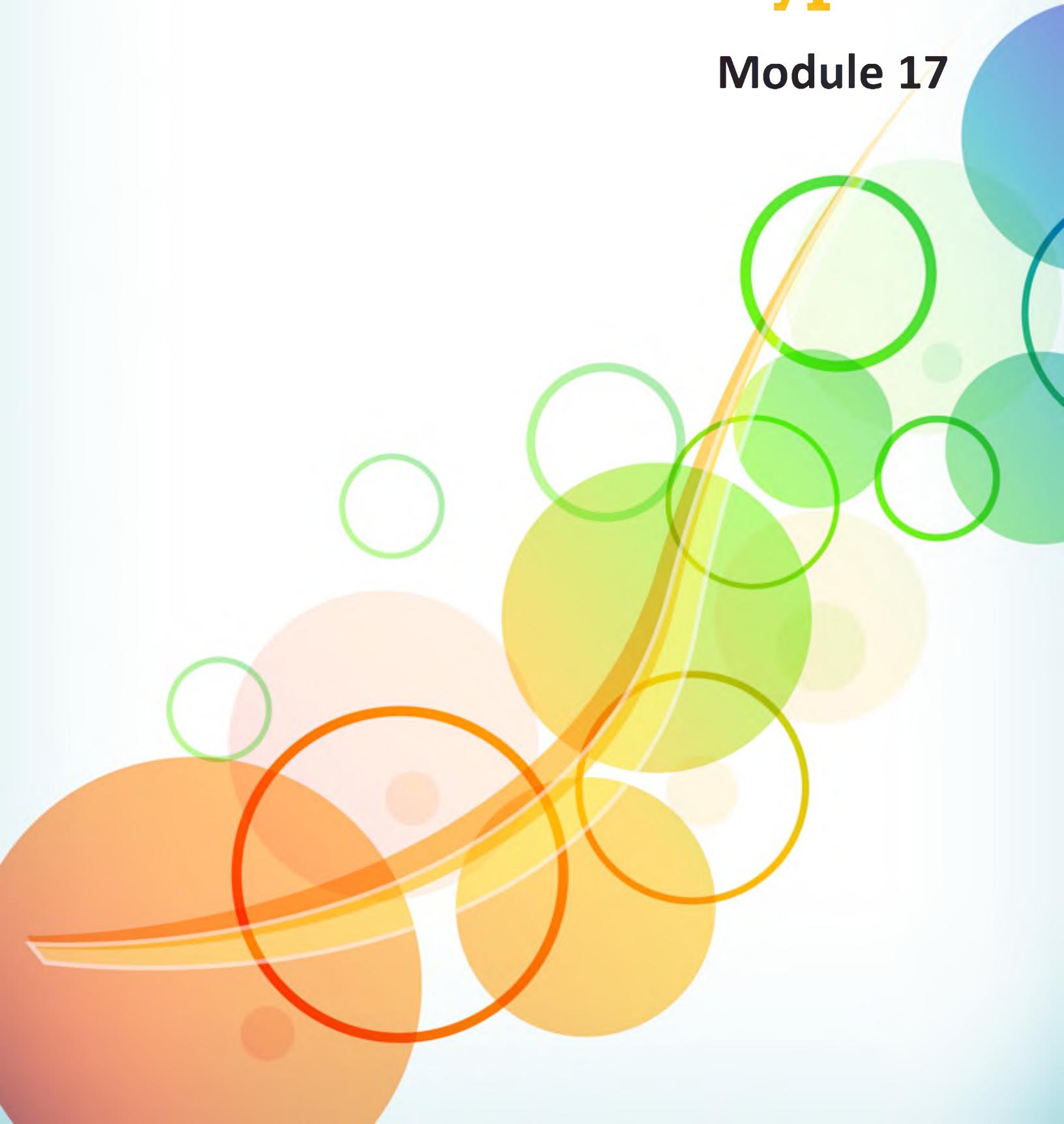


Evading IDS, Firewalls, and Honeypots

Module 17



Evading IDS, Firewalls, and Honeypots

Module 17

Engineered by Hackers. Presented by Professionals.



