly different from the interpretation in more powerful regular expression server machines.

In particular, internationalization and Unicode causes serious troubles. We recommend to tailor the validation by client side JavaScript manually in complex situations.

2.4.2.3 Environmental Variables (Hidden Inputs)

Environment variables are "hidden" inputs. They exist and affect the behaviors of programs. Ignoring their existence during programming can lead to security breaches.

- **PATH**
  - When running a command in a shell, the shell searches for the command using the **PATH** environment variable.
  - What would happen in the following?

    system("mail");

  - The attacker can change PATH to the following, and cause "mail" in the current directory to be executed.

    PATH=".:$PATH"; export PATH

- **IFS**
  - The IFS variable determines the characters which are to be interpreted as whitespace. It stands for Internal Field Separators. Suppose we set this to include the forward slash character:

    IFS="/ \t\n"; export IFS

    PATH=".:$PATH"; export PATH

  - Now call any program which uses an absolute PATH from a Bourne shell (e.g. system(), or popen() system calls). This is now interpreted like the following that would attempt to execute a command called bin in the current directory of the user.

    system("/bin/mail root"); ---> system(" bin mail root");

  - The IFS bug has pretty much been disallowed in shells now.
- **LD_LIBRARY_PATH**
  - Dynamic library directories: When searching for dynamic libaries, UNIX systems tend to look for libraries to load in a seach path provided by this environment variable.

- Virtually every Unix program depends on libc.so and virtually every windows program relies on DLL's. If these libraries become exchanged with Trojan horses many things can go wrong.
- Attackers can modify this path and cause the program load the attackers' libraries.

```
setenv LD_LIBRARY_PATH /tmp:$LD_LIBRARY_PATH
```

or the user's current directory

```
setenv LD_LIBRARY_PATH .:$LD_LIBRARY_PATH
```

- Most modern C runtime libraries have fixed this problem by ignoring the LD_LIBRARY_PATH variable when the EUID is not equal to the UID or the EGID is not equal to the GID.
- Secure applications can be linked *statically* with a trusted library to avoid this.
- In Windows machines, when loading DLLs, generally, the current directory is searched for DLLs before the system directories. If you click on a Microsoft Word document to start Office, the directory containing that document is searched first for DLLs.

❖ **LD_Preload**

- Many UNIX systems allow you to "pre-load" shared libraries by setting an environment variable LD_PRELOAD. This allows you to do interesting things like replace standard C library functions or even the C interfaces to system calls with your own functions.
- Modern systems ignore LD_PRELOAD if the program is a setuid program.

```
% cc -o malloc_interposer.so -G -Kpic malloc_interposer.c

% setenv LD_PRELOAD $cwd/malloc_interposer.so
```

- How to get rid of environment variables?

```
extern char **environ;

int main(int argc, char **argv)

{environ = 0;}
```

- The above strategy doesn't necessarily work for every program. For example, loading shared libraries at runtime needs LD_LIBRARY_PATH.

- **vi vulnerability**
  - Behavior:
    - vi file
    - hang up without saving it
    - vi invokes expreserve, which saves buffer in protected area
    - expreserve invokes mail to send a mail to user
  - Facts:
    - expreserve is a setuid program, and mail is called with root privilege.
    - expreserve uses system("mail user") or system("/bin/mail user");
    - expreserve does not take care of the environment variables.
  - Attack:
    - Change PATH, IFS

      IFS="/binal\t\n" causes "m" be invoked, instead of "/bin/mail"

2.4.2.4 Process Attributes

- **umask** value
  - It decides the default permission of newly created files.
  - A child process inherits this value from its parent.
  - Consider this situation:

A set-UID program stores temporary data in a /tmp/tempfile. The integrity of this file is important. If the programmer assumes that umask value is 077, the assumption might fail.

The attacker can run this program from its shell, and the set-UID will inherit the umask value from the shell.

How to secure it: either explicitly set the umask value (using umask(077)) or explicitly set the permission of the newly created file (using chmod("newfile",0755));

- Core Dump
- If your program holds sensitive data like unencrypted passwords, then you should forbid the process from being core dumped.

- How to disable core dumps?

```
#include <sys/time.h>
#include <sys/resource.h>
#include <unistd.h>
Int main(int argc, char **argv)
{struct rlimit rlim;
        getrlimit(RLIMIT_CORE, &rlim);
        rlim.rlim_max = rlim.rlim_cur = 0; if
        (setrlimit(RLIMIT_CORE, &rlim)) {exit(-1); } ... return 0; }
```

Solaris by default (at least in Solaris 8) does not allow setuid processes to core dump for obvious security reasons.

2.4.3 Test the Input Validation

It is very important to test your input validation. We give here only short list of different aspects in validation testing:

- Test for false positives: Your business must not generate validation errors.
- Test for false negatives: Test with invalid data to check the correctness of the validation routines.
- Test with and without JavaScript: As described above JavaScript interprets regular expression slightly different compared with server engines. These differences can generate false positives at the client side.
- Perform a test **without** any validation at the client and the server. This test can expose insecurities in other tiers of the application. Some developers rely on the correct validation of the application frontend which is not sufficient.

Check Your Progress 2

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) Give various input types which can be used to attack an application.

.....

.....

.....

.....

2) Give an example of STRUTS input validation.

.....

.....

.....

.....

3) How PATH environmental variable can be used to attack?

.....

.....

.....

.....

4) Discuss different aspects of validation testing.

.....

.....

.....

.....

2.5 AVOIDING INPUT VALIDATION

Input validation is a security issue if an attacker discovers that your application makes unfounded assumptions about the type, length, format, or range of input data. The attacker can then supply carefully crafted input that compromises your application.

When network and host level entry points are fully secured; the public interfaces exposed by your application become the only source of attack. The input to your application is a means to both test your system and a way to execute code on an attacker's behalf. Does your application blindly trust input? If it does, your application may be susceptible to the following:

2.5.1 Buffer Overflows

Buffer overflow vulnerabilities can lead to denial of service attacks or code injection. A denial of service attack causes a process crash; code injection alters the program execution address to run an attacker's injected code. The following code fragment illustrates a common example of buffer overflow vulnerability.

javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl04');Copy Code
void SomeFunction(char *pszInput)

{char szBuffer[10]; // Input is copied straight into the buffer when no type checking is performed strcpy(szBuffer, pszInput); . . . }

Managed .NET code is not susceptible to this problem because array bounds are automatically checked whenever an array is accessed. This makes the threat of buffer overflow attacks on managed code much less of an issue. It is still a concern, however, especially where managed code calls unmanaged APIs or COM objects.

Countermeasures to help prevent buffer overflows include:

- Perform thorough input validation. This is the first line of defense against buffer overflows. Although a bug may exist in your application that permits expected input to reach beyond the bounds of a container, unexpected input will be the primary cause of this vulnerability. Constrain input by validating it for type, length, format and range.
- When possible, limit your application's use of unmanaged code, and thoroughly inspect the unmanaged APIs to ensure that input is properly validated.
- Inspect the managed code that calls the unmanaged API to ensure that only appropriate values can be passed as parameters to the unmanaged API.
- Use the /GS flag to compile code developed with the Microsoft Visual C++ development system. The /GS flag causes the compiler to inject security checks into the compiled code. This is not a fail-proof solution or a replacement for your specific validation code; it does, however, protect your code from commonly known buffer overflow attacks. For more information, see the .NET Framework Product documentation.

Example of Code Injection through Buffer Overflows

An attacker can exploit a buffer overflow vulnerability to inject code. With this attack, a malicious user exploits an unchecked buffer in a process by supplying a carefully constructed input value that overwrites the program's stack and alters a function's return address. This causes execution to jump to the attacker's injected code.

The attacker's code usually ends up running under the process security context. This emphasizes the importance of using least privileged process accounts. If the current thread is impersonating, the attacker's code ends up running under the security context defined by the thread impersonation token. The first thing an attacker usually does is call the **RevertToSelf** API to revert to the process level security context that the attacker hopes has higher privileges.

Make sure you validate input for type and length, especially before you call unmanaged code because unmanaged code is particularly susceptible to buffer overflows.

2.5.2 Cross-Site Scripting

An XSS attack can cause arbitrary code to run in a user's browser while the browser is connected to a trusted Web site. The attack targets your application's users and not the application itself, but it uses your application as the vehicle for the attack.

Because the script code is downloaded by the browser from a trusted site, the browser has no way of knowing that the code is not legitimate. Internet Explorer security zones provide no defense. Since the attacker's code has access to the cookies associated with the trusted site and are stored on the user's local computer, a user's authentication cookies are typically the target of attack.

Example of Cross-Site Scripting

To initiate the attack, the attacker must convince the user to click on a carefully crafted hyperlink, for example, by embedding a link in an email sent to the user or by adding a malicious link to a newsgroup posting. The link points to a vulnerable page in your application that echoes the unvalidated input back to the browser in the HTML output stream. For example, consider the following two links.

Here is a legitimate link:

javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl06');Copy Code
www.yourwebapplication.com/logon.aspx?username=bob

Here is a malicious link:

javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl07');Copy Code
www.yourwebapplication.com/logon.aspx?username=<script>alert('hacker code')</script>

If the Web application takes the query string, fails to properly validate it, and then returns it to the browser, the script code executes in the browser. The preceding example displays a harmless pop-up message. With the appropriate script, the attacker can easily extract the user's authentication cookie, post it to his site, and subsequently make a request to the target Web site as the authenticated user.

Countermeasures to prevent XSS include:

- Perform thorough input validation. Your applications must ensure that input from query strings, form fields, and cookies are valid for the application. Consider all user input as possibly malicious, and filter or

sanitize for the context of the downstream code. Validate all input for known valid values and then reject all other input. Use regular expressions to validate input data received via HTML form fields, cookies, and query strings.

- Use **HTMLEncode** and **URLEncode** functions to encode any output that includes user input. This converts executable script into harmless HTML.

2.5.3 SQL Injection

SQL Injection Attacks by Example

- SQL injection is a technique for exploiting web applications that use client-supplied data in SQL queries, but without first stripping potentially harmful characters. As results, the web applications might run SQL code that was not intended.

- Some applications get users' inputs from a web form, and then construct a SQL query directly using the users' input. For example, the following SQL query is constructed using the value of **$EMAIL** submitted on the form by users:

  SELECT *email, passwd, login_id, full_name*

  FROM *table*

  WHERE *email* = '**$EMAIL**';

- The above application is commonly used when members of online account forget their password. They just need to type their email addresses; if the email address is in the database (the user is a member), the password of that email will be sent to that email address. **The goal of SQL inject attack in this example is to be able to log into the online account without being a member.**

- Guessing field name: the first steps are to guess some fields names of the database.

  - The following guess **email** as the name of a field.

  - If we get a server error, it means our SQL is malformed and a syntax error was thrown: it's most likely due to a bad field name. If we get any kind of valid response, we guessed the name correctly.

This is the case whether we get the "email unknown" or "password was sent" response.

SELECT *fieldlist*

FROM *table*

WHERE *field* = **'x' AND email IS NULL; --**';

- Guessing the table name
  - Similarly, if messages says "email unknown" or "password was sent", we know that our guess is correct.

    SELECT email, passwd, login_id, full_name

    FROM *table*

    WHERE **email = 'x' AND 1=(SELECT COUNT(*) FROM *tabname*); --**';

  - However, the above only confirms that **tabname** is valid name, not necessary the one that we are using. The following will help.

    SELECT email, passwd, login_id, full_name

    FROM members

    WHERE email = **'x' AND members.email IS**

    NULL; --';

- Guessing a member's email address: **$EMAIL = x' OR full_name LIKE '%Bob%**
  - If the SQL is successful, usually you will see a message like this: "We sent your password to <…>", where <…> is the email address whose full_name matches with %Bob% (% is a wildcard)

    SELECT email, passwd, login_id, full_name

    FROM members

    WHERE email = **'x' OR full_name LIKE**

    '%Bob%';

- Brute-force password guessing (after we have learned a valid email address)

  SELECT email, passwd, login_id, full_name

  FROM members

  WHERE email = **'bob@example.com' AND passwd =**

  'hello123';

- If the database isn't readonly, we can try the following to add a new member:

- The "- -" at the end marks the start of an SQL comment. This is an effective way to consume the final quote provided by application and not worry about matching them.
- There might be some challenges doing so:
  - The web form might not give you enough room to type the entire string.
  - The web application user might not have **INSERT** permission on the **members** table.

The application might not behave well because we haven't provided values for other fields.

- A valid "member" might require not only a record in the **members** table, but associated information in other tables (e.g. **accessrights**), so adding to one table alone might not be sufficient.

SELECT email, passwd, login_id, full_name

FROM members

WHERE email = **'x';**

INSERT INTO members ('email','passwd','login_id','full_name') VALUES ('xyz@hacker.net','hello','xyz','xyz Hacker');--';

❖ Modify an existing member's email address

- If this is successful, the attacker can now go to the regular "I lost my password" link, type the updated email address, and receive Bob's password in the email.

SELECT email, passwd, login_id, full_name

FROM members

WHERE email = **'x';**

UPDATE members

SET email = 'xyz@hacker.net'

WHERE email = 'bob@example.com';

❖ How to prevent SQL injection attacks?

- Sanitize the input
- Configure error reporting: the above attacks take advantage of the error messages returned by the sever. It can make attacker's life more difficult

by not telling the users the actual error message in the SQL query. For example, you can just say "something is wrong".

- Use bound parameters, so user's input is simply treated as data; quotes, semicolons, backslashes, and SQL comment notation will have no impact, because they are treated as just data, and will not be parsed by SQL.

See the following Java code:

Insecure version

```
Statement s = connection.createStatement();

ResultSet rs = s.executeQuery("SELECT email FROM member WHERE name = " + formField);
```

Secure version

```
PreparedStatement ps = connection.prepareStatement(

"SELECT email FROM member WHERE name = ?");

ps.setString(1, formField);

ResultSet rs = ps.executeQuery();
```

2.5.4 Canonicalization

Different forms of input that resolve to the same standard name (the canonical name), is referred to as *canonicalization*. Code is particularly susceptible to canonicalization issues if it makes security decisions based on the name of a resource that is passed to the program as input. Files, paths, and URLs are resource types that are vulnerable to canonicalization because in each case there are many different ways to represent the same name. File names are also problematic. For example, a single file could be represented as:

javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl13');Copy Code

c:\temp\somefile.dat

somefile.dat

c:\temp\subdir\..\somefile.dat

c:\ temp\ somefile.dat

..\somefile.dat

Ideally, your code should not accept input file names. If it does, the name should be converted to its canonical form prior to making security decisions, such as whether access should be granted or denied to the specified file.

Countermeasures to address canonicalization issues include:

- Avoid using file names as input where possible and instead use absolute file paths that cannot be changed by the end user.
- Make sure that file names are well formed (if you must accept file names as input) and validate them within the context of your application. For example, check that they are within your application's directory hierarchy.
- Ensure that the character encoding is set correctly to limit how input can be represented. Check that your application's Web.config has set the **requestEncoding** and **responseEncoding** attributes on the **<globalization>** element.

2.6 ENCODING

2.6.1 The Basics

In computers and in data transmission between them, i.e. in digital data processing and transfer, data is internally presented as octets, as a rule. An *octet* is a small unit of data with a numerical value between 0 and 255, inclusively. The numerical values are presented in the normal (decimal) notation here, but notice that other presentations are used too, especially octal (base 8) or hexadecimal (base 16) notation. Octets are often called *bytes*, but in principle, octet is a more definite concept than byte. Internally, octets consist of eight bits (hence the name, from Latin *octo* 'eight'), but we need not go into bit level here. However, you might need to know what the phrase "first bit set" or "sign bit set" means, since it is often used. In terms of numerical values of octets, it means that the value is greater than 127. In various contexts, such octets are sometimes interpreted as *negative* numbers, and this may cause various problems.

Different conventions can be established as regards to how an octet or a sequence of octets presents some data. For instance, four consecutive octets often form a unit that presents a real number according to a specific standard. We are here interested in the presentation of character data (or string data; a *string* is a sequence of characters) only.

In the simplest case, which is still widely used, one octet corresponds to one character according to some mapping table (encoding). Naturally, this allows at most 256 different characters being represented. There are several different encodings, such as the well-known ASCII encoding and the ISO Latin family of encodings. The correct interpretation and processing of character data of course requires knowledge about the encoding used. For HTML documents, such information should be sent by the Web server along with the document itself, using so-called HTTP headers.

Previously the ASCII encoding was usually assumed by default (and it is still very common). Now a days ISO Latin 1, which can be regarded as an extension of ASCII, is often the default. The current trend is to avoid giving such a special position to ISO Latin 1 among the variety of encodings.

Definitions

The following definitions are not universally accepted and used. In fact, one of the greatest causes of confusion around character set issues is that terminology varies and is sometimes misleading.

Character Repertoire

A set of distinct characters. No specific internal presentation in computers or data transfer is assumed. The repertoire per se does not even define an ordering for the characters; ordering for sorting and other purposes is to be specified separately. A character repertoire is usually defined by specifying names of characters and a sample (or reference) presentation of characters in visible form. Notice that a character repertoire may contain characters which *look* the same in some presentations but are regarded as logically distinct, such as Latin uppercase A, Cyrillic uppercase A, and Greek uppercase alpha. For more about this, see a discussion of the character concept later in this document.

Character Code

A mapping, often presented in tabular form, which defines a one-to-one correspondence between characters in a character repertoire and a set of nonnegative integers. That is, it assigns a unique numerical code, a *code position*, to each character in the repertoire. In addition to being often presented as one or more tables, the code as a whole can be regarded as a single table and the code positions as indexes. As synonyms for "code position", the following terms are also in use: *code number*, *code value*, *code element*, *code point*, *code set value* - and just *code*. Note: The set of nonnegative integers corresponding to characters need not consist of consecutive numbers; in fact, most character codes have "holes", such as code positions reserved for control functions or for eventual future use to be defined later.

Character Encoding

A method (algorithm) for presenting characters in digital form by mapping sequences of code numbers of characters into sequences of octets. In the simplest case, each character is mapped to an integer in the range 0 - 255 according to a character code and these are used as such as octets. Naturally, this only works for character repertoires with at most 256 characters. For larger sets, more complicated encodings are needed. Encodings have names, which can be registered.

Notice that a character code assumes or implicitly defines a character repertoire. A character encoding could, in principle, be viewed purely as a method of mapping a sequence of integers to a sequence of octets. However, quite often an encoding is specified in terms of a character code (and the implied character repertoire). The *logical* structure is still the following:

1. A character *repertoire* specifies a collection of characters, such as "a", "!", and "ä".

2. A character *code* defines numeric codes for characters in a repertoire. For example, in the ISO 10646 character code the numeric codes for "a", "!", "ä", and "‰" (per mille sign) are 97, 33, 228, and 8240. (Note: Especially the per mille sign, presenting $^{0}/_{00}$ as a single character, can be shown incorrectly on display or on paper. That would be an illustration of the symptoms of the problems we are discussing.)

3. A character *encoding* defines how sequences of numeric codes are presented as (i.e., mapped to) sequences of octets. In one possible encoding for ISO 10646, the string a!ä‰ is presented as the following sequence of octets (using two octets for each character): 0, 97, 0, 33, 0, 228, 32, 48.

2.6.2 Examples of Encoding

2.6.2.1 ASCII

The Basics of ASCII

The name *ASCII*, originally an abbreviation for "American Standard Code for Information Interchange", denotes an old character repertoire, code, and encoding.

Most character codes currently in use contain ASCII as their subset in some sense. ASCII is the safest character repertoire to be used in data transfer. However, not even all ASCII characters are "safe"!