Ethical Hacking and Countermeasures v8

Module 17: Evading IDS, Firewalls, and Honeypots

Exam 312-50

The screenshot shows a news article from InformationWeek.com. The header reads "Security News" and features the CEH logo. The main headline is "Russian Service Rents Access To Hacked Corporate PCs". Below the headline is a sub-headline: "Service provides stolen remote desktop protocol credentials, letting buyers remotely log in to corporate servers and PCs, bypassing numerous security defenses." A small image of a bank building is shown. The text discusses how the service sells access credentials for some of the world's biggest enterprises, enabling buyers to bypass security defenses and log on to a server or PC located inside a corporate firewall. It also mentions a new report by Brian Krebs. The URL <http://www.informationweek.com> is at the bottom.



Security News

Russian Service Rents Access To Hacked Corporate PCs

Source: <http://www.informationweek.com>

Service provides stolen **remote desktop protocol** credentials, letting buyers remotely log in to corporate servers and PCs, bypassing numerous security defenses.

Want to infiltrate a business? An online service sells access credentials for some of the world's biggest enterprises, enabling buyers to bypass security defenses and remotely log on to a server or PC located inside a corporate firewall.

That finding comes by way of a new report from information security reporter Brian Krebs, who's discovered a **Russian-language** service that traffics in stolen Remote Desktop Protocol (RDP) credentials. RDP is a proprietary Microsoft standard that allows for a remote computer to be controlled via a **graphical user interface**.

The RDP-renting service, dubbed Dedicatexpress.com, uses the tagline "The whole world in one service" and is advertised on multiple underground cybercrime forums. It serves as an online marketplace, linking RDP-credential buyers and sellers, and it currently offers access to 17,000 PCs and servers worldwide.

Here's how **Dedicateexpress.com** works: Hackers submit their stolen RDP credentials to the service, which pays them a commission for every rental. According to a screen grab published by Krebs, the top submitters are "lopster," with **12,254 rentals**, followed by "_sz_", with **6,645 rentals**. Interestingly, submitters can restrict what the machines may be used for--for example, specifying that machines aren't to be used to run online gambling operations or PayPal scams, or that they can't be run with **administrator-level** credentials.

New users pay \$20 to join the site, after which they can search for available PC and server RDP credentials. Rental prices begin at just a few dollars and vary based on the machine's processor speed, upload and download bandwidth, and the length of time that the machine has been consistently available online.

According to Krebs, the site's managers have said they won't traffic in Russian RDP credentials, suggesting that the site's owners are based in Russia and don't wish to antagonize Russian authorities. According to security experts, Russian law enforcement agencies typically turn a blind eye to cybercrime gangs operating inside their borders, providing they don't target Russians, and that these gangs in fact occasionally assist authorities.

When reviewing the Dedicateexpress.com service, Krebs said he quickly discovered that access was being rented, for \$4.55, to a system that was listed in the Internet address space assigned to Cisco, and that several machines in the IP address range assigned to Microsoft's managed hosting network were also available for rent. In the case of Cisco, the RDP credentials--username and password--were both "Cisco." Krebs reported that a Cisco source told him the machine in question was a "bad lab machine."

As the Cisco case highlights, poor username and password combinations, combined with **remote-control applications**, give attackers easy access to corporate networks.

Still, even complex usernames and passwords may not stop attackers. Since Dedicateexpress.com was founded in 2010, it's offered access to about 300,000 different systems in total, according to Krebs. Interestingly, 2010 was the same year that security researchers first discovered the Georbot Trojan application, which scans PCs for signs that remote-control software has been installed and then captures and transmits related credentials to attackers. Earlier this year, security researchers at ESET found that when a **Georbot-infected PC** was unable to contact its designated command-and-control server to receive instructions or transmit stolen data, it instead contacted a server based in the country of Georgia.

When it comes to built-in remote access to Windows machines, RDP technology was first included in the **Windows XP Professional**--but not Home--version of the operating system, and it has been included in every edition of Windows released since then. The current software is dubbed Remote Desktop Services (for servers) and Remote Desktop Connection (for clients).

Might **Windows 8 security improvements** help prevent unauthorized people from logging onto PCs using stolen remote desktop protocol credentials? That's not likely, since Microsoft's new operating system--set to debut later this week--includes the latest version, Remote Desktop Protocol 8.0, built in.

Microsoft has also released a free Windows 8 Remote Desktop application, filed in the "productivity" section of Windows Store. According to Microsoft, "the new Metro-style Remote Desktop app enables you to conveniently access your PC and all of your corporate resources from anywhere."

"As many of you already know, a salient feature of Windows Server 2012 and Windows 8 is the ability to deliver a rich user experience for remote desktop users on corporate LAN and WAN networks," read a recent blog post from Shanmugam Kulandaivel, a senior program manager in Microsoft's Remote Desktop Virtualization team.

Despite such capabilities now being built into numerous operating systems--including **Linux** and **Mac OS X**--many security experts recommend deactivating or removing such tools when they're not needed. "Personally, I am a big fan of uninstalling unnecessary software, and it is always sound advice to minimize one's software footprint and related attack surface," said Wolfgang Kandek, CTO of Qualys. He made those comments earlier this year, after the source code for Symantec's pcAnywhere Windows remote-access software was leaked to the Internet by hacktivists. Security experts were concerned that attackers might discover an exploitable zero-day vulnerability in the remote-access code, which would allow them to remotely access any machine that had the software installed.



Copyright © 2012 UBM Tech

By Mathew J. Schwartz

<http://www.informationweek.com/security/attacks/russian-service-rents-access-to-hacked-c/240009580>

Module Objectives



- Ways to Detect an Intrusion
- Types of Intrusion Detection Systems
- General Indications of Intrusions
- Firewall Architecture
- Types of Firewall
- Firewall Identification
- How to Set Up a Honeypot
- Intrusion Detection Tools
- How Snort Works



- Firewalls
- Honeypot Tools
- Evading IDSes
- Evading Firewalls
- Detecting Honeypots
- Firewall Evasion Tools
- Packet Fragment Generators
- Countermeasures
- Firewall/IDS Penetration Testing