ASCII has been used and is used so widely that often the word *ASCII* refers to "text" or "plain text" in general, even if the character code is something else! The words "ASCII file" quite often mean any text file as opposite to a binary file.

The definition of ASCII also specifies a set of control codes ("control characters") such as linefeed (LF) and escape (ESC). But the *character repertoire* proper, consisting of the *printable* characters of ASCII, is the following (where the first item is the blank, or space, character):

```
! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
```

P Q R S T U V W X Y Z [\ ] ^ _

` a b c d e f g h i j k l m n o

p q r s t u v w x y z { | } ~

The *appearance* of characters varies, of course, especially for some special characters. Some of the variation and other details are explained in *The ISO Latin 1 character repertoire - a description with usage notes*.

A Formal View on ASCII

The *character code* defined by the ASCII standard is the following: code values are assigned to characters consecutively in the order in which the characters are listed above (rowwise), starting from 32 (assigned to the blank) and ending up with 126 (assigned to the tilde character ~). Positions 0 through 31 and 127 are reserved for control codes. They have standardized names and descriptions, but in fact their usage varies a lot.

The *character encoding* specified by the ASCII standard is very simple, and the most obvious one for any character code where the code numbers do not exceed 255: each code number is presented as an octet with the same value.

Octets 128 - 255 are not used in ASCII. (This allows programs to use the first, most significant bit of an octet as a parity bit, for example.)

National Variants of ASCII

There are several national variants of ASCII. In such variants, some special characters have been replaced by national letters (and other symbols). There is great variation here, and even within one country and for one language there might be different variants. The original ASCII is therefore often referred to as *US-ASCII*; the formal standard (by ANSI) is *ANSI X3.4-1986*.

The phrase "original ASCII" is perhaps not quite adequate, since the creation of ASCII started in late 1950s, and several additions and modifications were made in the 1960s. The 1963 version had several unassigned code positions. The ANSI standard, where those positions were assigned, mainly to accommodate lower case letters, was approved in 1967/1968, later modified slightly. For the early history, including pre-ASCII character codes, ASCII: American Standard Code for Information Infiltration, and the computer history documents, including the background and creation of ASCII, written by Bob Bemer, "father of ASCII".

The international standard *ISO 646* defines a character set similar to US-ASCII but with code positions corresponding to US-ASCII characters @[\]{|} as "national use positions". It also gives some liberties with characters #$^`~. The standard also defines "international reference version (IRV)", which is (in the 1991 edition of ISO 646) identical to US-ASCII. Ecma International has issued

the ECMA-6 standard, which is equivalent to ISO 646 and is freely available on the Web.

Within the framework of ISO 646, and partly otherwise too, several "national variants of ASCII" have been defined, assigning different letters and symbols to the "national use" positions. Thus, the characters that appear in those positions - including those in US-ASCII - are somewhat "unsafe" in international data transfer, although this problem is losing significance. The trend is towards using the corresponding codes strictly for US-ASCII meanings; national characters are handled otherwise, giving them their own, unique and universal code positions in character codes larger than ASCII. But old software and devices may still reflect various "national variants of ASCII".

The following table lists ASCII characters which might be replaced by other characters in national variants of ASCII. (That is, the code positions of these US-ASCII characters might be occupied by other characters needed for national use.)

The lists of characters appearing in national variants are not intended to be exhaustive, just typical *examples*.

dec	oct	hex	glyph	official Unicode name	National variants
35	43	23	#	number sign	£ Ù
36	44	24	$	dollar sign	¤
64	100	40	@	commercial at	É § Ä à ³
91	133	5B	[	left square bracket	Ä Æ ° â ¡ ÿ é
92	134	5C	\	reverse solidus	Ö Ø ç Ñ ½ ¥
93	135	5D	]	right square bracket	Å Ü § ê é ¿ |
94	136	5E	^	circumflex accent	Ü î
95	137	5F	_	low line	è
96	140	60	`	grave accent	é ä µ ô ù
123	173	7B	{	left curly bracket	ä æ é à ° ¨
124	174	7C	|	vertical line	ö ø ù ò ñ *f*
125	175	7D	}	right curly bracket	å ü è ç ¼
126	176	7E	~	tilde	ü ¯ ß ¨ û ì ´ _

Almost all of the characters used in the national variants have been incorporated into ISO Latin 1. Systems that support ISO Latin 1 in principle may still reflect the use of national variants of ASCII in some details; for example, an ASCII character might get *printed* or *displayed* according to some national variant. Thus, even "plain ASCII text" is thereby not always portable from one system or application to another.

Subsets of ASCII for Safety

Mainly due to the "national variants" discussed above, some characters are less "safe" than other, i.e. more often transferred or interpreted incorrectly.

In addition to the letters of the English alphabet ("A" to "Z", and "a" to "z"), the digits ("0" to "9") and the space (" "), only the following characters can be regarded as really "safe" in data transmission:

! " % & ' ( ) * + , - . / : ; < = > ?

Even these characters might eventually be *interpreted* wrongly by the recipient, e.g. by a human reader seeing a glyph for "&" as something else than what it is intended to denote, or by a program interpreting "<" as starting some special markup, "?" as being a so-called wildcard character, etc.

When you need to *name* things (e.g. files, variables, data fields, etc.), it is often best to use only the characters listed above, even if a wider character repertoire is possible. Naturally you need to take into account any additional restrictions imposed by the applicable syntax. For example, the rules of a programming language might restrict the character repertoire in identifier names to letters, digits and one or two other characters.

2.6.2.2 Unicode

Unicode is a standard, by the Unicode Consortium, which defines a character repertoire and character code intended to be fully compatible with ISO 10646, and an encoding for it. Unicode was originally designed to be a 16-bit code, but it was extended so that currently code positions are expressed as integers in the hexadecimal range 0..10FFFF (decimal 0..1 114 111). That space is divided into 16-bit "planes". Until recently, the use of Unicode has mostly been limited to "Basic Multilingual Plane (BMP)" consisting of the range 0..FFFF.

Unicode *character repertoire* can be regarded as a *superset* of most character repertoires in use. However, the *code positions* of characters vary from one character code to another.

"Unicode" is the commonly used name

In practice, people usually talk about Unicode, partly because we prefer names to numbers, partly because Unicode is more explicit about the *meanings* of characters,

Encodings for Unicode

Originally, before extending the code range past 16 bits, the "native" Unicode encoding was *UCS-2*, which presents each code number as two consecutive octets *m* and *n* so that the number equals 256*m*+*n*. This means, to express it in computer jargon, that the code number is presented as a **two-byte integer**. According to the Unicode consortium, the term UCS-2 should now be avoided, as it is associated with the 16-bit limitations.

UTF-32 encodes each code position as a 32-bit binary integer, i.e. as four octets. This is a very obvious and simple encoding. However, it is inefficient in terms of the number of octets needed. If we have normal English text or other text which contains ISO Latin 1 characters only, the length of the Unicode encoded octet sequence is four times the length of the string in ISO 8859-1 encoding. UTF-32 is rarely used, except perhaps in internal operations (since it is very simple for the purposes of string processing).

UTF-16 represents each code position in the Basic Multilingual Plane as two octets. Other code positions are presented using so-called *surrogate pairs*, utilizing some code positions in the BMP reserved for the purpose. This, too, is a very simple encoding when the data contains BMP characters only.

Unicode can be, and often is, encoded in other ways, too, such as the following encodings:

UTF-8

Character codes less than 128 (effectively, the ASCII repertoire) are presented "as such", using one octet for each code (character) All other codes are presented, according to a relatively complicated method, so that one code (character) is presented as a sequence of two to four octets, each of which is in the range 128 - 255. This means that in a sequence of octets, octets in the range 0 - 127 ("bytes with most significant bit set to 0") directly represent ASCII characters, whereas octets in the range 128 - 255 ("bytes with most significant bit set to 1") are to be interpreted as really encoded presentations of characters.

UTF-7

Each character code is presented as a sequence of one or more octets in the range 0 - 127 ("bytes with most significant bit set to 0", or "seven-bit bytes", hence the name). Most ASCII characters are presented as such, each as one octet, but for obvious reasons some octet values must be reserved for use as

"escape" octets, specifying the octet together with a certain number of subsequent octets forms a multi-octet encoded presentation of one character.

2.7 OUTPUT ENCODING

Regardless of what the encoding is for your documents, an XSL engine can convert the output to a different encoding if you need it. When the document is loaded into memory, XML applications such as XSLT engines convert it to Unicode. The XSL engine then uses the stylesheet templates to create a transformed version of the content in memory structures. When it is done, it *serializes* the internal content into a stream of bytes that it feeds to the outside world. During the serialization process, it can convert the internal Unicode to some other encoding for the output.

An XSL stylesheet usually sets the output encoding in an xsl:output element at the top of the stylesheet file. The following shows that element for the html/docbook.xsl stylesheet:

```
<xsl:output method="html"
        encoding="ISO-8859-1"
        indent="no"/>
```

The encoding="ISO-8859-1" attribute means all documents processed with that stylesheet are to be output with the ISO-8859-1 encoding. If a stylesheet's xsl:output element does not have an encoding attribute, then the default output encoding is UTF-8. That is what the fo/docbook.xsl stylesheet for print output does.

When the output method="html", the XSLT processor also adds an HTML META tag that identifies the HTML file's encoding:

```
<meta content="text/html; charset=ISO-8859-1" http-equiv="Content-Type">
```

When a browser opens the HTML file, it reads this tag and knows the bytes it finds in the file map to the ISO-8859-1 character set for display. What if the document contains characters that are not available in the specified output encoding? As with input, the characters are expressed as numerical character references such as ™. It is up to the browser to figure out how to display such characters. Most browsers cover a pretty wide range of character entities, but there are so many that sometimes a browser does not have a way to display a given character.

Most modern graphical browsers can display HTML files encoded with UTF-8, which covers a much wider set of characters than ISO-8859-1. To change the output encoding for the non-chunking docbook.xsl stylesheet, you have to use a

stylesheet customization layer. That is because the XML specification does not permit the encoding attribute to be a variable or parameter value. Your stylesheet customization must provide a new <xsl:output> element such as the following:

```
<?xml version='1.0'?>

<xsl:stylesheet  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

        version="1.0">

<xsl:import href="/path/to/html/docbook.xsl"/>

<xsl:output method="html"

      encoding="UTF-8"

      indent="no"/>

 </xsl:stylesheet>
```

This is a complete stylesheet customization that you can save in a file such as docbook-utf8.xsl and use in place of the stock html/docbook.xsl stylesheet. All it does is import the stock stylesheet and set a new output encoding, in this instance to UTF-8. Any HTML files generated with this stylesheet will have their characters encoded as UTF-8, and the file will include a meta tag like the following:

```
<meta content="text/html; charset=UTF-8" http-equiv="Content-Type">
```

Changing the output encoding of the chunking stylesheet is much easier. It can be done with the *chunker.output.encoding* parameter, either on the command line or in a customization layer. That's because the chunking stylesheet uses *EXSLT* extensions to generate HTML files.

Special characters

XML is based on Unicode, which contains thousands of characters and symbols. A given XML document specifies its encoding, which is a mapping of characters to bytes. But not all encodings include all Unicode characters. Also, your keyboard may not enable you to directly enter all characters in the encoding. Any characters you cannot enter directly are entered as entities, which consists of an ampersand, followed by a name, followed by a semicolon.

There are two kinds of character entities:

Numerical character references

An entity name that consists of a # followed by the Unicode number for the character, such as á. The number can be expressed as a decimal number

such as &#431, or a hexadecimal number (which is indicated using x as a prefix), such as ".

Named character entities

A readable name such as ™ can be assigned to represent any Unicode character.

The following table shows examples of some characters expressed in both kinds of entities.

Table 2: Examples of character references

Character	Decimal character reference	Hexadecimal character reference	Named entity
á	&#225;	&#x00E1;	&aacute;
ß	&#223;	&#x00DF;	&szlig;
©	&#169;	&#x00A9;	&copy;
¥	&#165;	&#x00A5	&yen;
±	&#177;	&#x00B1;	&plusmn;
✓	&#8658;	&#x2713;	&check;

Note

Leading zeros can be omitted in numerical references. So á is the same as á.

The set of "numerical entities" (character references) is defined as part of the Unicode standard adopted by XML, but the names used for named entities are not standardized. There are several standard named character entity sets defined by ISO, however, and these are incorporated into the DocBook DTD. Among the collection of files in the DocBook XML DTD distribution, there is an ent subdirectory that contains a set of files that declare named entities. Each declaration looks like the following:

<!ENTITY plusmn "±"> <!-- PLUS-MINUS SIGN -->

This declaration assigns the numerical character reference ± to the plusmn entity name. Either ± or ± can be used in your DocBook document to represent the plus-minus sign.

Note

If you use DocBook named character entities in your document, you must also make sure your document's DOCTYPE properly identifies the DocBook DTD, and the processor must be able to find and load the DTD. If not, then the entities will be considered unresolved and the processing will fail.

Special characters in output

When an XSLT processor reads an entity from the input stream, it tries to resolve the entity. If it is a numerical character reference, it converts it to a Unicode character in memory. If it is a named entity, it checks the list of entity names that were loaded from the DTD. For the DocBook named character entities, these resolve to numerical character references. Then these numerical character references are also converted to Unicode characters in memory. All the characters in the document are handled as Unicode characters in memory during processing.

Space characters

The Unicode standard includes several characters that represent spaces that serve different purposes. Below is a table that summarizes most of them.

Table 3: Space characters

Unicode name	Character reference	DocBook entity	Description
SPACE			Ordinary space.
NO-BREAK SPACE			Space that may not be broken at the end of a line.
NARROW NO-BREAK SPACE			Thinner than NO-BREAK SPACE.
EN QUAD			Same as EN SPACE.
EM QUAD			Same as EM SPACE
EN SPACE			Half an EM SPACE.
EM SPACE			Usually a space equal to the type size in points.
THREE-PER-EM			One-third of an EM SPACE,

Unicode name	Character reference	DocBook entity	Description
SPACE			called a thick space.
FOUR-PER-EM-SPACE			One-fourth of an EM SPACE, called a mid space.
SIX-PER-EM-SPACE			One-sixth of an EM SPACE, similar to thin space.
FIGURE SPACE			Width of a digit in some fonts.
PUNCTUATION SPACE			Space equal to the narrow punctuation of a font.
THIN SPACE			One-fifth of an EM SPACE, usually.
HAIR SPACE			Thinner than a thin space.
ZERO WIDTH SPACE			No visible space. Used to allow a line break in a word without generating a hyphen.
ZERO WIDTH NO-BREAK SPACE			No visible space. Used to prevent a line break, as for example, after the hyphen in a word that contains its own hyphen.

Not all of these characters may work in the various output forms. For example, the characters in the Unicode range   to   represent spaces of different widths. However, many print fonts do not support all of the characters in that range. So the print stylesheet uses a set of templates in the stylesheet module fo/spaces.xsl to convert them to fo:leader elements of different lengths. The length unit used for the leaders is em, which scales to the current font size. If you need to customize the width of any of those special space characters, you can change the stylesheet parameters that are defined in that module.

In HTML output, these characters will pass through the stylesheet and be rendered in the output encoding for the HTML file. However, some browsers do not support all of these space characters, and may display a placeholder character instead of the intended space.

Check Your Progress 3

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) Name the problems occurred due to input validation avoidance.

.....

.....

.....

.....

2) How to prevent SQL injection attacks?

.....

.....

.....

.....

3) Define character encoding.

.....

.....

.....

.....

4) Differentiate UTF-7 and UTF-8

.....

.....

.....

.....

2.8 LET US SUM UP

This unit is an effort towards answering some of the fundamental queries about input validation, its necessity and effects of ignoring input validation. As you seen in this unit we have learnt about various input validation approaches like No input validation, Blacklisting, whitelisting and sanitizing etc. These input validation approaches are very useful to validate the inputs supplied by the outside world. You should keep in mind the guidelines givens before validating the inputs. Mostly, Integrated development environments which you are using are very useful in validating the inputs as you seen the case with STRUTS, ASP.NET and Environmental variable. Do remember not to avoid Input

validation otherwise you have to face problems like buffer overflows, SQL injections etc. Finally, we have discussed about the various encoding techniques like ASCII and Unicode etc.

2.9 CHECK YOUR PROGRESS: THE KEY

Check Your Progress 1

1) The term "blacklisting" means to filter evil input only. Unfortunately "evil input" can not be exactly defined. Usually blacklisting is based on the detection of possibly or probably or certain malicious input. The most common example is filtering of the JavaScript tag <script> to avoid Cross Site Scripting. Sometimes Blacklisting is an appropriate approach. It can be used to fill the gap between the release of a so called "Zero Day Exploit" and the release of a well tested patch. Often such an exploit is identified by a characteristic attack pattern which can filter. This can not be done using No input validation.

2) (i)All input may only pass if it is validated as known good input.

   (ii)This is also called positive input validation in contrary to the negative validation of the blacklisting.

3) (i)Validation in front of the application and before the input data will arrive the web server

   (ii) Inside the web server but before the "coded" part of the application

4) In real life the situation is often more complex. Usually a real life application comprises several components, some of them are self developed products, and other components are black box components by miscellaneous vendors. In such cases one can use a mix of the approaches above. Here we recommend to ask an experienced web application security consultant.

Check Your Progress 2

1) · GET parameters

   · Forms fields (hidden fields)

   · Selection lists and drop down lists

   · File Upload (file name and file content)

   · Cookies

   · HTTP-headers

   · Java applet communication

· Flat file import

2) public static boolean validateRequired(Object bean, ValidatorAction va, Field field, ActionErrors errors, HttpServletRequest request) {String value = null; if (isString(bean)) {value = (String) bean; } else {value = ValidatorUtil.getValueAsString(bean, field.getProperty());} if (GenericValidator.isBlankOrNull(value)) {errors.add(field.getKey(), StrutsValidatorUtil.getActionError(request, va, field)); return false; } else {return true;}

3) The attacker can change PATH to the following, and cause "mail" in the current directory to be executed.

PATH=".:$PATH"; export PATH

4) (i) Test for false positives: Your business must not generate validation errors.

(ii) Test for false negatives: Test with invalid data to check the correctness of the validation routines.

(iii)Test with and without JavaScript: As described above JavaScript interprets regular expression slightly different compared with server engines. These differences can generate false positives at the client side.

(iv) Perform a test **without** any validation at the client and the server. This testan expose insecurities in other tiers of the application. Some developers rely on the correct validation of the application frontend which is not sufficient.

Check Your Progress 3

1) (i) Buffer Overflows

(ii) Cross-site scripting

(iii) SQL Injection

(iv) Canonicalization

2) (i) Sanitize the input

(ii) Configure error reporting: the above attacks take advantage of the error messages returned by the sever. It can make attacker's life more difficult by not telling the users the actual error message in the SQL query.

(iii) Use bound parameters

3) A method (algorithm) for presenting characters in digital form by mapping sequences of code numbers of characters into sequences of octets. In the simplest case, each character is mapped to an integer in the range 0 - 255 according to a character code and these are used as such as octets. Naturally, this only works for character repertoires with at most 256 characters. For larger sets, more complicated encodings are needed.

4) In UTF-8, Character codes less than 128 (effectively, the ASCII repertoire) are presented "as such", using one octet for each code (character) All other codes are presented, according to a relatively complicated method, so that one code (character) is presented as a sequence of two to four octets, each of which is in the range 128 - 255. In UTF-7, Each character code is presented as a sequence of one or more octets in the range 0 - 127 ("bytes with most significant bit set to 0", or "seven-bit bytes", hence the name).