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

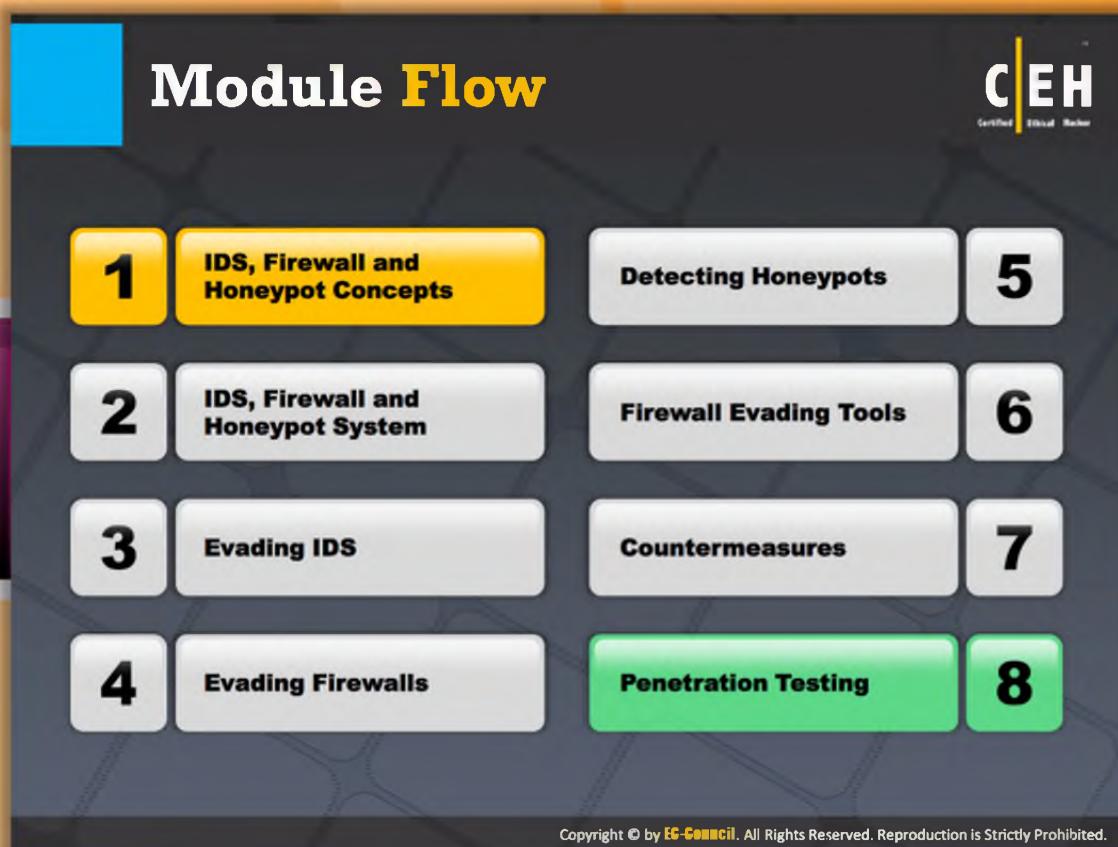


Module Objectives

Today, hacking and computer system attacks are common, making the importance of intrusion detection and active protection all the more relevant. Intrusion detection systems (IDSes), intrusion prevention systems (IPSeS), firewalls, and honeypots are the security mechanisms implemented to secure networks or systems. But attackers are able to manage even these security mechanisms and trying to break into the legitimate system or network with the help of various evasion techniques.

This module will familiarize you with:

- ❑ Ways to Detect an Intrusion
- ❑ Types of Intrusion Detection Systems
- ❑ General Indications of Intrusions
- ❑ Firewall Architecture
- ❑ Types of Firewalls
- ❑ Firewall Identification
- ❑ How to Set Up a Honeypot
- ❑ Intrusion Detection Tools
- ❑ How Snort Works
- ❑ Firewalls
- ❑ Honeypot Tools
- ❑ Evading IDSes
- ❑ Evading Firewalls
- ❑ Detecting Honeypots
- ❑ Firewall Evasion Tools
- ❑ Packet Fragment Generators
- ❑ Countermeasures
- ❑ Firewall/IDS Penetration Testing

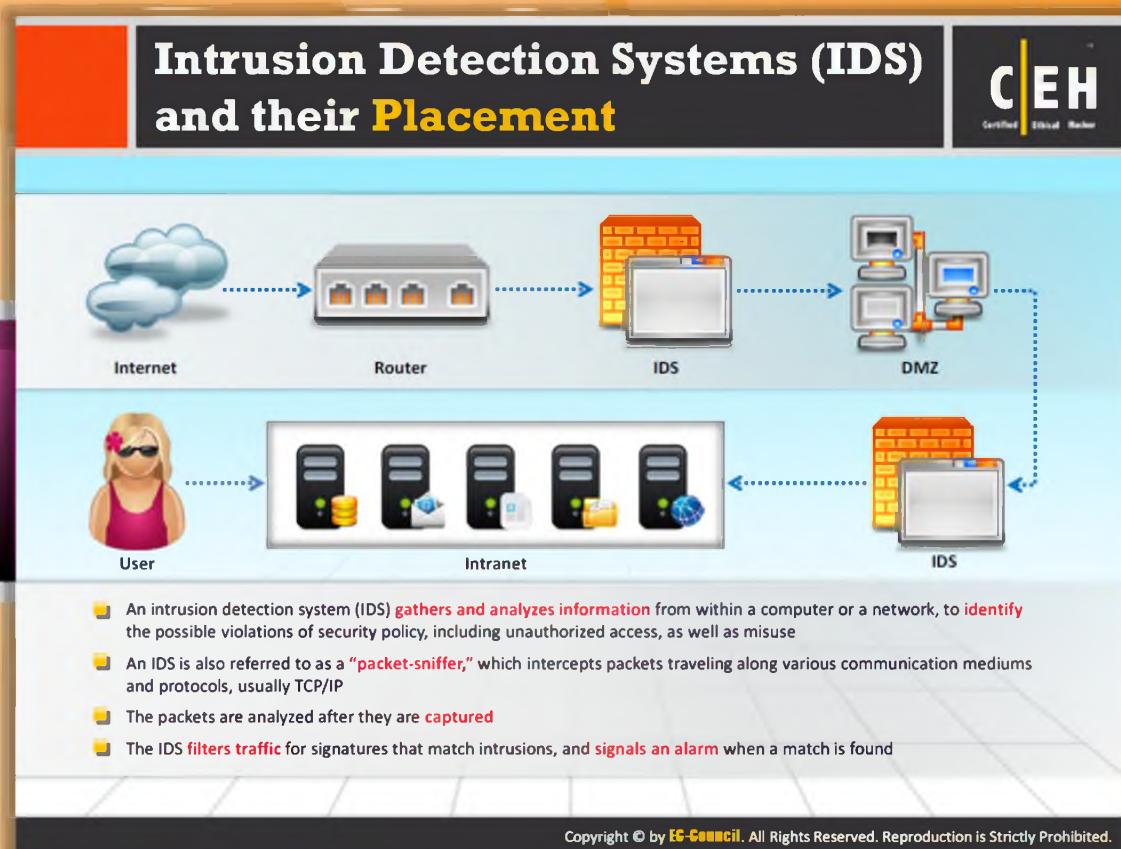


Module Flow

To understand IDSe, firewalls, and honeypots, evasion techniques used by the attackers to break into the target network or system, it is necessary to understand these mechanisms and how they prevent intrusions and offer protection. So, let us begin with basic IDS, firewall, and honeypot concepts.

	IDS, Firewall and Honeypot Concepts		Detecting Honeypots
	IDS, Firewall and Honeypot System		Firewall Evading Tools
	Evading IDS		Countermeasure
	Evading Firewall		Penetration Testing

This section introduces you with the basic IDS, firewall, and honeypot concepts.



Intrusion Detection Systems (IDSe) and their Placement

An intrusion detection system is used to **monitor** and **protect networks** or systems for malicious activities. To alert security personnel about intrusions, intrusion detection systems are highly useful. IDSe are used to monitor network traffic. An IDS checks for **suspicious activities**. It notifies the administrator about intrusions immediately.

- An intrusion detection system (IDS) gathers and analyzes information from within a computer or a network, to identify the possible violations of security policy, including **unauthorized access**, as well as misuse
- An IDS is also referred to as a "**packet-sniffer**," which intercepts packets traveling along various communication mediums and protocols, usually TCP/IP
- The packets are analyzed after they are captured
- An IDS evaluates a suspected intrusion once it has taken place and signals an alarm