2.10 SUGGESTED READINGS

- A Guide to Buildung Secure Web Applications and Web Services, The Open Web Application Security Project, 2006, http://www.owasp.org/index.php/OWASP_Guide_Project
- Edberg, Peter. Character Encodings Concepts.
- Character set standards in the Standards and Specifications List by Diffuse
- Guide to Character Sets, by Diffuse
- Session Management, secologic Whitepaper, 2007, http://www.secologic.de
- Sicherheit von Webanwendungen, Maßnahmen und Best Practices, Bundesamt für Sicherheit in der Informationstechnik, 2006, http://www.bsi.de
- STRUTS Validator Guide, 2000-2004, The Apache Software Foundation, http://struts.apache.org/1.2.4/userGuide/dev_validator.html
- The Ten Most Critical Web Application Security Vulnerabilities, The Open Web Application Security Project, 2004, http://www.owasp.org/index.php/OWASP_Top_Ten_Project

UNIT 3 AUTHENTICATION, AUTHORIZATION AND SESSION MANAGEMENT

Structure

3.0 Introduction
3.1 Objectives
3.2 Secure Application Development Principles
  3.2.1 Brief Background
  3.2.2 Principles
  3.2.3 Difference between Authentication and Authorization
3.3 Authentication
  3.3.1 Three Parties to Authentication
  3.3.2 Basic Access Authentication
  3.3.3 Digest Access Authentication
  3.3.4 Form Based Authentication
  3.3.5 Password Based Authentication
  3.3.6 TLS/SSL
3.4 Authorization
  3.4.1 Types of Authorization
  3.4.2 Attribute Based Access Control
  3.4.3 Discretionary Access Control
  3.4.4 Mandatory Access Control
  3.4.5 Role Based Access Control
  3.4.6 File System Access Control
3.5 Session Management
  3.5.1 Basic Terms
  3.5.2 Introduction to Session Management
  3.5.3 Session Management Types
  3.5.4 Session Management Scheme
3.6 Let Us Sum Up
3.7 Check Your Progress: The Key
3.8 Suggested Readings

3.0 INTRODUCTION

Since the early days of writing, authorities understood it was necessary to provide some mechanism to protect the confidentiality of written correspondence and to have some means of detecting tampering.

Julius Caesar is credited with the invention of the Caesar cipher (50 B.C.) for secret messaging and World War II brought about much advancement in information security.

The end of the 20th century and early years of the 21st century saw rapid advancements in Information Communication Technology. The availability of smaller, more powerful and less expensive computing equipment made electronic data processing within the reach of small business and the home user. These computers quickly became interconnected through Internet or World Wide Web.

The rapid growth and widespread use of electronic data processing and electronic business conducted through the Internet, along with numerous occurrences of international terrorism, fueled the need for better methods of protecting the computers and the information they store, process and transmit. The academic disciplines of information security emerged along with numerous professional organizations – all sharing the common goals of ensuring the security and reliability of information systems.

3.1 OBJECTIVES

After studying this unit, you should be able to:

- identify the principle of secure application development;
- explain different advantages and types of authentication;
- recognize why authorization is needed ;
- elucidate the purpose of protocols used information security;
- differentiate between authorization and access control;
- explain the need of session management; and
- find different methods that can be used session management.

3.2 SECURE APPLICATION DEVELOPMENT PRINCIPLES

Information security is not just about firewalls and tunnels. Security must also be built into applications as well. To encourage application teams to integrate security into the development life cycle some principle areas of secure application development: authentication, authorization and session management need to be taken care of. There comes a point in most projects when you need to be able to restrict access to certain parts of your web site

information. Mainly it is due to the requirement of providing privacy to each user. To do this, you will need to implement a security system that can confirm the identity of a user and then restrict the information each user has access to, based on their permissions to each area. Therefore we need to implement the two processes known as *authentication* and *authorization*.

3.2.1 Brief Background

Authentication is typically performed by asking a visitor for their username and password. If the visitor enters the correct password, you can be fairly sure they are who they claim to be.

Just because the user has been authenticated, it doesn't necessarily mean they should be authorized to access the information they are trying to visit. If, for example, you signed into a Google account with your username and password, you wouldn't expect to be able to read the e-mails of other users, because you wouldn't be authorized to do so.

Authorization is usually performed by checking the authenticated user has the appropriate permissions to access the page they are trying to visit. The permission check can be as simple as ensuring that a user has in fact signed in but can also be very sophisticated involving checks about the user's roles, group, or even which computer the user is accessing the web site from.

3.2.2 Principles

Information security revolves around protecting the confidential data from malicious access for which the three processes are considered mandatory for implementation. The principle of secure application development states to integrate security into the development life cycle of a product which can be implemented by the security process. These processes are *authentication, authorization* and *session management.* Authentication is mainly responsible for tying the other two processes seamlessly. Technically these processes are defined as follows:

-**Authentication** is a process of establishing confidence in the truth of some claim.

-**Authorization** is a process of deciding what an individual is ought to be allowed to do

-**Session Management** is a process of keeping track of user's activity of interaction with a computer system or website.

3.2.3 Difference between Authentication and Authorization

1. Authentication establishes identity.
2. Authorization decides what privileges a given person or program has.

3. Authentication tells who the user are
4. Authorization tells what the user can do.
5. Authentication is to verify the identity of the claim
6. Authorization deals with the mechanism to provide access control.

3.3 AUTHENTICATION

Authentication is a process of establishing confidence in the truth of some claim. In the context of information security the term "authentication" is meant to be "verification of a claimed identity". Authentication is usually implemented to satisfy the security requirements needed by a user. Generally, authentication is conducted by verifying a visitor for their username and password. If the visitor enters the correct password, you can be fairly sure they are who they claim to be.

There are three different types of authenticators, which can be used either alone or in combination (called multi-factor authentication):

1. Something you know - passphrases, passwords, PIN codes
2. Something you have - magnetic stripe or smart cards, tokens, physical keys
3. Something you are - biometrics: fingerprint, retinal scan, voice recognition, hand shape

3.3.1 Three Parties to Authentication

Authentication systems involve parties who play three key roles in the establishment of truth of claim.

1. Presenter: Who presents the credentials of the veracity.
2. Issuer: A third part who issues the credentials.
3. Verifier: Who verifies the credentials for their veracity.

In some cases one party may play two roles likes we may encounter a scenario where the issuer and verifier are the same. The issuer however on the other hand may perform separate checks to ensure that credentials are issued only to legitimate parties.

When analyzing different authentication systems, it is important to identify how these roles and responsibilities are performed. Further authentication can be classified into two broader categories depending upon the way this process is being used. It can be *user authentication* which is a process to determine that a user is who he/she claims to be and *entity authentication* a process to

determine an entity who it claims to be. Generally entity authentication is used to in case of session management.

Discussing about authentication for web application it is mainly required for all non-public parts of an organization's website. In some cases it may only be used to make a simple yes/no decision: for example, many subscription services really only care that you're on the list of subscribers. In other cases, there may be complex application and/or business logic associated with the access rules and user roles within the application. Authentication is not only important for authorization purposes, but also for logging and non-repudiation in the future.

3.3.2 Basic Access Authentication

In the context of an HTTP transaction, **basic access authentication** is a method for a web browser or other client program to provide a user name and password when making a request. Before transmission, the user name and password are concatenated with a colon separation. The resulting string is encoded with the Base 64 algorithm. For example, given the user name 'Aladdin' and password 'open sesame', the string 'Aladdin:open sesame' is Base64 encoded, resulting in 'QWxhZGRpbjpvcGVuIHNlc2FtZQ=='. The Base64-encoded string is transmitted and decoded by the receiver, resulting in the colon-separated user name and password string. Here, security is *not* the implemented in the encoding step. Rather, the intent of the encoding is to encode non-HTTP-compatible characters of the user name or password into HTTP-compatible characters for authentication.

Advantages

1. All web browsers support it.
2. It may be used in private system as its rarely used in publicly accessible internet websites.

Disadvantages

1. It assumes the connection between client and servers to be trusted and if SSL/TSL is not used, the credentials are passed as plain text and may be intercepted.
2. Existing browsers retain authentication information until the tab or browser is closed or the user clears the history.

Example with explanation

This example shows a typical transaction between HTTP client and HTTP server running on the localhost. It comprises the following steps.

3. The server may inspect nonce attributes submitted by clients, to prevent replay attacks.
4. Server is also allowed to maintain a list of recently issued or used values to prevent reuse.

Disadvantage

1. It is not intended to replace strong authentication protocols, such as public-key or Kerberos authentication.
2. Digest access authentication is vulnerable to a man-in-the-middle (MitM) attack.
3. Some servers require passwords to be stored using reversible encryption. However, it is possible to instead store the digested value of the username, realm, and password.

Example with explanation

This typical transaction consists of the following steps.

1. The client asks for a page that requires authentication but does not provide a user name and password. This is because the user simply entered the address or followed a link to that page.
2. The server responds with the 401 response code and the client needs to authenticate the claim by providing a randomly-generated, single-use value called a nonce.
3. The client will present the user with prompt for a user name and password. The user may decide to cancel at this point.
4. Once a user name and password have been supplied, the client adds an authentication header to the original request and re-sends it.
5. In this example, the server accepts the authentication and the page is returned. If the user name is invalid or the password incorrect, the server might return the 401 response code and the client would prompt the user again.

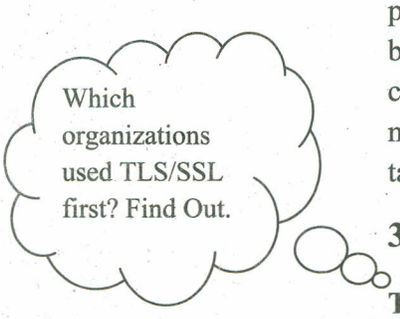
3.3.4 Form Based Authentication

Form based authentication is a mechanism in which HTML forms are used to acquire the credentials from the user as a part of the web application itself. Here the user is presented with an editable form and is asked to claim the authentication by entering the credentials. When the user is returning again to the same application the website may provide with a prefilled form having blank password field. It may also ask to confirm the password followed by changing the password every time.

This is a step further to the protocol based authentication. Generally it is implemented using HTTP + HTML or XHTML.

3.3.5 Password Based Authentication

Usernames and passwords are the most common form of authentication in use today. Despite the improved mechanisms over which authentication information can be carried (like HTTP Digest and client side certificates), most systems usually require a password as the token against which initial authorization is performed. Due to the conflicting goals that good password maintenance schemes must meet, passwords are often the weakest link in authentication architecture. More often than not, this is due to human and policy factors and can be only partially addressed by technical remedies. Some best practices are outlined here, as well as risks and benefits for each countermeasure. As always, those implementing authentication systems should measure risks and benefits against an appropriate threat model and protection target.



3.3.6 TLS/SSL

Transport Layer Security (TLS) and its predecessor, **Secure Sockets Layer (SSL)**, are cryptographic protocols that provide communication security over the Internet. TLS and SSL encrypt the segments of network connections above the Transport Layer, using asymmetric cryptography for privacy and a keyed message authentication code for message reliability.

Several versions of the protocols are in widespread use in applications such as web browsing, electronic mail, Internet faxing, instant messaging and voice-over-IP (VoIP).

TLS and SSL are most widely recognized as the protocols that provide secure HTTP (HTTPS) for Internet transactions between Web browsers and Web servers. TLS/SSL can also be used for other application level protocols, such as File Transfer Protocol (FTP), Lightweight Directory Access Protocol (LDAP), and Simple Mail Transfer Protocol (SMTP). TLS/SSL enables server authentication, client authentication, data encryption, and data integrity over networks such as the World Wide Web.

The TLS protocol allows client/server applications to communicate across a network in a way designed to prevent eavesdropping and tampering.

A TLS client and server negotiate a stateful connection by using a handshaking procedure. During this handshake, the client and server agree on various parameters used to establish the connection's security.

- The handshake begins when a client connects to a TLS-enabled server requesting a secure connection and presents a list of supported CipherSuites (ciphers and hash functions).

- From this list, the server picks the strongest cipher and hash function that it also supports and notifies the client of the decision.
- The server sends back its identification in the form of a digital certificate. The certificate usually contains the server name, the trusted certificate authority (CA) and the server's public encryption key.
- The client may contact the server that issued the certificate (the trusted CA as above) and confirm the validity of the certificate before proceeding.
- In order to generate the session keys used for the secure connection, the client encrypts a random number with the server's public key and sends the result to the server. Only the server should be able to decrypt it, with its private key.
- From the random number, both parties generate key material for encryption and decryption.

This concludes the handshake and begins the secured connection, which is encrypted and decrypted with the key material until the connection closes.

If any one of the above steps fails, the TLS handshake fails and the connection is not created.

SSL has two major modes of operation. The first is where the SSL tunnel is set up and only the server is authenticated, the second is where both the server and client are authenticated. In both cases the SSL session is setup before the HTTP transaction takes place.

SSL negotiation with server authentication only is a nine-step process.

1. The first step in the process is for the client to send the server a Client Hello message. This hello message contains the SSL version and the cipher suites the client can talk. The client sends its maximum key length details at this time.
2. The server returns the hello message with one of its own in which it nominates the version of SSL and the ciphers and key lengths to be used in the conversation, chosen from the choice offered in the client hello.
3. The server sends its digital certificate to the client for inspection. Most modern browsers automatically check the certificate (depending on configuration) and warn the user if it's not valid. By valid we mean if it does not point to a certification authority that is explicitly trusted or is out of date, etc.
4. The server sends a server done message noting it has concluded the initial part of the setup sequence.

5. The client generates a symmetric key and encrypts it using the server's public key (cert). It then sends this message to the server.
6. The client sends a cipher spec message telling the server all future communication should be with the new key.
7. The client now sends a Finished message using the new key to determine if the server is able to decrypt the message and the negotiation was successful.
8. The server sends a Change Cipher Spec message telling the client that all future communications will be encrypted.
9. The server sends its own Finished message encrypted using the key. If the client can read this message then the negotiation is successfully completed.

SSL negotiation with mutual authentication (client and server) is a twelve-step process.

1. The first step in the process is for the client to send the server a Client Hello message. This hello message contains the SSL version and the cipher suites the client can talk. The client sends its maximum key length details at this time.
2. The server returns the hello message with one of its own in which it nominates the version of SSL and the ciphers and key lengths to be used in the conversation, chosen from the choice offered in the client hello.
3. The server sends its digital certificate to the client for inspection. Most modern browsers automatically check the certificate (depending on configuration) and warn the user if it's not valid. By valid we mean if it does not point to a certification authority that is explicitly trusted or is out of date, etc.
4. The server sends a Certificate request after sending its own certificate.
5. The server sends a server done message noting it has concluded the initial part of the setup sequence.
6. The client provides its Certificate.
7. The client generates a symmetric key and encrypts it using the server's public key (cert). It then sends this message to the server.
8. The client sends a Certificate verify message in which it encrypts a known piece of plaintext using its private key. The server uses the client certificate to decrypt, therefore ascertaining the client has the private key.
9. The client sends a cipher spec message telling the server all future communication should be with the new key.

10. The client now sends a finished message using the new key to determine if the server is able to decrypt the message and the negotiation was successful.
11. The server sends a Change Cipher Spec message telling the client that all future communications will be encrypted.
12. The server sends its own Finished message encrypted using the key. If the client can read this message then the negotiation is successfully completed.

i. E-Mail Authentication

It is an authentication mechanism in which the receiver of the e-mail needs to authenticate the sender using the facility provided by an Internet service provider (ISP). There are a number of methods to authenticate the sender i.e. to check for spoofing and spamming, they are:

Sender Policy Framework (SPF)

It is an email validation system which is used to avoid email spam by detecting email spoofing, a common vulnerability. Here sender's IP addresses are verified. SPF maintains a specific SPF record (or TXT record) in the Domain Name System (DNS) which is used to administer which hosts are acceptable to send mail from a given domain. It uses the protocol stack defined by the TCP/IP protocol suite for networking.

DomainKeys Identified Mail (DKIM)

It is another method for associating a domain name to an email. The association is provided by using digital signature which can be validated by receiver. Responsibility is claimed by an independent party by adding a DKIM-Signature: field to the message's header. A DKIM signature can also cover fields, such as the From: , Subject: , and the message body. Therefore, the strength of a DKIM-Signature can be set as per the message modifications that are required. However, DKIM doest not support integrity of the message it is only used for authentication purpose.

ii. Chat Authentication

This is a mechanism which deals authentication for proceeding with chat services. Depending upon the chatting tool used we have different approach used for it. The most readily available chatting services are provided by Facebook, Google Talk and yahoo messenger wherein the user needs to create its own account and authentication is done using username and password.

iii. Two-Factor Authentication

A two factor authentication is a process of establishing confidence in the truth of some claim using two means for the identification of the user. Here two

independent means are used to authenticate the user. Some of the schemes falling under this category are

iv. SMS Authentication

The basic principle behind this scheme is that once the first level of authentication is done, after that the user is sent a verification code on his/her registered mobile phone. This code is generally a one-time password, which is valid to be used only single time. This scheme provides an additional layer of security, as the user apart from the username and password is asked to provide a verification code also known as PIN code at the time of authentication. Similar to the SMS authentication, e-mail authentication can also be used as an authentication mechanism wherein the verification code i.e. one-time password is sent on the registered e-mail address of the user. The user needs to login and find that out to continue with the authentication process.

v. Certificate Authentication

The use of certificate authentication is mainly to support e-commerce activities. A certificate is a digital document that at a minimum includes a Distinguished Name (DN) and an associated public key. The certificate is digitally signed by a trusted third party known as the Certificate Authority (CA). The CA vouches for the authenticity of the certificate holder. For every transaction to be performed electronically a certificate is to be presented by the principal as its credentials. The recipient then validates the certificate's signature against its database for known and trusted CA certificates. A "personal certificate" identifies an end user in a transaction; a "server certificate" identifies the service provider.

The certificate system seems simple enough. However, there is a great deal of infrastructure that makes certificate based authentication possible. Besides the basic standard structure, certificates must be unique and unalterable. The signature on which the validity of a certificate is made must be backed by policies that cannot be repudiated.

vi. Third Party Token

It is a method using which third party applications can access actual user data by getting automatic authorization from the user, without needing username and password. To provide this feature token based systems provide are used. The implementation can be complicated, but the basic process looks like this:

1. The third-party application initially needs to authenticate the user.
2. The user is directed to a special URL within the web application and supposed to login in using username and pasword.

3. Our application confirms whether the user allow the third-party application to act on his behalf.

4. Our application then generates a token code and passes it back to the third-party application.

5. The third-party application then uses the token to sign API method calls.

   Each token is unique, that application can use it and only for a specific user. Once the token is generated, we pass it back to the third-party application, so we'll need some way of contacting it. This is a secretive transfer about which the user is unaware, so the connection needs to be secure. Thus, it is used in collaboration with HTTPS.

vii. Authentication Platform

FireID

It is one of the most comprehensive authentication solutions for the banking industry. The FireID Authentication Platform for Banks provides transaction verification, authentication for mobile and online banking, and internal VPN access for bank use. Using something the end-user already has, a mobile phone, FireID protects financial transactions against sophisticated threats "including man-in-the-browser attacks" and ensures that only legitimate payments are made.