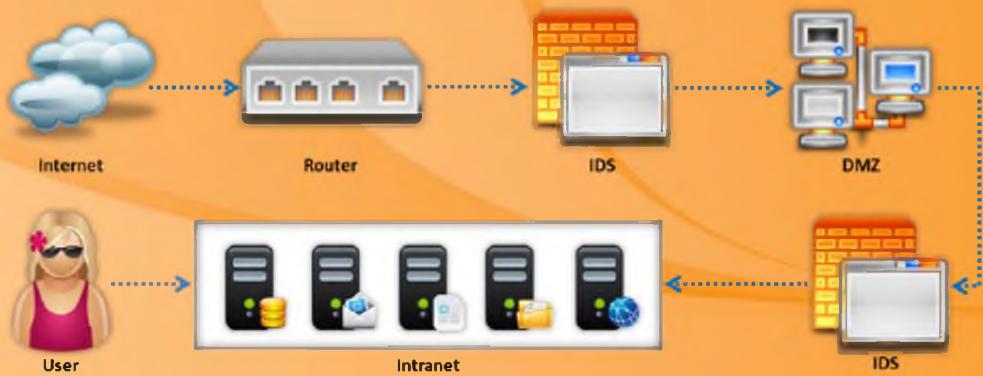
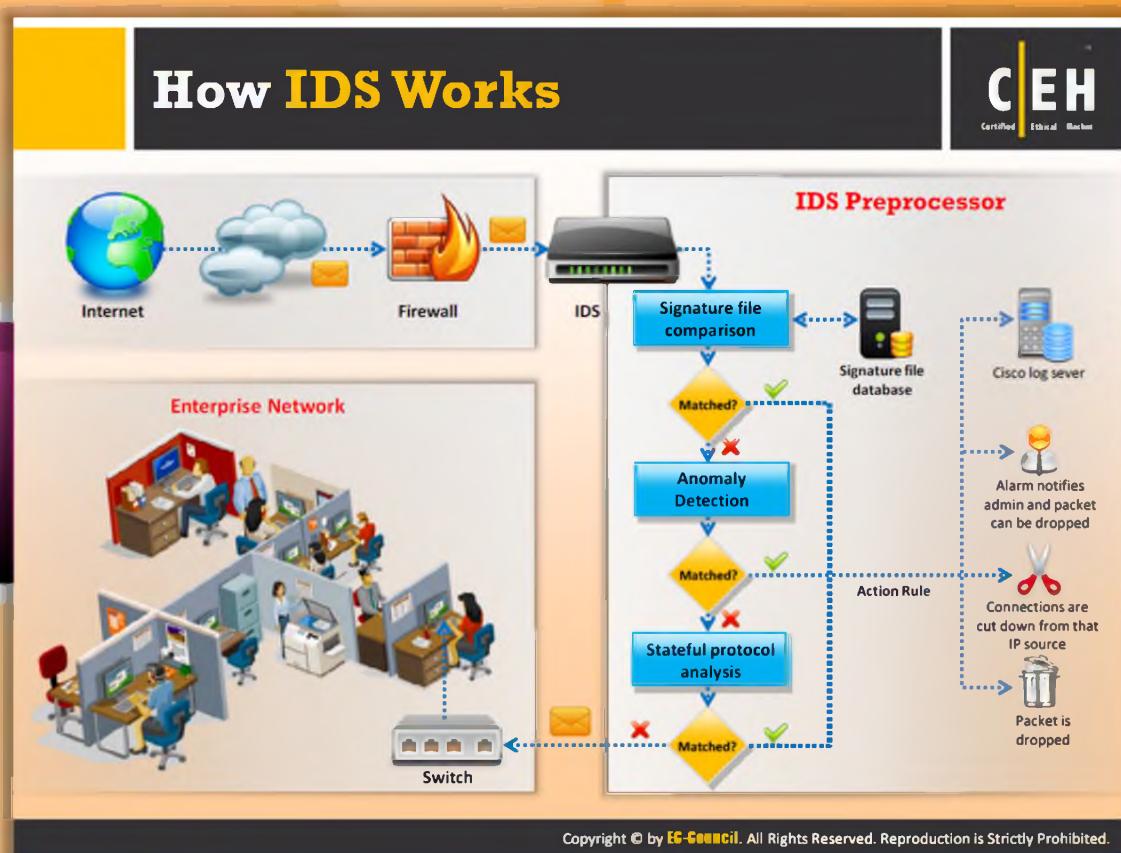


FIGURE 17.1: Intrusion Detection Systems (IDSe) and their Placement



How an IDS Works

The main purposes of IDSes are that they not only **prevent intrusions** but also alert the **administrator immediately** when the attack is still going on. The administrator could identify methods and techniques being used by the intruder and also the source of attack.

An IDS works in the following way:

- IDSes have sensors to **detect signatures** and some advanced IDSes have behavioral activity detection to determine malicious behavior. Even if signatures don't match this activity detection system can alert administrators about possible attacks.
- If the signature matches, then it moves to the next step or the **connections are cut down** from that IP source, the packet is dropped, and the alarm notifies the admin and the packet can be dropped.
- Once the signature is matched, then sensors pass on **anomaly detection**, whether the received packet or request matches or not.
- If the packet passes the anomaly stage, then stateful protocol analysis is done. After that through switch the packets are passed on to the network. If anything mismatches again, the connections are cut down from that **IP source**, the packet is dropped, and the alarm notifies the admin and packet can be dropped.