DualShield

Deepnet DualShield is an award-winning, versatile, unified authentication platform that delivers multi-factor authentication across diverse channels, applications, users and security tokens. DualShield enables organizations to deploy a wide variety of authentication methods and to protect a wide range of remote access channels and applications with strong, multi-factor authentication in a single, unified platform .For cloud or web based applications, such as Google Apps, Salesforce and any SAML enabled application, DualShield Single Sign-On server acts as an identity provider that authenticates users and provides information used to authorize users. When a user attempts to login to a DualShield protected cloud or web application, the request is automatically redirected to DualShield SSO. DualShield SSO parses the request, authenticates the user to an organization's AD/LDAP directory, and generates a SAML response to the cloud or web application. Once successfully verified, the user is automatically logged in to the application.

Check Your Progress 1

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) How many parties are broadly involved in authentication? Explain the role of each party.

.....

.....

.....

.....

2) What are the various protocols used for security over internet?.

.....

.....

.....

.....

3) What is the difference between user authentication and entity authentication?

.....

.....

.....

.....

4) Compare Basic access authentication with digest access authentication?

.....

.....

.....

.....

3.4 AUTHORIZATION

Authorization give you the ability to restrict access to particular destinations within your application (a module, a screen, a function, a field/object) based on the user's profile

- Authorization is a process in which the authenticated user is allowed access only to the resources he has been authorized to interact with.
- Allows the user to access particular systems and functions within those systems.

- The granularity of the user's access might extend to allow control to particular objects within a system.
- Multi-tier authorization allows trusted users to control authorization to a subset of other users, gives the system greater flexibility, while greatly reducing the operational cost of the application.
- Currently most authorization mechanisms are single tier, allowing only controlling the parts of a system the user can gain access to and placing a significant burden on maintenance, support, and administration of the system as a single type of user is overwhelmed with that function.

Access control sounds very similar, but there is a subtle difference: access control refers to the permissions assigned to an authenticated person or entity. In other words, access control first requires successful authentication, and then it can use the identity of the logged-in user to determine to which resources he has access. However, it is also possible to enforce access control without any authentication taking place, which in most cases will revert to limited/guest access. An example of this is an e-commerce application that allows users to shop without logging in.

Incorrectly implemented authorization can have some serious security implications. Even a highly secure three-factor authentication (passcode, token, and biometrics) is useless without proper authorization: a user could potentially access other users' data or administrative functions. Some common examples of broken access control are:

1. Insecure ID's (guessable/sequential order numbers, etc.)
2. Forced browsing past access control checks (only the first page is checked)
3. Path traversal, incorrect file permissions
4. Client-side caching (pages may still reload after session expires)

3.4.1 Types of Authorization

Authorization to access information and other computing services (run, view, create, delete, change) begins with administrative policies and procedures. The policies prescribe what information and computing services can be accessed, by whom, and under what conditions. The access control mechanisms are then configured to enforce these policies.

Access control mechanisms are now days considered to be crucial design element in secure application development. Like in a web application one should protect front-end and back-end information and system resources by implementing access control restrictions on what users can do, which resources they have access to, and what functions they are allowed to perform on the

piece of information. Ideally, an access control scheme should protect against the computing services. Additionally, access control mechanisms can also help limit malicious code execution, or unauthorized actions through outer world. Access Control refers to way of controlling access to web resources, including certain restrictions based on the time of day, the IP address of the HTTP client browser, the domain of the HTTP client browser, the type of encryption the HTTP client can support, number of times the user has authenticated that day, the possession of any number of types of hardware/software tokens, etc.

Before choosing the access control mechanisms we need to design the process as per our requirements. Following are some guidelines which support this.

1. Try to quantify the relative value of information to be secured in terms of Confidentiality, Sensitivity, Classification, Privacy, and Integrity related to the organization as well as the individual users.
2. Consider the worst case financial loss that unauthorized disclosure, modification, or denial of service or any other attack could cause.
3. Determine the relative interaction that information owners and creators will have within the web application
4. Specify the process for granting and revoking user access control rights on the system, for all manual and automated access.
5. Try to determine which specific user functions, administrative functions and maintenance functions are to be built in the application.
6. Try to align your access control mechanisms as closely as possible to your organization's security policy.

Most modern operating systems also define sets of permissions that are variations or extensions of three basic types of access:

- Read (R): The subject can
    - Read file contents
    - List directory contents
- Write (W): The subject can change the contents of a file or directory with the following tasks:
    - Add
    - Create
    - Delete
    - Rename

- Execute (X): If the file is a program, the subject can cause the program to be run. (In UNIX systems, the 'execute' permission doubles as a 'traverse directory' permission when granted for a directory.)

These rights and permissions are implemented differently in systems. The access control mechanism a system offers will be based upon one of four approaches to access control or it may be derived from a combination of the four approaches.

1. Attribute Based Access Control
2. Discretionary Access Control
3. Mandatory Access Control
4. Role Based Access Control

3.4.2 Attribute Based Access Control

In attribute-based access control (ABAC), access is granted not based on the rights of the subject associated with a user after authentication, but based on attributes of the user. The user has to prove so called claims about his attributes to the access control engine. An attribute-based access control policy specifies which claims need to be satisfied in order to grant access to an object. For instance the claim could be "younger than 45". Any user that can prove this claim is granted access. Users can be anonymous as authentication and identification are not strictly required. One does however require means for proving claims anonymously. This can for instance be achieved using anonymous credentials or XACML (extensible access control markup language).

3.4.3 Discretionary Access Control

Discretionary access control (DAC) is an access policy determined by the owner of an object. The owner decides who is allowed to access the object and what privileges they have based on the identity of the user and/or membership in a certain group.

Two important concepts in DAC are

- File and data ownership: Every object in the system has an *owner*. In most DAC systems, each object's initial owner is the subject that caused it to be created. The access policy for an object is determined by its owner.
- Access rights and permissions: These are the controls that an owner can assign to other subjects for specific resources.

3.4.4 Mandatory Access Control

Mandatory access control (MAC) is an access policy determined by the system, not by the owner. MAC is used in multilevel systems that process highly sensitive data belonging to an organization. A multilevel system is a single computer system that handles multiple classification levels between subjects and objects.

- Sensitivity labels: In a MAC-based system, all subjects and objects must have labels assigned to them. A subject's sensitivity label specifies its level of trust. An object's sensitivity label specifies the level of trust required for access. In order to access a given object, the subject must have a sensitivity level equal to or higher than the requested object.
- Data import and export: Controlling the import of information from other systems and export to other systems (including printers) is a critical function of MAC-based systems, which must ensure that sensitivity labels are properly maintained and implemented so that sensitive information is appropriately protected at all times.

Two methods are commonly used for applying mandatory access control:

- Rule-based (or label-based) access control: This type of control further defines specific conditions for access to a requested object. All MAC-based systems implement a simple form of rule-based access control to determine whether access should be granted or denied by matching:
    - An object's sensitivity label
    - A subject's sensitivity label
- Lattice-based access control: These can be used for complex access control decisions involving multiple objects and/or subjects. A lattice model is a mathematical structure that defines greatest lower-bound and least upper-bound values for a pair of elements, such as a subject and an object.

3.4.5 Role Based Access Control

Role-based access control (RBAC) is an access policy determined by the system, not by the owner. RBAC is used where multi-level security requirements may also exist. RBAC differs from DAC in that DAC allows users to control access to their resources, while in RBAC, access is controlled at the system level, outside of the user's control. Although RBAC is non-discretionary, it can be distinguished from MAC primarily in the way permissions are handled. MAC controls read and write permissions based on a user's clearance level and additional labels. RBAC controls collections of permissions that may include complex operations such as an e-commerce

transaction, or may be as simple as read or write. A role in RBAC can be viewed as a set of permissions.

Three primary rules are defined for RBAC:

1. Role assignment: A subject can execute a transaction only if the subject has selected or been assigned a role.
2. Role authorization: A subject's active role must be authorized for the subject. With rule 1 above, this rule ensures that users can take on only roles for which they are authorized.
3. Transaction authorization: A subject can execute a transaction only if the transaction is authorized for the subject's active role. With rules 1 and 2, this rule ensures that users can execute only transactions for which they are authorized.

Additional constraints may be applied as well, and roles can be combined in a hierarchy where higher-level roles subsume permissions owned by sub-roles. Most IT vendors offer RBAC in one or more products.

3.4.6 File System Access Control

A File system Access Control is a mechanism for granting permission to the user to access a specific file. Generally it uses a data structure like tables containing entries which specify individual user or group rights to specific system files. Access control entries (ACE) is the term associated with these table entries in the Microsoft Windows NT, OpenVMS, Unix-like, and Mac OS X operating systems. The privileges or permissions specify access rights, which are whether a user can read from file and write to file. In some implementations a user, or group of users, may also modify the control rights of the file. Like for protection and maintaining the secrecy of the document the creator can change the rights from read-write to read only and vice-versa

Check Your Progress 2

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) Why authorization is necessary?

.....

.....

.....

.....

2) Differentiate between authorization and access control.

.....

.....

.....

.....

3) Which type of access control mechanism is used in case of web applications?

.....

.....

.....

.....

4) What are the rules defined for Role based access control mechanisms?

.....

.....

.....

.....

3.5 SESSION MANAGEMENT

3.5.1 Basic Terms

Before discussing in detail about the third most important principle for secure application development, it would be pertinent to have an insight about some basic terms required for its understanding

Cookie

A **cookie**, also known as an **HTTP cookie**, **web cookie**, or **browser cookie**, is used for an origin website to send state information to a user's browser and for the browser to return the state information to the origin site. The state information can be used for authentication, identification of a user session, user's preferences, shopping cart contents, or anything else that can be accomplished through storing text data.

Cookies are not software. They cannot be programmed, cannot carry viruses, and cannot install malware on the host computer. However, they can be used by spyware to track user's browsing activities

There are two categories of cookies, secure or non-secure and persistent or non-persistent, with which we can have the following four combinations.

1. Persistent and Secure
2. Persistent and Non-Secure
3. Non-Persistenet and Secure
4. Non-Persistent and Non-Secure

Persistent vs. Non-Persistent

Persistent cookies are stored in a text file (cookies.txt under Netscape and multiple *.txt files for Internet Explorer) on the client and are valid for as long as the expiry date is set for. Non-Persistent cookies are stored in RAM on the client and are destroyed when the browser is closed or the cookie is explicitly killed by a log-off script.

Secure vs. Non-Secure

Secure cookies can only be sent over HTTPS (SSL). Non-Secure cookies can be sent over HTTPS or regular HTTP. The title of secure is somewhat misleading. It only provides transport security. It ensures that the cookie is always encrypted when transmitting from client to server. This makes the cookie less likely to be exposed to cookie theft via eavesdropping.

Session Tokens

A session token is a unique identifier that is generated and sent from a server to a client to identify the current interaction session. The client usually stores and sends the token as an HTTP cookie and/or sends it as a parameter in GET or POST queries. The reason to use session tokens is that the client only has to handle the identifier—all session data is stored on the server (usually in a database, to which the client does not have direct access) linked to that identifier. Examples of the names that some programming languages use when naming their HTTP cookie include JSESSIONID (JSP), PHPSESSID (PHP), and ASPSESSIONID (ASP).

Cryptographic Algorithms for Session Tokens

All session tokens should be user unique, non-predictable, and resistant to reverse engineering. A trusted source of randomness should be used to create the token (like a pseudo-random number generator, etc.). Additionally, for more security, session tokens should be tied in some way to a specific HTTP client instance to prevent hijacking and replay attacks. Examples of mechanisms for enforcing this restriction may be the use of page tokens which are unique for any generated page and may be tied to session tokens on the server. In general, a session token algorithm should never be based on or use as variables any user personal information (user name, password, home address, etc.)

3.5.2 Introduction to Session Management

We have now covered authentication and authorization. The final member of security mechanisms is session management. Application developers, especially those creating web applications, must pay special attention to this important mechanism. This includes aspects such as the security of the security tokens and session inactivity timeouts. Session hijacking through session identifier or cookie stealing are all common flaw that plagues a number of web applications. Essentially, an attacker simply obtains the session identifier value and then crafts a request that contains that ID. When that request reaches the server, the application will simply look up its session data store and determine that the user was logged in and will therefore provide access to all the resources that are available to that user. As a result, an attacker could potentially get unauthorized access to another user's data.

If an application cannot avoid creating its own session management mechanism — for instance, if the underlying application server does not support sessions — then Cookies should be the preferred mechanism for storing the session identifier. Cookieless sessions, while technically possible, are far more difficult to secure

When using cookies, developers must clearly define the cookie path, expiration date, and time, as well as ensure that the cookie is not persistent. After a maximum of 30 minutes of inactivity, the user must be forced to log in again so as to decrease the proximity of attack. Further, cookies must have the secure flag turned on this ensures that the browser will never transmit this cookie in a clear-text request. Another common mistake that developers often make is failing to issue a fresh session ID when the application switches from an insecure mode (used typically before a user logs in to the application) to a secure mode (after login). Thus, an attacker who has obtained a cookie by sniffing traffic can wait for the legitimate user to log in and then use that cookie to impersonate him or her.

Session IDs, when generated, must have sufficient entropy to prevent an attacker from guessing a valid session ID value and thus hijacking that specific session.

When the user is logged out, the session must be invalidated on the server side as well as on the client side. This is typically done by deleting the session entry from the server session data store, as well as by clearing the cookie in the client application or browser and in the history.

3.5.3 Session Management Types

Desktop session management

Desktop session management is done using a program which save and restore desktop sessions. A desktop session is all windows which are currently running. Desktop session also stores the current content of the windows which are running. For Linux-based systems session management is provided by X session manager. For Microsoft Windows systems, no implicit session manager is included in the system. Third-party applications like twinsplay are used for session management.

Browser session management

Session management in a web browser is used to provide a user with the facility to save all open pages along with the settings and restore them again when required. By default if the application or the system crashes then also the settings can be restored automatically on the next run. OmniWeb, Google Chrome and Opera are examples of web browsers that support session management. While Mozilla Firefox supports session management through third-party plugins or extensions. It provides the features of add-ons which are used to support it.

Web server session management

For a web server session management a Transmission Control Protocol (TCP) network connection must be established with each new Hypertext Transfer Protocol (HTTP) GET or POST request. Thus, the web server cannot rely on an established TCP network connection for longer than a single HTTP GET or POST operation as for a stateless HTTP protocol this provide a session state. This will provide a user to authenticate oneself only once.

For every session a session identifier (session ID) is generated to store the session management details by the web server. These sessions ID's are used to store and map the data for a specific session using the storage mechanism implemented like flat files or database.

For situations where we need to share knowledge of session state between multiple server, cluster nodes that are running web server software are used. Methods for sharing session state between nodes in a cluster include:

- Multicasting session information to member nodes,
- Sharing session information with a partner node using distributed shared memory or memory virtualization,
- Sharing session information between nodes using network sockets,

- Storing session information on a shared file system such as the network file system or the global file system,
- Storing the session information outside the cluster in a database.

If session information is considered transient, volatile data that is not required for non-repudiation of transactions and doesn't contain data that is subject to compliance auditing, then any method of storing session information can be used. However, if session information is subject to audit compliance, consideration should be given to the method used for session storage, replication, and clustering.

3.5.4 Session Management Scheme

Under the hood, all web applications use of HTTP for client-server communication, which is a stateless protocol by design. In other words, a standard web server has no concept of session history; it sees all page requests as completely separate, even if they come from the same browser or IP address. The majority of web applications require some form of persistent "memory" of what the user has already done; otherwise the user experience would be horrible: either all data would have to be entered on a single page, or the user would have to re-enter his login and password for every page.

Session Time-out

Session tokens that do not expire on the HTTP server can allow an attacker unlimited time to guess or brute force a valid authenticated session token. An example is the "Remember Me" option on many retail websites. If a user's cookie file is captured or brute-forced, then an attacker can use these static-session tokens to gain access to that user's web accounts. Additionally, session tokens can be potentially logged and cached in proxy servers that, if broken into by an attacker, may contain similar sorts of information in logs that can be exploited if the particular session has not been expired on the HTTP server.

Regeneration of Session Tokens

To prevent Session Hijacking and Brute Force attacks from occurring to an active session, the HTTP server can seamlessly expire and regenerate tokens to give an attacker a smaller window of time for replay exploitation of each legitimate token. Token expiration can be performed based on number of requests or time.

Session Forging/Brute-Forcing Detection and/or Lockout

Many websites have prohibitions against unrestrained password guessing (e.g., it can temporarily lock the account or stop listening to the IP address). With regard to session token brute-force attacks, an attacker can probably try hundreds or thousands of session tokens embedded in a legitimate URL or

cookie for example without a single complaint from the HTTP server. Many intrusion-detection systems do not actively look for this type of attack; penetration tests also often overlook this weakness in web e-commerce systems. Designers can use "booby trapped" session tokens that never actually get assigned but will detect if an attacker is trying to brute force a range of tokens. Resulting actions can either ban originating IP address (all behind proxy will be affected) or lock out the account (potential DoS). Anomaly/misuse detection hooks can also be built in to detect if an authenticated user tries to manipulate their token to gain elevated privileges.

Session Re-Authentication

Critical user actions such as money transfer or significant purchase decisions should require the user to re-authenticate or be reissued another session token immediately prior to significant actions. Developers can also somewhat segment data and user actions to the extent where re-authentication is required upon crossing certain "boundaries" to prevent some types of cross-site scripting attacks that exploit user accounts.

Session Token Transmission

If a session token is captured in transit through network interception, a web application account is then trivially prone to a replay or hijacking attack. Typical web encryption technologies include but are not limited to Secure Sockets Layer (SSLv2/v3) and Transport Layer Security (TLS v1) protocols in order to safeguard the state mechanism token.

Session Tokens on Logout

With the popularity of Internet Kiosks and shared computing environments on the rise, session tokens take on a new risk. A browser only destroys session cookies when the browser thread is torn down. Most Internet kiosks maintain the same browser thread. It is therefore a good idea to overwrite session cookies when the user logs out of the application.

Page Tokens

Page specific tokens or "nonces" may be used in conjunction with session specific tokens to provide a measure of authenticity when dealing with client requests. Used in conjunction with transport layer security mechanisms, page tokens can aide in ensuring that the client on the other end of the session is indeed the same client which requested the last page in a given session. Page tokens are often stored in cookies or query strings and should be completely random. It is possible to avoid sending session token information to the client entirely through the use of page tokens, by creating a mapping between them on the server side, this technique should further increase the difficulty in brute forcing session authentication tokens.

Check Your Progress 3

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) What is a cookie?

.....

.....

.....

.....

2) Explain the concept behind session tokens.

.....

.....

.....

.....

3) What is the disadvantage in using session tokens?

.....

.....

.....

.....

4) What do you mean by session re-authentication?

.....

.....

.....

.....

3.6 LET US SUM UP

This unit is an effort towards answering some of the fundamental queries about the principles of secure application development. As discussed the major principles are authentication, authorization and session management. Authentication is mainly required to tie the other two principles together. We have tried to give an overview of how to authentication is done and what are the various modes of authentication. With number of advantages of each scheme, it has some disadvantages and problems. Then we discussed the next phase that is authorization. Although it is similar to access control but for authorization you must have authenticated your identity first only then you get access to the desired information. Therefore the various types of access control

are discussed next in this unit. And then comes the last principle of session management which is used to manage the authorization rights for a specific session. Therefore various mechanisms used for session management are discussed. Depending upon the confidentiality, privacy etc you can identify which scheme to be used for you application.

3.7 CHECK YOUR PROGRESS: THE KEY

Check Your Progress 1

1) Authentication systems involve parties who play three key roles in the establishment of truth of claim.

   1. Presenter: Who presents the credentials of the veracity.
   2. Issuer: A third part who issues the credentials.
   3. Verifier: Who verifies the credentials for their veracity.

2) Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols that provide communication security over the Internet. TLS and SSL encrypt the segments of network connections above the Transport Layer, using asymmetric cryptography for privacy and a keyed message authentication code for message reliability.

   Several versions of the protocols are in widespread use in applications such as web browsing, electronic mail, Internet faxing, instant messaging and voice-over-IP (VoIP).

3) User authentication which is a process to determine that a user is who he/she claims to be and entity authentication a process to determine an entity who it claims to be. Generally entity authentication is used to in case of session management.

4) Basic access authentication is a method for a web browser or other client program to provide a user name and password when making a request. Before transmission, the user name and password are concatenated with a colon separation. The resulting string is encoded with the Base 64 algorithm. Digest access authentication is a method a web server can use to negotiate credentials with a user's web browser. It uses encryption to send the password over the network which is safer than the Basic access authentication that sends plaintext. Like Basic, Digest access authentication verifies that both parties to a communication know a shared secret (a password); unlike Basic, this verification can be done without sending the password in the clear, which is Basic's biggest weakness. As with most other authentication protocols, the greatest sources of risks are usually

found not in the core protocol itself but in policies and procedures surrounding its use.

Check Your Progress 2

1) Authorization give you the ability to restrict access to particular destinations within your application (a module, a screen, a function, a field/object) based on the user's profile. Authorization is a process in which the authenticated user is allowed access only to the resources he has been authorized to interact with. It is necessary because of the following reasons:

   - Allows the user to access particular systems and functions within those systems.
   - The granularity of the user's access might extend to allow control to particular objects within a system.

2) Access control sounds very similar to authorization, but there is a subtle difference: access control refers to the permissions assigned to an authenticated person or entity. In other words, access control first requires successful authentication, and then it can use the identity of the logged-in user to determine to which resources he has access. However, it is also possible to enforce access control without any authentication taking place, which in most cases will revert to limited/guest access. An example of this is an e-commerce application that allows users to shop without logging in. While authorization gives you the ability to restrict access to particular destinations within your application (a module, a screen, a function, a field/object) based on the user's profile. Thus in case of authorization authentication is must while for access control its optional based certain criterias.

3) Any of the access control mechanism can be used for the web application development depending upon the requirement of the system. The various access control mechanism which are frequently used for web application development are attribute based access control, Discretionary Access Control, Mandatory Access Control and Role Based Access Control

4) In case of role based access control we have Three primary rules are defined which are as follows:

   1. Role assignment: A subject can execute a transaction only if the subject has selected or been assigned a role.
   2. Role authorization: A subject's active role must be authorized for the subject. With rule 1 above, this rule ensures that users can take on only roles for which they are authorized.

3. Transaction authorization: A subject can execute a transaction only if the transaction is authorized for the subject's active role. With rules 1 and 2, this rule ensures that users can execute only transactions for which they are authorized.

Check Your Progress 3

1) A **cookie**, also known as an **HTTP cookie**, **web cookie**, or **browser cookie**, is used for an origin website to send state information to a user's browser and for the browser to return the state information to the origin site. The state information can be used for authentication, identification of a user session, user's preferences, shopping cart contents, or anything else that can be accomplished through storing text data.

   Cookies are not software. They cannot be programmed, cannot carry viruses, and cannot install malware on the host computer. However, they can be used by spyware to track user's browsing activities

   There are two categories of cookies, secure or non-secure and persistent or non- persistent, with which we can have the following four combinations.

   1. Persistent and Secure
   2. Persistent and Non-Secure
   3. Non-Persistenet and Secure
   4. Non-Persistent and Non-Secure

2) A session token is a unique identifier that is generated and sent from a server to a client to identify the current interaction session. The client usually stores and sends the token as an HTTP cookie and/or sends it as a parameter in GET or POST queries. The reason to use session tokens is that the client only has to handle the identifier—all session data is stored on the server (usually in a database, to which the client does not have direct access) linked to that identifier. Examples of the names that some programming languages use when naming their HTTP cookie include JSESSIONID (JSP), PHPSESSID (PHP), and ASPSESSIONID (ASP).

3) The disadvantage of using session token is large overhead in terms of system resources, and that the session may be interrupted if the system is restarted.

4) Critical user actions such as money transfer or significant purchase decisions should require the user to re-authenticate or be reissued another session token immediately prior to significant actions. Developers can also somewhat segment data and user actions to the extent where re-

authentication is required upon crossing certain "boundaries" to prevent some types of cross-site scripting attacks that exploit user accounts.

3.8 SUGGESTED READINGS

- Bhargav, Abhay and Kumar, B.V. *(2011), Secure java: for web development*, CRC Press publication.
- Miller, Frederic P., Vandome, Agnes F. and McBrewster, John (2010), *Authorization:Authorization, Information Security, Computer Security, Access Control, Human Resources, Authentication, Data, Computer Program, Personal Computer Hardware, Application Software, Security Engineering Computer Today*, Alphascript Publishing.
- http://www.um.es/atica/documentos/OWASPGuide2.0.1.pdf
- Smith, Richard E. (2002), *Authentication: From passwords to public key*, Addison Wesley publication

UNIT 4 ENCRYPTION, CONFIDENTIALITY AND DATA PROTECTION

Structure

4.0 Introduction
4.1 Objectives
4.2 Need of Secure Applications
4.3 Encryption
4.4 Cryptography
4.5 Confidentiality
4.6 Data Protection
4.7 Eight Steps for Integrating Security into Application Development
4.8 Let Us Sum Up
4.9 Check Your Progress: The Key
4.10 Suggested Readings

4.0 INTRODUCTION

Data security is a matter of great concern. To protect data different techniques are used by the organizations. Especially when the data in used on the internet and it moves in the cyber world from one compute to another computer crossing different geographical boundaries. There are chances of data theft on the way by some malicious users. To protect data against such threats the data protection is a must. Here in this unit some techniques like encryption are used to provide the overview of how we can safeguard our data against such threats.

4.1 OBJECTIVES

After studying this unit, you should be able to describe:

- Encryption;
- Cryptography;
- Algorithm;
- Confidentiality; and
- Data protection.

4.2 NEED OF SECURE APPLICATIONS

Organizations consist of large amount information which is asset to the organization. And if a company wants to sustain in this competitive world it

should intelligently share this information with its customers, business partners and employees. This asset should be well protected from threats which may lead to financial loss and other harm to the company. Examples of such loss can be disclosure of trade secrets, damaged reputation, decreased customer confidence, etc.

The aim of computer security is to find ways to protect this asset by the selection and application of appropriate safeguards. Successful companies arrange a computer security strategy known as defense-in-depth, which is a layered approach that relies on people, operations and intelligent application of multiple techniques and technologies to achieve the desired level of information assurance by arranging the effective safeguards intelligently, companies are able to manage the risk factors by decreasing the vulnerabilities to threats, which results in less possibility of compromise and financial consequences.

Companies depend on computing services and applications to share information assets. These services and applications consist of a combination of software and custom applications that provide solutions to meet business goals. These custom applications are created in in-house and are increasingly interconnected with companies, intranets and internet. By using these applications companies develop the business opportunities and these applications are vulnerable to compromise.

4.3 ENCRYPTION

Encryption is a process of changing information (plaintext) using an algorithm (cipher) to convert it into a form that is not readable to anyone except those possessing special knowledge , referred to as key. By following the above mentioned process we obtain a encrypted information (in cryptography referred to as ciphertext). In many contexts, the word encryption also implicitly refers to the reverse process, decryption (e.g. "software for encryption" can also perform decryption) to convert the encrypted piece of information into understandable format.

Encryption is not new, it has been in use since long, it was used by military and government to facilitate secret communication. Encryption is now commonly used in protecting information within many kinds of civilian systems. For example, the Computer Security Institute reported that in 2007, 71% of companies surveyed utilized encryption for some of their data in transit, and 53% utilized encryption for some of their data in storage. Encryption can be used to prevent data from loss, data which is "at rest" files such as files on computers and storage devices (e.g usb flash drive). In few recent years we have come across cases which involve theft of customer's personal data through the theft of laptops or backup drops. Encryption of such helps them to

be protected. Digital rights management systems which prevent unauthorized use or reproduction of copyrighted material and protect software against reverse engineering (see also copy protection) are another somewhat different example of using encryption on data at rest.

The process of encryption also helps also has advantage in protecting data in transit , for example data being transferred via networks (e.g. the Internet , e-commerce), mobile phones , wireless microphones , wireless intercom systems , Bluetooth devices , and bank automatic teller machines .There have been numerous reports of data in transit being intercepted in recent years . Encryption helps to protect data.

Encryption can maintain the privacy of the message but still other techniques are required to protect the integrity and authenticity of the message. For example, verification of a message authentication code (MAC) or a digital signature. Many cryptographic software and hardware and standards are available but security is still a challenging problem .A single slip in system design can allow attacks to be successful. Sometimes an adversary can obtain unencrypted information without directly undoing the encryption

One of the earliest public key encryption applications was called Pretty Good Privacy (PGP). It was written in 1991 by Phil Zimmermann and was purchased by Symantec in 2010.

In order to avoid tampering we should make sure that the digital signature and encryption should be applied at the message creation time. Or else any node between the sender and the encryption agent could potentially tamper it. It has been seen that due to the manufacturers ignorance, most mobile devices come without user – controlled encryption or digital signature capabilities.

METHODS OF ENCRYPTING DATA

There are three types of methods of encryption (cryptography)

1. Private key/Secret key or symmetric key cryptography
2. Public key/ Asymmetric key cryptography
3. Hash Function

Public-key/Asymmetric Key cryptography

By the last 300-400 years Public-key cryptography has become a new development in cryptography. Stanford University professor Martin Hellman and graduate student Whitfield Diffie described Modern PKC publicly in 1976. They described a two-key crypto system in their system in which two parties could communicate a secure communication over a non-secure communications channel without sharing a secret key.

The so-called one-way functions or mathematical functions are existed in PKC that are easy to compute whereas their inverse function is relatively difficult to compute. Let us consider by taking 2 simple examples.

Multiplication vs. factorization: Suppose we have two numbers, 9 and 16, and that we want to calculate the product; the product is being calculated in almost no time i.e. 144. Now suppose that we have a number, 144, and we need to tell the pair of integers which are multiplied together to obtain that number. we will eventually come up with the solution whereas calculating the product took milliseconds, and factoring will take longer because we first need to find the 8 pair of integer factors and then determine which one is the correct pair.

Exponentiation vs. logarithms: Suppose we want to take the number 3 to the 6th power; which is easy to calculate 36=729. But if we have the number 729 and want to tell the two integers that we used that are x and y such that logx 729 = y, it will take more time to find all possible

The examples given above represent two of the functional pairs that are used with PKC which are at the ease of multiplication and exponentiation versus the relative difficulty of factoring and calculating logarithms, respectively. The mathematical "trick" in PKC is to find a trap door in the one-way function so that the inverse calculation becomes easy by having some knowledge of information.

Generic PKC consists of two keys which are related mathematically where knowledge of one key does not make one to easily determine the other key.

The function of one key is to encrypt the plaintext and the other key is used to decrypt the cipher text. The importance of these keys are the requirement of both for the process to work, this approach is also called asymmetric cryptography.

Private Key/Secret Key/Symmetric Key Cryptography

In secret key cryptography, a single key is used for both encryption and decryption. The sender uses the key or some rules to encrypt the plaintext and sends the cipher text to the receiver. The receiver applies the same key or set of rules to decrypt the received message and to recover the plaintext. As a single key is used for both the functions, it is also called symmetric encryption.

In this form of cryptography, the key is known to both the sender and the receiver which is secret. The major difficulty with this approach is of course the distribution of the key.

Secret key cryptography schemes are being categorized either as stream ciphers or block ciphers. Stream ciphers operate on a single bit at a time and implement feedback mechanism to keep the key constantly changing and block cipher encrypts one block of data at a time using the same key on each block. In block

cipher the same plaintext block will always encrypt to the same cipher text using the same key where as in stream cipher the same plaintext will encrypt to different cipher text

Stream ciphers come in several flavors. In Self-synchronizing stream ciphers each bit is calculated in the key stream as a function of the previous n bits in the key stream. It is termed "self-synchronizing" because the decryption process can stay synchronized with the encryption process by knowing its farness from the n-bit key stream. The one problem consists is error propagation in which one garbled bit in transmission will result in n garbled bits at the receiving side. Synchronous stream ciphers generate the key stream in a style independent of the message stream by using the same key stream generation function both at the sender and receiver side. While stream ciphers do not consist of transmission errors as they are periodic in nature by which the key stream will eventually repeat.

Block ciphers functions in one of several modes n which the following four are the most important

Electronic Codebook (ECB) mode is the simplest application in which the secret key is used to encrypt the plaintext block to form a cipher text block. Two identical plaintext blocks always generate the same cipher text block. While it is the most common mode of block ciphers but is susceptible to a variety of brute-force attacks.

Cipher Block Chaining (CBC) mode adds a feedback mechanism to the encryption scheme. In CBC, the plaintext is exclusively-ORed (XORed) with previous ciphertext block prior to encryption. In this mode the two identical blocks of plaintext never encrypt to the same cipher ext.

Cipher Feedback (CFB) mode is a block cipher implementation as a self-synchronizing stream cipher. CFB mode allows data to be encrypted in units smaller than the block size, which might be useful in some applications such as encrypting interactive terminal input. If we were using 1-byte CFB mode, for example, each incoming character is placed into a shift register the same size as the block, encrypted, and the block transmitted. At the receiving side, the ciphertext is decrypted and the extra bits in the block (i.e., everything above and beyond the one byte) are discarded.

Output Feedback (OFB) mode is a block cipher implementation conceptually similar to a synchronous stream cipher. OFB prevents the same plaintext block from generating the same ciphertext block by using an internal feedback mechanism that is independent of both the plaintext and ciphertext bitstreams.

Hash Function

A deterministic procedure known as cryptographic hash function takes an arbitrary block of data and returns a fixed-size bit string known as the (cryptographic) hash value where an accidental or intentional change to the data will change the hash value. The encoded data is called the "message," and the hash value is known as the message digest or simply digest.

The ideal cryptographic hash function consumes four significant properties:

- Computation of the hash value for any given message is easy (but not necessarily quick).
- Generating a message that has a given hash is infeasible.
- Modification a message without changing the hash is infeasible.
- Finding two different messages with the same hash is also infeasible.

Cryptographic hash functions have many information security applications, notably in digital signatures, message authentication codes (MACs), and other forms of authentication. They can also be used as ordinary hash functions, to index data in hash tables, for fingerprinting, to detect duplicate data or uniquely identify files, and as checksums to detect accidental data corruption. Indeed, in information security contexts, cryptographic hash values are sometimes called (digital) fingerprints, checksums, or just hash values, all these terms stand for functions having different properties and purposes.

APPLICATIONS OF ENCRYPTION

Its Private use - the encryption software packages are available commercially and by free download from the Internet. One free e-mail encryption package is Pretty Good Privacy (PGP) is available since 1991. An alternative is S/MIME, supported by most e-mail vendors.

Its Securing networks - an encryption protocol called Sockets Layer (SSL) provides secure communications and user authentication over open unsecured networks like the Internet. It is widely used in web browsers by a small padlock icon, for example when an online user submits credit card details. This system protects the data and also checks that a given website is authentic and also verifies the identity of the user. private networks are used using similar networks.

Wireless (or Wi-Fi) networks are also vulnerable to interception. An international security standard called Wi-Fi Protected Access (WPA2) can be applied to encrypt data sent over wireless networks.

Access control - Digital television providers control the subscriber's access by encrypting audio and video signals. Subscribers have a descrambling

device comprising of the decryption algorithm and decryption key, which decrypts the pictures and sound.

4.4 CRYPTOGRAPHY

Cryptography can be defined as the practice and study of hiding information. Modern cryptography follows the disciplines of mathematics, computer science, and electrical engineering. ATM cards, computer passwords and electronic commerce are some of the applications of cryptography.

Cryptography was the synonym of encryption, before the modern age , this is the process of converting the message from understandable form to non – understandable form.

The sender maintains the ability to decrypt the information therefore avoid unwanted person from reading it. Since the computer computer came into existence the methods for cryptography are becoming increasingly complex.

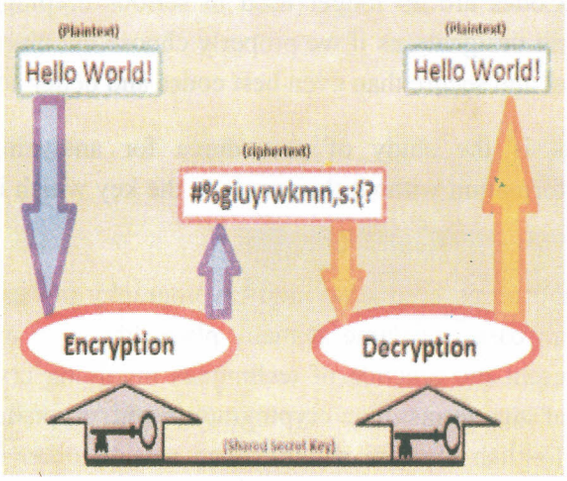


Fig. 1

Terminology

Modern cryptography is designed based on facts like scientific approach, designs cryptographic algorithms around computational hardness assumptions, because of all these factors it is very difficult to break the message .It is not possible to break such systems in theory as well as practically. Because of such schemes our data becomes secure. There exist information-theoretically secure schemes that provably cannot be broken—an example is the one-time pad--but these schemes are more difficult to implement than the theoretically breakable but computationally secure mechanisms. Some legal issues are raised due to cryptography and some of them are still to be solved.

Terms like cryptography and encryption meant the same until modern times, which is the process of converting ordinary information which is called

plaintext into unintelligible gibberish which is called ciphertext. The opposite of encryption is decryption, in which, the opposite of encryption happens, it involves converting text from unreadable ciphertext back to plaintext. The term Ciphers is used to represent a pair of algorithms that create the encryption and decryption. Both the Algorithm and instance by a key control the operation of cipher. Which is a secret parameter and is known only to the communicants for a specific message exchange context. A "cryptosystem" can be defined as a ordered list of elements of finite possible plaintexts, finite possible cyphertexts, finite possible keys, and the encryption and decryption algorithms which correspond to each key. Keys constitute the important portion of the ciphers, as ciphers without variable keys can be easily broken with only the knowledge of the cipher used and are therefore considered useless for most purposes. Earlier authentication and integrity features were absent from the ciphers.

In terms of conversational use, the term "code" is used to represent encryption or hiding the meaning. However in terms of crptography it has a more specific meaning. It represents converting a unit of plaintext or ordinary text with a code word. Codes are no longer used in serious cryptography—except for things like unit designations if we properly choose the ciphers both serve to be more practical and secure than even best codes and better adapted computers.

Cryptanalysis is the study of procedures for analyzing the meaning of encrypted information without the access to the key which is normally required to do so.

Sometimes the terms cryptology and cryptography are used interchangeably. But this is not case everytime at many places the term cryptography is used only to represent the practice of techniques involving cryptography, and the integration of the terms like cryptography and cryptanalysis is termed as cryptology. English is more suitable than several other languages in which cryptology is always used in the second sense above. In the English Wikipedia the general term used for the complete field is cryptography which is done by cryptographers.

Cryptolinguistics represents the study of features of the language which can be used in cryptography.

History of Cryptography and Cryptanalysis

Privacy of the message and maintaining it was the only concern of cryptography before modern era—conversion of messages from a readable form into an unreadable one and back again at the other end, rendering it unreadable by interceptors or eavesdroppers without secret knowledge. The main objective of encryption was to maintain the secrecy in communications, such as those of spies, military leaders, and diplomats. But today the message integrity is also taken care of, sender/receiver identity authentication, digital signatures, interactive proofs and secure computation, among others.