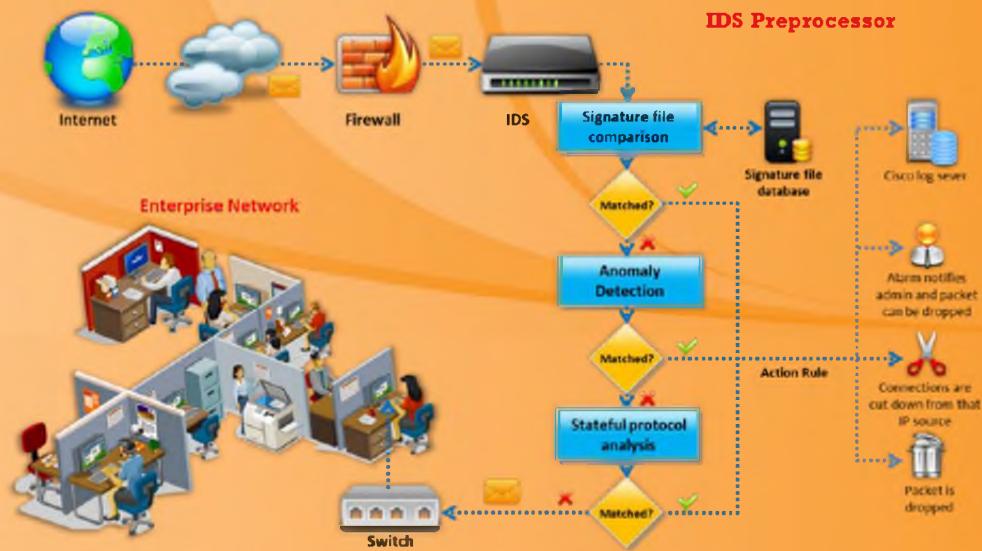


FIGURE 17.2: How an IDS Works

Ways to Detect an Intrusion

C|EH
Certified Ethical Hacker

Signature Recognition

It is also known as misuse detection. Signature recognition tries to **identify events** that misuse a system



Anomaly Detection



It detects the **intrusion based** on the fixed behavioral characteristics of the users and components in a computer system

Protocol Anomaly Detection



In this type of detection, models are built to explore **anomalies** in the way vendors deploy the **TCP/IP specification**

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Ways to Detect an Intrusion

An intrusion is detected in three ways.



Signature Detection

Signature recognition is also known as **misuse detection**. It tries to identify events that indicate an abuse of a system. It is achieved by creating models of intrusions. Incoming events are compared with intrusion models to make a detection decision. While creating signatures, the model must detect an attack without disturbing the normal traffic on the system. Attacks, and only attacks, should match the model or else false alarms can be generated.

- The simplest form of signature recognition uses simple pattern matching to compare the network packets against binary signatures of known attacks. A binary signature may be defined for a specific portion of the packet, such as the **TCP flags**.
- Signature recognition can detect known **attacks**. However, there is a possibility that other packets that match might represent the signature, triggering bogus signals. Signatures can be customized so that even well-informed users can create them.
- Signatures that are formed improperly may trigger **bogus signals**. In order to detect misuse, the number of signatures required is huge. The more the signatures, the more

attacks can be detected, though traffic may incorrectly match with the signatures, reducing the performance of the system.

- ➊ The bandwidth of the network is consumed with the increase in the signature database. As the signatures are compared against those in the database, there is a probability that the maximum number of comparisons cannot be made, resulting in certain packets being dropped.
- ➋ New virus attacks such as **ADMutate** and **Nimda** create the need for multiple signatures for a single attack. Changing a single bit in some attack strings can invalidate a signature and create the need for an entirely new signature.
- ➌ Despite problems with signature-based intrusion detection, such systems are popular and work well when configured correctly and monitored closely



Anomaly Detection

Anomaly detection is otherwise called “**not-use detection**.” Anomaly detection differs from the signature recognition model. The model consists of a database of anomalies. Any event that is identified with the **database** is considered an anomaly. Any deviation from normal use is **labeled an attack**. Creating a model of normal use is the most difficult task in creating an anomaly detector.

- ➊ In the traditional method of anomaly detection, important data is kept for checking variations in network traffic for the model. However, in reality, there is less variation in **network traffic** and too many statistical variations making these models imprecise; some events labeled as anomalies might only be irregularities in network usage.
- ➋ In this type of approach, the inability to instruct a model thoroughly on the normal network is of grave concern. These models should be trained on the specific network that is to be policed.



Protocol Anomaly Detection

Protocol anomaly detection is based on the anomalies specific to a protocol. This model is integrated into the **IDS model** recently. It identifies the **TCP/IP protocol** specific flaws in the network. Protocols are created with specifications, known as RFCs, for dictating proper use and communication. The protocol anomaly detector can identify new attacks.