Classic Cryptography

In the ancient times anything that was secret or secret writing were written only with local pen and paper analogs, as most people could not read , so could not understand. As the time moved on and the literacy rate began to increase, at that time the requirement of literacy was felt. The main classical ciphers like transposition ciphers, in which rearrangement of the order of letters in a message takes place for example 'hello world' becomes 'ehlol owrdl' in a trivially simple rearrangement scheme, and in substitution ciphers, we systematically replace letters or groups of letters with other letters or groups of letters for example, 'fly at once' becomes 'gmz bu podf' by replacing each letter with the one following it in the Latin alphabet. Simple versions of ciphers have never provided us with confidentiality from enterprising opponents. An early form of substitution cipher was the Caesar cipher, in which we replace each letter in the ordinary text by a letter some fixed number of positions further down the alphabet. Suetonius reports that Julius Caesar used it with a shift of three to communicate with his generals. Atbash is an example of an early Hebrew cipher. The earliest known use of cryptography is some carved ciphertext on stone in Egypt (ca 1900 BC), but this may have been done for the amusement of literate observers rather than as a way of concealing information. Cryptography is recommended in the Kama Sutra as a way for lovers to communicate without inconvenient discovery.

The Greeks of Classical times were known for ciphers. Steganography was also first developed in earlier times. An early example, from Herodotus, concealed a message—a tattoo on a slave's shaved head—under the regrown hair. Another method which was followed by greeks was developed by Polybius which is now known as “Polybius Square". Some more examples of steganography involves the use of invisible ink, microdots, and digital watermarks to conceal information and to maintain the privacy of the message.

The result of classical ciphers is the plaintext which always reveals statistical information about the plaintext, which can be of use to break the message .Since the discovery of frequency analysis perhaps by the Arab mathematician and polymath, Al-Kindi also known as Alkindus, in the 9th century, nearly all such ciphers were easily broken by the informed attacker. Such classical ciphers still are popular today, though mostly as puzzles. Al-Kindi also wrote a book on cryptography entitled Risalah fi Istikhraj al-Muamma which means Manuscript for the Deciphering Cryptographic Messages, in which he defined very first cryptanalysis techniques.16th-century book-shaped French cipher machine, with arms of Henri II of France Enciphered letter from Gabriel de Luetz d'Aramon, French Ambassador to the Ottoman Empire, after 1546, with partial decipherment

Essentially all ciphers remained capable of being attacked by cryptanalysis using the frequency analysis technique till the development of the

polyalphabetic cipher, most clearly by Leon Battista Alberti around the year 1467, though there is some indication that it was already known to Al-Kindi. Alberti's idea was to make use of different ciphers like, substitution of alphabets for some portion of a message perhaps for each successive plaintext letter at the limit. He also developed probably the first automatic cipher device,which consisted of a wheel which implemented a partial realization of his invention. In the polyalphabetic Vigenère cipher, encryption has a key word, which decides letter substitution depending on which letter of the key word is used. In the mid-19th century Charles Babbage showed that the Vigenère cipher was vulnerable to Kasiski examination, but this was first published about ten years later by Friedrich Kasiski.

Frequency analysis is considered a powerful and general technique against many ciphers, but apart from frequency analysis ,encryption has still been often effective in practice; many a new cryptanalyst or people who were about to become cryptanalyst were not aware of the technique. Without making the use of frequency analysis breaking a message needed the knowledge of the cipher used and also the key used, thus making espionage, bribery, burglary, defection, etc., more attractive approaches to the cryptanalytically uninformed. It was finally explicitly recognized in the 19th century that secrecy of a cipher's algorithm is not a sensible nor practical safeguard of message security; in fact, a need of making the cryptographic scheme secure was felt even if the adversary fully understands the cipher algorithm itself. Maintaining the security of the key used should alone be sufficient for a good cipher to maintain confidentiality under an attack. This fundamental principle was first explicitly stated in 1883 by Auguste Kerckhoffs and is generally called Kerckhoffs's Principle; alternatively and more bluntly, it was restated by Claude Shannon, the inventor of information theory and the fundamentals of theoretical cryptography, as Shannon's Maxim—'the enemy knows the system'.

Various kinds of physical devices and aids have been used to help with ciphers. One of them which is believed to be the earliest is the scytale of ancient Greece, a rod supposedly used by the Spartans as an aid for a transposition cipher. In medieval times, other aids were invented like cipher grille, which was also used for a kind of steganography. Polyalphabetic ciphers was considered a bit sophisticated aid such as Alberti's own cipher disk, Johannes Trithemius' tabula recta scheme, and Thomas Jefferson's multi-cylinder (not publicly known, and reinvented independently by Bazeries around 1900). Early in the 20th century, many mechanical encryption/decryption devices were invented and several patented, some of them are rotor machines—famously including the Enigma machine used by the German government and military from the late '20s and during World War II. The ciphers implemented by better quality examples of these machine designs brought about a substantial increase in cryptanalytic difficulty after WWI.

The Computer Era

The invention of digital computers and electronics made the ciphers more complex after WWII. Furthermore,as we know computers make use of many different kinds of data and allowed encryption for all of them ,representable in any binary format, unlike classical ciphers which only encrypted written language texts; this was new and significant. The use of Computer has replaced language cryptography, both for cipher design and cryptanalysis. Ciphers can be categorized by their operation on binary bit sequences, which is not same as classical and mechanical schemes which generally manipulate traditional characters directly. However, computers also assisted cryptanalysis, which serves as a solution to some extent for increased cipher complexity. Nonetheless, good modern ciphers have stayed ahead of cryptanalysis; it is typically the case that use of a quality cipher is very efficient, breaking cipher is difficult and requires an effort that is many orders of magnitude larger, and vastly larger than that required for any classical cipher, making cryptanalysis inefficient and impractical as to be effectively impossible.

Credit card with smart-card capabilities. The 3-by-5-mm chip embedded in the card is shown, enlarged. Smart cards combine low cost and portability with the power to compute cryptographic algorithms

Extensive open academic research on cryptography is done recently; it started in mid-1970s. The algorithm designed by IBM personnel reently became the Federal (i.e., US) Data Encryption Standard; Whitfield Diffie and Martin Hellman published their key agreement algorithm, and the RSA algorithm was published in Martin Gardner's Scientific American column. Since then, cryptography has applications like communications, computer networks, and computer security generally. Some modern cryptographic techniques can only keep their keys secret if certain mathematical problems are intractable, such as the integer factorization or the discrete logarithm problems, so there are deep connections with abstract mathematics. Cryptographic techniques are still not secure (but see one-time pad); at best, there are instances that show that some techniques are secure if some computational problem is not possible to solve, or this or that assumption about implementation or practical use is met.

Cryptographic history is important but system designers must pay attention on the future developments while designing the algorithms. For example, with the increased improvements in computer processing power, there is possibility of brute-force attacks, so the length of key required also increases. Potential effects of quantum computing are also taken into account by some cryptographic system designers; the announced imminence of small implementations of these machines may be making the need for this preemptive caution rather more than merely speculative.

Language and editing patterns in cryptography were main focus before 20th century. Since then the focus has shifted, and cryptography now makes use of mathematics, including aspects of information theory, computational complexity, statistics, combinatorics, abstract algebra, number theory, and finite mathematics generally. Cryptography can be considered a part of engineering, but an unusual one as it deals with active, intelligent, and malevolent opposition other kinds of engineering need deal only with neutral natural forces. There is also active research examining the relationship between cryptographic problems and quantum physics.

Symmetric-Key Cryptography

In symmetric key cryptography the sender and the receiver both share the same key. This is also known as symmetric – key cryptography .This was the only kind of encryption publicly known until June 1976.

The modern study of symmetric-key ciphers focuses mainly on the study of block ciphers and stream ciphers and the areas where they are used. A block cipher is a modern abstraction of Alberti's polyalphabetic cipher: block ciphers is the one that takes as input a block of plaintext and a key, and output a block of ciphertext of the same size. The length of messages is larger than a single block, so we need some method of combining together successive blocks. Some methods have been developed, some of them offer better security in one aspect and some offer better security in another aspect. They are the modes of operation and must be carefully considered when using a block cipher in a cryptosystem.

The Data Encryption Standard (DES) and the Advanced Encryption Standard (AES) comes under block cipher designs which have been designated cryptography standards by the US government. Despite the fact that DES is disapproved by an official standard, DES remains quite popular; it is used in a various areas, from ATM encryption to e-mail privacy and secure remote access. Many other block ciphers are there, which have been designed and released, with some changes made in the quality. Many have been thoroughly broken, such as FEAL.

Exactly opposite in nature are stream ciphers, which are 'block' type, which create an arbitrarily long stream of key material, which is integrated with the plaintext in a bit-by-bit pattern or character-by-character pattern , like the one-time pad. In a stream cipher, it is the cipher that operates and changes the hidden internal state as output is generated. That internal state is initially set up using the secret key material. RC4 is a widely used stream cipher; see Category: Stream ciphers. Block ciphers can be used as stream ciphers; see Block cipher modes of operation.

The third type of cryptographic algorithm uses cryptographic hash functions. In this third type of cryptographic algorithm input of any message length is taken,

and a short output is generated, fixed length hash which can be used in a digital signature. If the hash function is good, an attacker cannot find two messages that produce the same hash. MD4 is a long-used hash function which is now broken; MD5, a strengthened variant of MD4, is also widely used but broken in practice. The U.S. National Security Agency developed the Secure Hash Algorithm series of MD5-like hash functions: SHA-0 was a flawed algorithm that the agency withdrew; SHA-1 is widely deployed and more secure than MD5, but cryptanalysts have identified attacks against it; the SHA-2 family improves on SHA-1, but it isn't yet widely deployed, and the U.S. standards authority thought it "prudent" from a security perspective to develop a new standard to "significantly improve the robustness of NIST's overall hash algorithm toolkit." Thus, there is a design competition occurring and its aim is to select a new U.S. national standard, to be called SHA-3, by 2012.

Message authentication codes (MACs) are similar to cryptographic hash functions, except that in MACs a secret key can be used to authenticate the hash value upon receipt.

Public-Key Cryptography

As we know in symmetric-key cryptography the same key is used for encryption and decryption of a message, though a message or group of messages may have a different key than others. A disadvantage of symmetric ciphers is that, for security reasons key management is necessary. Each distinct pair of communicating parties must, ideally, share a different key, and perhaps each ciphertext exchanged as well. The number of keys used increases as the square of the number of network members, due to which a complex key management system is required to keep them all straight and secret. The difficulty of securely establishing a secret key between two communicating parties,raises a problem , when a secure channel is not present between them , which represents chicken-and-egg problem which is a considerable practical obstacle for cryptography users in the real world.

Whitfield Diffie and Martin Hellman, authors of the first published paper on public-key cryptography

In a groundbreaking 1976 paper, Whitfield Diffie and Martin Hellman proposed the concept of public-key or asymmetric key cryptography in which two distinct but mathematically related keys are used—a public key and a private key. A public key system has such a structure that manipulating one key (the 'private key') is computationally not possible from the other (the 'public key'), in spite of the fact that they are related to each other . Instead, both keys are generated secretly, as an interrelated pair.The historian David Kahn described public-key cryptography as "the most revolutionary new concept in the field since polyalphabetic substitution emerged in the Renaissance".

The public key may be freely distributed in public key cryptosystems, but its paired private key must be kept secret. The public key is mainly used for encryption, while the private or secret key is used for decryption. Diffie and Hellman showed that public-key cryptography was possible by presenting the Diffie–Hellman key exchange protocol.

In 1978, Ronald Rivest, Adi Shamir, and Len Adleman invented RSA, another type of public-key system. In 1997,it was publicly known and was called as asymmetric key cryptography had been invented by James H. Ellis at GCHQ, a British intelligence organization, and that, in the early 1970s, both the Diffie–Hellman and RSA algorithms had been previously developed (by Malcolm J. Williamson and Clifford Cocks, respectively).

The Diffie–Hellman and RSA algorithms, were not only the first publicly known algorithms but were also considered high quality public key algorithms, and they have been among the most widely used. Others include the Cramer–Shoup cryptosystem, ElGamal encryption, and various elliptic curve techniques. See Category: Asymmetric-key cryptosystems.

Padlock icon from the Firefox Web browser, was invented too express a page that has been sent in SSL or TLS-encrypted protected form. Though, this icon does not guarantee security; any corrupt browser might misguide a user by having such an icon when a transmission is not actually being protected by SSL or TLS.

Public- key cryptography can be used to implement digital signatures. Digital signature is similar to an ordinary signature; they both share common characteristic that they both are easy for a user to generate, but comparatively difficult for anyone else to imitate. Digital signatures can also be permanently attached to a content of the message being signed; they cannot then be transmitted from one document to another, if tried to do so the attempt will be traceable. In digital signature schemes, there are two algorithms: one for signing, in which a secret key is used to process the message and one for verification, in which the matching public key is used with the message to check the validity of the signature. RSA and DSA fall under the category of the most popular digital signature schemes. Digital signatures are central to the operation of public key infrastructures and many network security schemes (e.g., SSL/TLS, many VPNs, etc.).

Computational complexity of "hard" problems, from number theory served the basis for derivation of public key algorithms. For example, the hardness of RSA is related to the integer factorization problem, while Diffie–Hellman and DSA are related to the discrete logarithm problem. More recently, elliptic curve cryptography has developed in which security is based on number theoretic problems involving elliptic curves. Most of the public key algorithms are made up of operations like modular multiplication and exponentiation because of the

difficulty in underlying problems, which are much more computationally expensive than the techniques used in most block ciphers, especially with typical key sizes. As a consequence, public-key cryptosystems are a combination of cryptosystems, in which a fast high-quality symmetric-key encryption algorithm is used for the message itself, while the relevant symmetric key is sent with the message, but is encrypted using a public-key algorithm. Similarly, combination of signature schemes are often used, in which a cryptographic hash function is calculated, and only the resulting hash is digitally signed.

Cryptanalysis

Different variety of the Enigma machine, used by Germany's military and civil authorities from the late 1920s through World War II, performed a complex electro-mechanical polyalphabetic cipher. Breaking and reading of the Enigma cipher at Poland's Cipher Bureau, for 7 years before the war, and subsequent decryption at Bletchley Park, was important to Allied victory.

The objective of cryptanalysis is to find defects or insecurity in a cryptographic scheme, thus permitting its destruction.

It is not right to believe that every encryption method can be broken. In connection with his WWII work at Bell Labs, Claude Shannon proved that the one-time pad cipher cannot be broken, but the constraint for this to take place is that the key material should be random, never reused, kept secret from all possible attackers, and should be of equal or greater length than the message. Most ciphers, apart from the one-time pad, can be broken with enough computational effort by brute force attack, but the amount of effort needed may be exponentially dependent on the key size, as compared to the effort needed to make use of the cipher. In such cases, security can be proved if the effort required (i.e., "work factor", in Shannon's terms) which is beyond the ability of any adversary. This means it must be proved that there is presently no method which can be performed in best possible manner to break the cipher. Since it is not proved till date, the one-time-pad remains the only theoretically unbreakable cipher.

There are wide variety of cryptanalytic attacks, and they can be categorized into several ways. A common difference that can be shown by what an attacker knows and what capabilities are available. In a ciphertext-only attack, the cryptanalyst can access ciphertext. In a known-plaintext attack, the cryptanalyst has right to see the ciphertext and it's corresponding plaintext. In a chosen-plaintext attack, the cryptanalyst may choose a plaintext and learn its corresponding ciphertext; an example is gardening, used by the British during WWII. Finally, in a chosen-ciphertext attack, the cryptanalyst may be able to choose ciphertexts and learn their corresponding plaintexts.

Poznań monument (center) to Polish cryptologists whose breaking of Germany's Enigma machine ciphers, beginning in 1932, altered the course of World War II.

Cryptanalysis of symmetric-key ciphers consists of finding out attacks against the block ciphers or stream ciphers that are more efficient than any attack that could be against a perfect cipher. For example, a simple brute force attack against DES needs one known plaintext and 255 decryptions, trying approximately half of the possible keys, to reach a point at which chances are better than even the key sought will have been found. But this may not be enough assurance; a linear cryptanalysis attack against DES requires 243 known plaintexts and approximately 243 DES operations. This is a considerable improvement on brute force attacks.

Computational difficulty of various problems forms the basis of public- key algorithms. The most popular of these is integer factorization for example, the RSA algorithm is based on a problem related to integer factoring, but the discrete logarithm problem is also important. Much public-key cryptanalysis includes numerical algorithms for arriving at a solution of these computational problems, or some of them, efficiently. For example, the best known algorithms for solving the elliptic curve-based version of discrete logarithm are much more time-consuming than the best known algorithms for factoring, at least for problems of more or less equivalent size. Thus, other things being equal, if we want similar strength of attack resistance, factoring-based encryption techniques must use larger keys than elliptic curve techniques. For the above mentioned reason, public-key cryptosystems based on elliptic curves have become famous since their invention in the mid-1990s.

While pure cryptanalysis makes use of drawbacks in the algorithms themselves, other attacks on cryptosystems are based on actual use of the algorithms in real devices, and are called side-channel attacks. If a cryptanalyst has access to, for example, the amount of time the device took to encrypt a number of plaintexts or report an error in a password or PIN character, he may be able to use a timing attack to break a cipher that is otherwise resistant to analysis. An attacker might also go through the pattern and length of messages to collect some valuable information; this is termed as traffic analysis, and can be quite useful to an alert adversary. Cryptosystems if poorly managed, for instance allowing the use of too short keys, will make any system vulnerable, regardless of other virtues. And, of course, social engineering, and other attacks against the personnel who work with cryptosystems or the messages they handle (e.g., bribery, extortion, blackmail, espionage, torture ...) may be the most productive attacks of all.

Cryptographic Primitives

Most of the theoretical work done in cryptography is based on development made in cryptography used earlier—algorithms with basic cryptographic properties—and their relationship to other cryptographic problems. We can invent much more complex cryptographic tools from the basic primitives. These primitives provide fundamental properties, which are used to develop more complex tools called cryptosystems or cryptographic protocols, which guarantee one or more high-level security properties. Note however, that the difference between cryptographic primitives and cryptosystems is quite not supported; for example, the RSA algorithm is sometimes considered a cryptosystem, and sometimes a primitive. Typical examples of cryptographic primitives include pseudorandom functions, one-way functions, etc.

Cryptosystems

The concept used in cryptography prior to the existing cryptography are often used to develop more complex algorithms, called a cryptographic system, or cryptosystem. Cryptosystems like El-Gamal encryption are developed to provide specific functionality for example public key encryption while guaranteeing certain security properties. Cryptosystems use the properties of the underlying cryptographic primitives to support the system's security properties. Of course, as the distinction between primitives and cryptosystems is somewhat arbitrary, a sophisticated cryptosystem can be developed from integration of some or more primitive cryptosystems. In many instances, the cryptosystem's structure involves to and fro communication among two or more parties in space (e.g., between the sender of a secure message and its receiver) or across time (e.g., cryptographically protected backup data).The type of cryptosystems described above are termed as cryptographic protocols.

Some widely known cryptosystems include RSA encryption, Schnorr signature, El-Gamal encryption, PGP, etc. More complex cryptosystems include electronic cash systems, signcryption systems, etc. Some more 'theoretical' cryptosystems include interactive proof systems, systems for secret sharing etc.

Recently most security properties of most cryptosystems were presented using observation alone or using ad hoc reasoning. Recently, great amount of efforts have been made to develop formal techniques for establishing the security of cryptosystems; this has been generally called provable security. The general idea of provable security involves providing arguments about the computational difficulty needed to compromise some security aspect of the cryptosystem (i.e., to any adversary).

The study of how best to implement and integrate cryptography in software applications is itself a distinct field.

Legal Issues

Prohibitions

Intelligence gathering and law enforcement agencies have some interest in cryptography. Secret communications may be criminal or even treasonous. Because it provides privacy, and the reduction of privacy attendant on its prohibition, cryptography is also of considerable interest to civil rights supporters. But, there are a lot of legal issues which have been raised in context with cryptography, especially since the arrival of inexpensive computers have made widespread access to high quality cryptography possible.

In some countries, even the domestic use of cryptography is not allowed. Until 1999, France significantly restricted the use of cryptography domestically, though it has relaxed many of these. In China, a license is still needed to make use of cryptography. Many countries place limitations on the use of cryptography. Among the more restrictive are laws in Belarus, Kazakhstan, Mongolia, Pakistan, Singapore, Tunisia, and Vietnam.

In the United States, cryptography is legally allowed to be used for domestic purposes, but there has been disagreement over legal issues related to cryptography. One particularly important issue has been the export of cryptography and cryptographic software and hardware. Probably because of the importance of cryptanalysis in World War II and an expectation that cryptography would continue to be important for national security, many Western governments have, at some point, strictly regulated export of cryptography. After World War II, it was illegal in the US to sell or distribute encryption technology overseas; in fact, encryption was designated as auxiliary military equipment and put on the United States Munitions List. Until the advent of the personal computer, asymmetric key algorithms, and the Internet, the scene was not especially problematic. However, with the increase in size of internet and computers became more widely available, high quality encryption techniques became well-known around the globe. As a result, export controls came to be seen to be an impediment to commerce and to research.

Export Controls

In the 1990s, there were several challenges to US export regulations of cryptography. One involved Philip Zimmermann's Pretty Good Privacy (PGP) encryption program; it was released in the US, together with its source code, and found its way onto the Internet in June 1991. After a complaint by RSA Security immermann was criminally investigated by the Customs Service and the FBI for several years. No charges were ever filed, however. Also, Daniel Bernstein, then a graduate student at UC Berkeley, brought a lawsuit against the US government challenging some aspects of the restrictions based on free speech grounds. The 1995 case Bernstein v. United States ultimately resulted in

a 1999 decision that printed source code for cryptographic algorithms and systems was protected as free speech by the United States Constitution.

In 1996, thirty-nine countries signed the Wassenaar Arrangement, an arms control treaty that deals with the export of arms and "dual-use" technologies such as cryptography. The treaty had a condition that the use of cryptography with short key-lengths (56-bit for symmetric encryption, 512-bit for RSA) would no longer be export-controlled. Cryptography exports from the US are now much less strictly regulated than in the past as a consequence of a major relaxation in 2000; there are no longer very many restrictions on key sizes in US-exported mass-market software. In practice today, since the relaxation in US export restrictions, and because almost every personal computer connected to the Internet, everywhere in the world, includes US-sourced web browsers such as Firefox or Internet Explorer, almost every Internet user worldwide has access to quality cryptography (i.e., when using sufficiently long keys with properly operating and unsubverted software, etc.) in their browsers; examples are Transport Layer Security or SSL stack. The Mozilla Thunderbird and Microsoft Outlook E-mail client programs similarly can connect to IMAP or POP servers via TLS, and can send and receive email encrypted with S/MIME. Many Internet users don't realize that their basic application software contains such extensive cryptosystems. These browsers and email programs are so ubiquitous that even governments whose intent is to regulate civilian use of cryptography generally don't find it practical to do much to control distribution or use of cryptography of this quality, so even when such laws are in force, actual enforcement is often effectively impossible.

NSA Involvement

Another issue which involves controversy is connected to cryptography in the United States, which is the influence of the National Security Agency on cipher development and policy. NSA formed a part to design DES during its development at IBM and its consideration by the National Bureau of Standards as a possible Federal Standard for cryptography. DES was designed to refrain it from differential cryptanalysis, a powerful and general cryptanalytic technique known to NSA and IBM that became publicly known only when it was rediscovered in the late 1980s. According to Steven Levy, IBM rediscovered differential cryptanalysis, but kept the technique secret at NSA's request. The technique became publicly known only when Biham and Shamir re-rediscovered and announced it some years later. The entire scenario represented the trouble in concluding what resources and knowledge an attacker might actually have.

Another example of NSA's involvement was the 1993 Clipper chip affair, an encryption microchip intended to be part of the Capstone cryptography-control initiative. Clipper was widely criticized by cryptographers for two reasons. The cipher algorithm was then classified (the cipher, called Skipjack, though it was

declassified in 1998 long after the Clipper initiative lapsed). The secret cipher caused concerns that NSA had deliberately made the cipher weak in order to assist its intelligence efforts. The whole initiative was discussed with the merits and demerits based on its violation of Kerckhoffs's Principle, as the scheme included a special escrow key held by the government for use by law enforcement, for example in wiretaps.

Digital Rights Management

Cryptography is dependent on digital rights management (DRM), a group of techniques for technologically controlling use of copyrighted material, being widely implemented and deployed at the behest of some copyright holders. In 1998, American President Bill Clinton signed the Digital Millennium Copyright Act (DMCA), which criminalized all production, dissemination, and use of certain cryptanalytic techniques and technology (now known or later discovered); specifically, those that could be used to encompass DRM technological schemes. This had a great impact on the cryptography research community since an argument can be made that any cryptanalytic research violated, or might violate, the DMCA. Similar statutes have since been enacted in several countries and regions, including the implementation in the EU Copyright Directive. Similar restrictions are called for by treaties signed by World Intellectual Property Organization member-states.

The United States Department of Justice and FBI have not been supporting by force the DMCA as rigorously as had been feared by some, but the law, nonetheless, involves controversies. Niels Ferguson, a well-respected cryptography researcher, has publicly stated that he will not release some of his research into an Intel security design for fear of prosecution under the DMCA. Both Alan Cox (longtime number 2 in Linux kernel development) and Professor Edward Felten (and some of his students at Princeton) have encountered problems related to the Act. Dmitry Sklyarov was arrested during a visit to the US from Russia, and jailed for five months pending trial for alleged violations of the DMCA arising from work he had done in Russia, where the work was legal. In 2007, the cryptographic keys responsible for Blu-ray and HD DVD content scrambling were discovered and released onto the Internet. In both cases, the MPAA sent out numerous DMCA takedown notices, and there was a massive internet backlash triggered by the perceived impact of such notices on fair use and free speech.

CRYPTOGRAPHY ALGORITHMS

RSA

RSA algorithm stands for Ron Rivest, Adi Shamir and Leonard Adleman is used for public-key cryptography which is based on the difficulty of factoring large integers known as the factoring problem described in 1978. A user creates and publishes the product of two large prime numbers, along with an auxiliary

value, as their public key while the prime factors must be kept secret. Anyone can use the public key to encrypt a message having currently published methods, and if the public key is large enough then only someone with knowledge of the prime factors can feasibly decode the message. Whether breaking RSA encryption is as hard as factoring is an open question known as the RSA problem.

DES

The Data Encryption Standard is a block cipher that uses shared secret encryption selected by the National Bureau of Standards as an official Federal Information Processing Standard (FIPS) for the United States in 1976. It is based on a symmetric-key algorithm that uses a 56-bit key. The algorithm was initially controversial due to its classified design elements, having a relatively short key length, and suspicions about a National Security Agency (NSA) backdoor. DES consequently came under intense academic scrutiny which motivated the modern understanding of block ciphers and their cryptanalysis.

DES is however considered to be insecure for many applications because of its 56-bit key size being too small. In January, 1999, distributed.net and the Electronic Frontier Foundation joined to publicly break a DES key in 22 hours and 15 minutes. There are also some analytical results which demonstrate theoretical weaknesses in the cipher, although they are infeasible to mount in practice. The Triple DES algorithm is believed to be practically secure although there are theoretical attacks. In recent years, the cipher has been superseded by the Advanced Encryption Standard (AES). Furthermore, DES has been withdrawn as a standard by the National Institute of Standards and Technology (formerly the National Bureau of Standards).

A distinction is made between DES as a standard and DES the algorithm which is referred to as the DEA (the Data Encryption Algorithm).

AES

Advanced Encryption Standard (AES) is termed as specification for the encryption of electronic data being adopted by the U.S. government and is now being used worldwide. It supersedes DES. a symmetric-key algorithm is described by AES in which the same key is used for both encrypting and decrypting the data.

In the United States of America, AES was announced by National Institute of Standards and Technology (NIST) as U.S. FIPS PUB 197 (FIPS 197) on November 26, 2001 after a five-year standardization process in which fifteen competing designs were presented and evaluated before it was selected as the most suitable. It became effective as a Federal government standard on May 26, 2002 after approval by the Secretary of Commerce and available in many different encryption packages. AES is the first publicly accessible and open

cipher approved by the National Security Agency (NSA) for top secret information.

Originally called Rijndael, the cipher was developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, and submitted by them to the AES selection process. The name Rijndael is a play on the names of the two inventors.

Strictly speaking, AES is the name of the standard, and the algorithm described is a variant of Rijndael. However, in practice the algorithm is also referred to as "AES"

DSA

The Digital Signature Algorithm (DSA) is defined as a United States Federal Government standard or FIPS for digital signatures. It was declared by the National Institute of Standards and Technology (NIST) in August 1991 for Consumption in their Digital Signature Standard (DSS), specified in FIPS 186, adopted in 1993. A minor revision was issued in 1996 as FIPS 186-1. The standard was expanded further in 2000 as FIPS 186-2 and again in 2009 as FIPS 186-3.

DSA is covered by U.S. Patent 5,231,668, inJuly26, 1991, and attributed to David W. Kravitz, a former NSA employee. Thepatentwasgiven to "The United States of America as represented by the Secretary of Commerce, Washington, D.C." and the NIST has made this patent available worldwide royalty-free. Dr. Claus P. Schnorr claims that his U.S. Patent 4,995,082 (expired) covered DSA; this claim is disputed. DSA is a variant of the ElGamal Signature Scheme.

Check Your Progress 1

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) Write a note on need of secure application?

.....

.....

.....

.....

2) Write a note on encryption.

.....

.....

.....

.....

3) Write a note on cryptography?

.....

.....

.....

.....

4) Write a note on need of algorithm?

.....

.....

.....

.....

4.5 CONFIDENTIALITY

Privacy is an ethical principle associated with several professions (e.g., medicineSS, law). In ethics, and (in some places) in law and alternative forms of legal resolution such as mediation, some types of communication between a person and one of these professionals are "privileged" and should be kept hidden to third parties.

Confidentiality has also been defined by the International Organization for Standardization (ISO) in ISO-17799 as "ensuring that information is accessible only to those authorized to have access" and is one of the cornerstones of information security. Confidentiality is one of the design goals for many cryptosystems, made possible in practice by the techniques of modern cryptography.

Confidentiality of information, enforced in an adaptation of the military's classic "need to know" principle, forms the cornerstone of information security in today's corporations. The so called 'confidentiality bubble' restricts information flows, with both positive and negative consequences.

Legal Confidentiality

The work of Lawyers involve that they keep confidential anything pertaining to the representation of a client. The duty of confidentiality is much broader and important than the attorney-client evidentiary privilege, which only covers communications between the attorney and the client.

Both the privilege and the duty serve the purpose of encouraging and bring confidence in clients to speak frankly about their cases. This is the way, lawyers will be able to carry out their duty to provide clients with zealous representation. Otherwise, the opposing side may be able to surprise the lawyer in court with something which he did not know about his client, which may

weaken the client's position. Also, a distrustful client might hide a relevant fact which he thinks is incriminating, but which a skilled lawyer could turn to the client's advantage (for example, by raising affirmative defenses like self-defense).

However, most jurisdictions have exceptions for situations where the lawyer has reason to believe that the client may kill or seriously injure someone, may cause substantial injury to the financial interest or property of another, or is using (or seeking to use) the lawyer's services to perpetrate a crime or fraud.

In such situations the lawyer has the discretion, but not the obligation, to disclose information designed to prevent the planned action. Most states have a version of this discretionary disclosure rule under Rules of Professional Conduct, Rule 1.6 (or its equivalent).

A few jurisdictions have made this traditionally discretionary duty mandatory. For example, see the New Jersey and Virginia Rules of Professional Conduct, Rule 1.6.

In some jurisdictions the lawyer must try to convince the client to conform his or her conduct to the boundaries of the law before disclosing any otherwise confidential information.

Note that these exceptions generally do not cover crimes that have already occurred, even in extreme cases where murderers have confessed the location of missing bodies to their lawyers but the police are still looking for those bodies. The U.S. Supreme Court and many state supreme courts have affirmed the right of a lawyer to withhold information in such situations. Otherwise, it would be impossible for any criminal defendant to obtain a zealous defense.

California is famous for having one of the strongest duties of confidentiality in the world; its lawyers must protect client confidences at "every peril to himself or herself." Until an amendment in 2004, California lawyers were not even permitted to disclose that a client was about to commit murder.

Recent legislation in the UK curtails the confidentiality professionals like lawyers and accountants can maintain at the expense of the state. Accountants, for example, are required to disclose to the state any suspicions of fraudulent accounting and, even, the legitimate use of tax saving schemes if those schemes are not already known to the tax authorities.

History of the English law of Confidentiality

The modern English law of confidence stems from the judgment of the Lord Chancellor, Lord Cottenham, in which he restrained the defendant from publishing a catalogue of private etchings made by Queen Victoria and Prince Albert (Prince Albert v Strange).

However, the jurisprudential basis of confidentiality remained largely unexamined until the case of Saltman Engineering Co. Ltd. v Campbell Engineering Co. Ltd., in which the Court of Appeal upheld the existence of an equitable doctrine of confidence, independent of contract.

In Coco v A.N.Clark (Engineers) Ltd [1969] R.P.C. 41, Megarry J developed an influential tri-partite analysis of the essential ingredients of the cause of action for breach of confidence: the information must be confidential in quality, it must be imparted so as to import an obligation of confidence, and there must be an unauthorised use of that information to the detriment of the party communicating it.

The law in its then current state of development was authoritatively summarised by Lord Goff in the Spycatcher case. He identified three qualifications limiting the broad general principle that a duty of confidence arose when confidential information came to the knowledge of a person (the confidant) in circumstances where he had notice that the information was confidential, with the effect that it would be just in all the circumstances that he should be precluded from disclosing the information to others. First, once information had entered the public domain, it could no longer be protected as confidential. Secondly, the duty of confidence applied neither to useless information, nor to trivia. Thirdly, the public interest in the preservation of a confidence might be outweighed by a greater public interest favoring disclosure.

The incorporation into domestic law of Article 8 of the European Convention on Human Rights by the Human Rights Act 1998 has since had a profound effect on the development of the English law of confidentiality. Article 8 provides that everyone has the right to respect for his private and family life, his home and his correspondence. In Campbell v MGN Ltd, the House of Lords held that the Daily Mirror had breached Naomi Campbell's confidentiality rights by publishing reports and pictures of her attendance at Narcotics Anonymous meetings. Although their lordships were divided 3-2 as to the result of the appeal and adopted slightly different formulations of the applicable principles, there was broad agreement that, in confidentiality cases involving issues of privacy, the focus shifted from the nature of the relationship between claimant and defendant to (a) an examination of the nature of the information itself and (b) a balancing exercise between the claimant's rights under Article 8 and the defendant's competing rights (for example, under Article 10, to free speech).

It presently remains unclear to what extent and how this judge-led development of a partial law of privacy will impact on the equitable principles of confidentiality as traditionally understood.

Medical Confidentiality

Confidentiality is required in the conversations taking place in between doctors and patients. Legal protections prevent physicians from revealing certain discussions with patients, even under oath in court. This physician-patient privilege only applies to secrets shared between physician and patient during the course of providing medical care.

The rule dates back to at least the Hippocratic Oath, which reads: Whatever, in connection with my professional service, or not in connection with it, I see or hear, in the life of men, which ought not to be spoken of abroad, I will not divulge, as reckoning that all such should be kept secret.

Traditionally, medical ethics has viewed the duty of confidentiality as a relatively non-negotiable tenet of medical practice. More recently, critics like Jacob Appel have argued for a more nuanced approach to the duty that acknowledges the need for flexibility in many cases.

Confidentiality is mandated in America by HIPAA laws, specifically the Privacy Rule, and various state laws, some more rigorous than HIPAA. However, numerous exceptions to the rules have been carved out over the years. For example, many American states require physicians to report gunshot wounds to the police and impaired drivers to the Department of Motor Vehicles. Confidentiality is also challenged in cases involving the diagnosis of a sexually transmitted disease in a patient who refuses to reveal the diagnosis to a spouse, and in the termination of a pregnancy in an underage patient, without the knowledge of the patient's parents. Many states in the U.S. have laws governing parental notification in underage abortion.

Clinical and Counseling Psychology

The ethical principle of confidentiality requires that information shared by the client with the therapist in the course of treatment is not shared with others. This is important for the therapeutic alliance, as it promotes an environment of trust. There are important exceptions to confidentiality, namely where it conflicts with the clinician's duty to warn or duty to protect. This includes instances of suicidal behavior or homicidal plans, child abuse, elder abuse and dependent adult abuse.

Check Your Progress 2

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) Write a note on confidentiality.

.....

.....

.....

.....

2) Discuss legal confidentiality?

.....

.....

.....

.....

3) Discuss Medical Confidentiality?

.....

.....

.....

.....

4) What a note confidentiality in psychology.

.....

.....

.....

.....

4.6 DATA PROTECTION

The need for data protection legislation is to ensure that personal data is not processed without the knowledge and, except in certain cases, the consent of the data subject, to ensure that personal data which is processed is accurate, and to enforce a set of standards for the processing of such information.

Unlike the previous Act, the 1998 Act covers data held in manual files as well as computer files, although not all the provisions apply immediately to existing manual files. It is, however, University policy that as far as is possible all records will comply with the provisions of the Act.

Data subjects have the right to check the validity of the data held about them by Bangor University. By submitting a request in writing and paying the fee required the data subject may obtain a copy of data held about him/her.

The appropriate form for making a data subject access request can be found by clicking on Guide to Requesting Information from the University in the navigation bar above.

BU also accept initial contact with regard to subject access requests via e-mail which can be done here .However should you decide to submit your request by e-mail please supply a contact address so that we can contact you should we require further information. Please bear in mind that a fee is payable whether you submit your request by electronic means or by post and also bear in mind

that at present BU does not send out the information requested by electronic means.

Some kinds of personal data are exempt from the provisions of the Act, and there are some exceptions to the data subject's right of access.

Need of Data Protection

Data processing operations have developed with breathtaking speed over the past few years, expanding from very large mainframe operations to small business networks. In today world, most business organizations, whether large or small, cannot work without their electronic data; ushering in a new age of risk and vulnerability to their continued existence. Critical data loss is expensive, may involve illegal acts, and can seriously threaten the continued survivability of the unprepared business. At a minimum, recovery of lost data can be very expensive. Moreover, the recently enacted Sarbanes – Oxley law provides civil and criminal penalties for some companies if data supporting financial reporting cannot be recovered on a timely basis. Protection of critical and compliance data must be taken very seriously. Your job may depend upon it.

A large distributor of periodicals and newspapers operated via branch offices throughout the United States and Canada. The home office relied on wide area networks to receive critical data input from the field, but depended on each branch to protect its own locally produced data. One branch, located in a downtown office, was destroyed by a fire, which consumed the entire row of buildings in the block. Having faithfully backed up their critical data, the staff relied on a fireproof safe in an upstairs office for protection. It took seven days for the fire department to cool the ashes enough to permit a search for the safe. It was found in the basement cavity; and when opened the company found the plastic covering for their media had melted and the tapes were unreadable.

The State of Ohio backed up critical data from one of their networks to a removable storage device, but delegated responsibility for offsite storage to an IT intern. Sadly, one night the intern's car was broken into and the device stolen. It was soon discovered that the device contained personal data of a very large number of state employees. The state incurred a huge unbudgeted expense estimated to be in excess of $3,000,000 to implement identity theft and credit monitoring measures for the affected employees Careful advance planning can make all the difference. Soon after word of the 9/11 World Trade Center disaster began to spread, a data protection company contacted an affected client to offer assistance. As a result, the client declared a disaster and began to activate their disaster recovery plan. The data protection company delivered critical media to the client's recovery site located on Long Island in time for recovery operations to begin by 3:00 PM the same day. The company survived and has now fully recovered.

Past studies have shown there is a 100% certainty of losing critical data from every computer enterprise at least once in a five year period. For those businesses that follow "best practices guidelines" for protecting their data, recovery can be simple and easy; those who don't are very likely to cease operations within two years.

Why Am I at Risk

Electronic data is susceptible to loss or corruption unless protective measures are taken. Risk has many forms, and nothing short of a thorough understanding of internal and external threats will force IT professionals to evolve plans and design actions to protect their critical data. From its earliest days the industry has recognized the absolute necessity for strong measures designed to permit timely recovery from serious loss. To insure the survival of their critical data IT professionals have adopted a series of best practices to eliminate or mitigate the risk. All credible threats must be taken seriously.

Threats

Threats can originate from both internal and external sources, all of which must be taken care while developing a recovery plan. Internal threats include equipment failure, power surges, employee dishonesty, sabotage, fire, flooding and environmental contamination just to mention a few. The greatest threat, however, is employee ignorance, carelessness, user error, unintentional mistakes or improper operating procedures. External threats are so dangerous that they can affect a single building or an entire region. Acts of nature, (such as hurricanes, tornadoes, floods, blizzards, and earthquakes), frequently shut down major data processing operations when they occur. Other external threats such as power failure, communication failure, environmental contamination, urban riots, strikes, localized explosions and transportation disasters can also shut down computer operations.

Leading Causes of Data Loss Consequences

Data compiled by the Disaster Recovery Institute finds that the majority of businesses experiencing a serious loss of critical data will fail within two years. The reason is simple: They are not prepared to deal with the consequences. These include bad publicity, loss of customer confidence, loss of internal workflow, loss of sales capability, inability to process customer orders, loss of cash flow, and the extremely high cost of manual restoration of critical databases. In short, a business that is unable to quickly recover computer operations is at high risk for failure.

Risk Mitigation and Elimination

The diligent IT professional who understands and follows approved best practices guidelines will design programs and procedures that will cover all

credible threats. However, many small and medium businesses may not have access to such knowledge. Offsite data protection professionals can help you develop a plan that prepares you for a full and timely recovery. Their facilities and services are designed with these concerns in mind to provide you with the ultimate protection.

How Can an Offsite Data Protection Company Help

Your offsite data protection company can assist you with developing your disaster recovery plan and a backup strategy. They can work with you in determining your business requirements, recovery point objectives and backup methodologies. Recovery point objectives have been traditionally associated with restoring specific operations and data, but are increasingly being concerned with corporate governance and privacy legislation. All of these issues should be reviewed to determine your resource allocation in dollars and manpower, as well as offsite data storage requirements. Once your expectations are clearly defined, they can be converted into a Service Level Agreement (SLA) between you and your offsite data protection vendor to provide certain critical services. Basic data protection services include the following:

Controlled Access

By Controlling who has access to your critical data we can protect our data. Your data protection vendor will advise you in setting up protocols and levels of access for you and your staff. Thereafter, these will be strictly enforced, insuring that only authorized persons are able to gain access to your offsite media. Visitor access to the facilities is closely controlled, and is not permitted unless they are properly identified and have a clear reason for being there. No one other than your authorized staff and member employees are ever allowed direct access to your media.

Facility Security

Offsite data protection facilities operate in a high security, fire resistive environment. Security features include central station alarms, automatic fire suppression systems, video monitoring systems, zone controls and strict standard operating procedures. Security systems and operating procedures have been expanded to cover courier vehicles as well. Employees undergo background checks and periodic drug testing.

Environmental Controls

Media vaults are designed to provide a clean, climate controlled environment which will meet the standards established by media manufacturers. Temperature and humidity are maintained within established ranges. Vaults are kept dust and contaminant free.

Emergency Access

Emergencies happen, often at times which are very inconvenient. Your offsite data protection vendor offers 24 X 7 emergency access to your media. If your system crashes in the middle of the night or at half time of the Super Bowl, a phone call will get your backup media delivered to you quickly and securely.

4.7 EIGHT STEPS FOR INTEGRATING SECURITY INTO APPLICATION DEVELOPMENT

Computerworld - Most companies spend a tremendous amount of resources, time and money to protect their network perimeters from Internet-borne threats and hackers. But no matter how good a defense may be, it usually sometimes falls short in addressing the vulnerabilities inside the network at the application layer.

Recently it has been found out from a research that the application layer is one of the highest-risk areas and where the most potential damage can occur, either through insider targets or lack of protection. As a result, private company information can be exposed, resulting in harm to a company, its customers and its reputation.

While many variables affect Web application security, improving security in a few key areas can help eliminate vulnerabilities. It's important that security must be included in the initial Web design and not included after the application is developed. While some experts argue over where and when security integration and testing should be applied in the development life cycle, but there is no argument that it is essential ingredient. The software industry is making headway in this area, with some providers offering incentives to development teams to integrate security during the application development process.

Integrating security into the application development life cycle is not an all-or-nothing decision, but rather a process of negotiation within policy, risk and development requirements. Engaging security teams -- in-house or outsourced -- during the definition stage of application development determines the security areas necessary to satisfy policy and risk tolerance in the context of the organization. The areas are broken out in the remainder of this article.

1. Initial Review

The initial step is the initial review, which will allow the security team to study initial risks. The security team should work with the development team to gain an understanding of the following: The purpose of the application in the context of its users and its market

2. Definition phase: Threat modeling

Threat modeling is the practice of working and studying with developers to identify critical areas of applications dealing with critical sensitive information. The model is used to guide information flow and identify critical areas of the application's infrastructure that require added security attention.

Once the application is modeled and the critical areas and entry points are identified, security teams should work with the developers to create mitigation strategies for potential vulnerabilities. Threat modeling should be created early in the development life cycle of every project to achieve a secure foundation while using resources efficiently. This process should be followed throughout the development process as the application evolves in complexity.

3. Design phase: Design review

Application design reviews are an important step in identifying potential security risks at the early development stage. This step should be conducted by an independent and objective moderator who is separate from the development team. This process involves studying the application documents and interviewing developers and application owners. This will help keep the business purpose of the application in the forefront for analysis and later recommendations. Reviews are held at every stage of the development process. This includes the beginning of the design phase before code is written, the end of each software developmental phase throughout the life cycle, and, finally, before the application goes live.

4. Development phase: Code review

During this phase, the development and coding of the system takes place. As modules and phases are completed, and once unit testing for each is finished, security testing against units should be conducted throughout the development process. This includes testing units and reviewing code for best security practices. During this phase, the focus shifts to the hardware and network environment, ensuring that segments and trust relationships are appropriate, servers are hardened at the operating system level, and application software is configured and administered securely.

5. Deployment phase: Risk assessment

While security reviews have been conducted throughout the cycle, at this point, a risk assessment done prior to deployment is a step toward benchmarking the live application. Once risk has been benchmarked for the "go live" application, a strategy for mitigation of any risk can be put into place.

6. Risk mitigation

Risk mitigation involves reducing the risk by prioritizing, evaluating and implementing the controls that the security team identifies as necessary to mitigate vulnerabilities discovered during the risk-assessment stage. The least costly approach to implementing the most appropriate controls to reduce the risks to the organization is advisable. For example, risk can be assumed or reduced to an acceptable level, risk can be avoided by removing the cause, and risk can be transferred by using other options that compensate, such as purchasing insurance. The security team should work closely with the appropriate teams in the decision-making process on the most suitable mitigation options for each identified risk.

7. Benchmark

The next step is to benchmark the resulting application against industry standards to deliver a security scorecard. This allows executives to determine whether the security integration efforts are in line with industry averages and where there are gaps to improve.

Many phases can be benchmarked and will correspond to one or more of the security criteria relevant to the organization. These include: NIST SP 800-30 guidelines

Benchmarking for internal improvements is only one step. Doing security benchmarking against other similar programs within an organization's specific vertical industry is another measurement to consider.

8. Maintenance phase: Maintain

In order to maintain the strong security posture established, it's necessary to r employ a periodic security checks of all critical applications and controls. Securing an application is adequate for that moment in time, but new risks are introduced every day that could affect its security.

While network security is one layer of defense and protection, critical systems and sensitive information are still vulnerable to software application flaws, insider breaches and inadequate protection. With real-world testing across large enterprises and multiple industries, serious flaws are often found in most software, both custom and popular third-party applications. As such, it is critical for companies to integrate security into the application development life cycle to ensure applications are properly protected against external and internal threats.

fingerprint = AF19 FA27 2F94 998D FDB5 DE3D F8B5 06E4 A169 4E46

Check Your Progress 3

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) What do you understand by the term data protection.

.....

.....

.....

.....

2) Discuss the causes of data loss consequences.

.....

.....

.....

.....

3) What do you understand by the terms environmental control and emergency access?

.....

.....

.....

.....

4) Write any two steps of integrating security into application development.

.....

.....

.....

.....

4.8 LET US SUM UP

Security is really about risk management. Risk can be defined as the probability of events which may occur and produce negative consequences. Risk management is the process of evaluating or studying the risk and taking steps to decrease or mitigate risk to an acceptable level. This unit covers the topics like the need of secure application. The encryption techniques are also discussed. Type cryptography is also explained with its different types. The role of algorithm is also discussed. The topics like confidentiality and data protection are also covered well.

4.9 CHECK YOUR PROGRESS: THE KEY

Check Your Progress 1

1) Organizations consist of large amount information which are asset to the organization. And if a company wants to sustain in this competitive world it should intelligently share this information with its customers, business partners and employees. This asset should be well protected from threats which may lead to financial loss and other harm to the company. Examples of such loss can be disclosure of trade secrets, damaged reputation, decreased customer confidence, etc.

   The aim of computer security is to find ways to protect this asset by the selection and application of appropriate safeguards. Successful companies arrange a computer security strategy known as defense-in-depth, which is a layered approach that relies on people, operations and intelligent application of multiple techniques and technologies to achieve the desired level of information assurance by arranging the effective safeguards intelligently, companies are able to manage the risk factors by decreasing the vulnerabilities to threats, which results in less possibility of compromise and financial consequences.

2) Encryption is a process of changing information (plaintext) using an algorithm (cipher) to convert it into a form that is not readable to anyone except those possessing special knowledge , referred to as key. By following the above mentioned process we obtain a encrypted information (in cryptography referred to as ciphertext). In many contexts, the word encryption also implicitly refers to the reverse process, decryption (e.g. "software for encryption" can also perform decryption) to convert the encrypted piece of information into understandable format.

   Encryption is not new, it has been in use since long, it was used by military and government to facilitate secret communication. Encryption is now commonly used in protecting information within many kinds of civilian systems. For example, the Computer Security Institute reported that in 2007, 71% of companies surveyed utilized encryption for some of their data in transit, and 53% utilized encryption for some of their data in storage. Encryption can be used to prevent data from loss, data which is "at rest" files such as files onm computers and storage devices (e.g usb flash drive). In few recent years we have come across cases which involve theft of customers personal data through the theft of laptops or backup drops. Encryption of such helps them to be protected. Digital rights management systems which prevent unauthorized use or reproduction of copyrighted material and protect software against reverse engineering (see also copy

protection) are another somewhat different example of using encryption on data at rest.

3) Cryptography can be defined as the practice and study of hiding information. Modern cryptography passes through the disciplines of mathematics, computer science, and electrical engineering .some of the applications of cryptography are ATM cards, computer passwords and electronic commerce.

   The synonym of encryption, before the modern age was cryptology, the conversion of message from understandable form to non- understandable form.

   The sender maintain the ability to decrypt the information therefore avoid unwanted person from reading it. Since the computer computer came into existence the methods for cryptography are becoming increasingly complex.

   Modern cryptography is based on scientific approach, designs cryptographic algorithms around computational hardness assumptions, all this makes such algorithms hard to be broken by an adversary. Such systems are not unbreakable in theory but it is infeasible to do so by any practical means. Such schemes are secure. There exist information-theoretically secure schemes that provably cannot be broken—an example is the one-time pad--but these schemes are more difficult to implement than the theoretically breakable but computationally secure mechanisms. Cryptography have given rise to issues which are legal and some of them are still to be solved.

4) An algorithm can be stated in simple words as "a set of rules that precisely defines a sequence of operations." For some people, a program is only an algorithm if it stops eventually; for others, a program is only an algorithm if it stops before a given number of calculation steps.

   A prototypical example of an algorithm is Euclid's algorithm to determine the maximum common divisor of two integers; an example (there are others) is described by the flow chart above and as an example in a later section.

   Boolos & Jeffrey (1974, 1999) offer an informal meaning of the word in the following quotation:

   No human being can write fast enough, or long enough, or small enoughto list all members of an enumerably infinite set by writing out their names, one after another, in some notation. But humans can do something equally useful, in the case of certain enumerably infinite sets: They can give explicit instructions for determining the nth member of the set, for arbitrary

finite n. Such instructions are to be given quite explicitly, in a form in which they could be followed by a computing machine, or by a human who is capable of carrying out only very elementary operations on symbols.

Check Your Progress 2

1) Privacy is an ethical principle associated with several professions (e.g., medicineSS, law). In ethics, and (in some places) in law and alternative forms of legal resolution such as mediation, some types of communication between a person and one of these professionals are "privileged" and should be kept hidden to third parties.

   Confidentiality has also been defined by the International Organization for Standardization (ISO) in ISO-17799 as "ensuring that information is accessible only to those authorized to have access" and is one of the cornerstones of information security. Confidentiality is one of the design goals for many cryptosystems, made possible in practice by the techniques of modern cryptography.

   Confidentiality of information, enforced in an adaptation of the military's classic "need to know" principle, forms the cornerstone of information security in today's corporations. The so called 'confidentiality bubble' restricts information flows, with both positive and negative consequences.

2) The work of Lawyers involve that they keep confidential anything pertaining to the representation of a client. The duty of confidentiality is much broader and important than the attorney-client evidentiary privilege, which only covers communications between the attorney and the client.

   Both the privilege and the duty serve the purpose of encouraging and bring confidence in clients to speak frankly about their cases. This is the way, lawyers will be able to carry out their duty to provide clients with zealous representation. Otherwise, the opposing side may be able to surprise the lawyer in court with something which he did not know about his client, which may weaken the client's position. Also, a distrustful client might hide a relevant fact which he thinks is incriminating, but which a skilled lawyer could turn to the client's advantage (for example, by raising affirmative defenses like self-defense).

   However, most jurisdictions have exceptions for situations where the lawyer has reason to believe that the client may kill or seriously injure someone, may cause substantial injury to the financial interest or property of another, or is using (or seeking to use) the lawyer's services to perpetrate a crime or fraud.

3) Confidentiality is required in the conversations taking place in between doctors and patients. Legal protections prevent physicians from revealing

certain discussions with patients, even under oath in court. This physician-patient privilege only applies to secrets shared between physician and patient during the course of providing medical care.

The rule dates back to at least the Hippocratic Oath, which reads: Whatever, in connection with my professional service, or not in connection with it, I see or hear, in the life of men, which ought not to be spoken of abroad, I will not divulge, as reckoning that all such should be kept secret.

Traditionally, medical ethics has viewed the duty of confidentiality as a relatively non-negotiable tenet of medical practice. More recently, critics like Jacob Appel have argued for a more nuanced approach to the duty that acknowledges the need for flexibility in many cases.

4) The ethical principle of confidentiality requires that information shared by the client with the therapist in the course of treatment is not shared with others. This is important for the therapeutic alliance, as it promotes an environment of trust. There are important exceptions to confidentiality, namely where it conflicts with the clinician's duty to warn or duty to protect. This includes instances of suicidal behavior or homicidal plans, child abuse, elder abuse and dependent adult abuse.

Check Your Progress 3

1) The need for data protection legislation is to ensure that personal data is not processed without the knowledge and, except in certain cases, the consent of the data subject, to ensure that personal data which is processed is accurate, and to enforce a set of standards for the processing of such information.

Unlike the previous Act, the 1998 Act covers data held in manual files as well as computer files, although not all the provisions apply immediately to existing manual files. It is, however, University policy that as far as is possible all records will comply with the provisions of the Act.

Data subjects have the right to check the validity of the data held about them by Bangor University. By submitting a request in writing and paying the fee required the data subject may obtain a copy of data held about him/her.

The appropriate form for making a data subject access request can be found by clicking on Guide to Requesting Information from the University in the navigation bar above.

BU also accept initial contact with regard to subject access requests via e-mail which can be done here .However should you decide to submit your request by e-mail please supply a contact address so that we can contact you should we require further information. Please bear in mind that a fee is

payable whether you submit your request by electronic means or by post and also bear in mind that at present BU does not send out the information requested by electronic means.

2) Data compiled by the Disaster Recovery Institute finds that the majority of businesses experiencing a serious loss of critical data will fail within two years. The reason is simple: They are not prepared to deal with the consequences. These include bad publicity, loss of customer confidence, loss of internal workflow, loss of sales capability, inability to process customer orders, loss of cash flow, and the extremely high cost of manual restoration of critical databases. In short, a business that is unable to quickly recover computer operations is at high risk for failure.

3) **Environmental Controls**

Media vaults are designed to provide a clean, climate controlled environment which will meet the standards established by media manufacturers. Temperature and humidity are maintained within established ranges. Vaults are kept dust and contaminant free.

Emergency Access

Emergencies happen, often at times which are very inconvenient. Your offsite data protection vendor offers 24 X 7 emergency access to your media. If your system crashes in the middle of the night or at half time of the Super Bowl, a phone call will get your backup media delivered to you quickly and securely.

4) **Definition phase: Threat modeling**

Threat modeling is the practice of working and studying with developers to identify critical areas of applications dealing with critical sensitive information. The model is used to guide information flow and identify critical areas of the application's infrastructure that require added security attention.

Once the application is modeled and the critical areas and entry points are identified, security teams should work with the developers to create mitigation strategies for potential vulnerabilities. Threat modeling should be created early in the development life cycle of every project to achieve a secure foundation while using resources efficiently. This process should be followed throughout the development process as the application evolves in complexity.

Design phase: Design review

Application design reviews are an important step in identifying potential security risks at the early development stage. This step should be conducted

by an independent and objective moderator who is separate from the development team. This process involves studying the application documents and interviewing developers and application owners. This will help keep the business purpose of the application in the forefront for analysis and later recommendations. Reviews are held at every stage of the development process. This includes the beginning of the design phase before code is written, the end of each software developmental phase throughout the life cycle, and, finally, before the application goes live.

4. 10 SUGGESTED READINGS

- http://prismintl.org
- http://www.bangor.ac.uk1
- http://www.sans.org

MPDD-IGNOU/P.O.1T/Feb,2012

ISBN-978-81-266-5890-9