- ➊ There are new attack methods and exploits that violate protocol standards being discovered frequently.
- ➋ The pace at which the **malicious signature attacker** is growing is incredibly fast. But the network protocol, in comparison, is well defined and changing slowly. Therefore, the signature database must be updated frequently to detect attacks.
- ➌ Protocol anomaly detection systems are easier to use because they require no signature updates

- ➊ Protocol anomaly detectors are different from the traditional IDS in how they present alarms.
- ➋ The best way to present alarms is to explain which part of the state system was compromised. For this, the IDS operators have to have a thorough knowledge of the protocol design; the best way is the documentation provided by the IDS.

Types of Intrusion Detection Systems

C|EH
Certified Ethical Hacker

Network-Based Intrusion Detection

- These mechanisms typically consist of a **black box** that is placed on the network in the promiscuous mode, listening for patterns indicative of an intrusion



Log File Monitoring

- These mechanisms are typically programs that **parse log files** after an event has already occurred, such as failed log in attempts



Host-Based Intrusion Detection

- These mechanisms usually include auditing for events that occur on a **specific host**
- These are not as common, due to the overhead they incur by having to monitor each system event



File Integrity Checking

- These mechanisms check for **Trojan horses**, or files that have otherwise been modified, indicating an intruder has already been there, for example, Tripwire



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Types of Intrusion Detection Systems

Basically there are four types of intrusion detection systems available. They are:



Network-based Intrusion Detection

The NIDS checks every packet entering the network for the presence of **anomalies** and **incorrect data**. Unlike the firewalls that are confined to the filtering of data packets with vivid malicious content, the NIDS checks every packet thoroughly. An **NIDS captures** and inspects all traffic, regardless of whether it is permitted. Based on the content, at either the IP or application-level, an alert is generated. Network-based intrusion detection systems tend to be more distributed than **host-based IDSe**s. The NIDS is basically designed to identify the anomalies at the router- and host-level. The NIDS audits the information contained in the data packets, logging information of malicious packets. A threat level is assigned to each risk after the data packets are received. The threat level enables the security team to be on alert. These mechanisms typically consist of a **black box** that is placed on the network in the promiscuous mode, listening for patterns indicative of an intrusion.



Host-based Intrusion Detection

In the host-based system, the **IDS analyzes** each system's behavior. The HIDS can be installed on any system ranging from a desktop PC to a server. The HIDS is more versatile than

the NIDS. One example of a host-based system is a program that operates on a system and receives application or operating system audit logs. These programs are highly effective for detecting insider abuses. Residing on the trusted network systems themselves, they are close to the network's authenticated users. If one of these users attempts unauthorized activity, host-based systems usually detect and collect the most pertinent information promptly. In addition to detecting unauthorized insider activity, host-based systems are also effective at detecting unauthorized file modification. HIDSes are more focused on changing aspects of the local systems. HIDS is also more platform-centric, with more focus on the Windows OS, but there are other **HIDSes** for **UNIX platforms**. These mechanisms usually include auditing for events that occur on a specific host. These are not as common, due to the overhead they incur by having to monitor each system event



Log File Monitoring

A Log File Monitor (LFM) monitors log files created by network services. The **LFT IDS** searches through the logs and identifies malicious events. In a similar manner to NIDS, these systems look for patterns in the log files that suggest an intrusion. A typical example would be parsers for **HTTP server** log files that look for intruders who try well-known security holes, such as the “**phf**” attack. An example is swatch. These mechanisms are typically programs that parse log files after an event has already occurred, such as failed log in attempts.



File Integrity Checking

These mechanisms check for Trojan horses, or files that have otherwise been modified, indicating an intruder has already been there, for example, Tripwire.

System Integrity Verifiers (SIV)

Tripwire is a **System Integrity Verifiers (SIV)** that monitors system files and detects changes by an intruder

A: Node tree view showing categories like Root Node Group, By Location, and By Service.

B: List of monitored system files with details such as Element, Change Type, Current Value, and Severity.

http://www.tripwire.com

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



System Integrity Verifiers (SIV)

Source: <http://www.tripwire.com>

A System Integrity Verifier (SIV) **monitors system files** to determine whether an intruder has changed the files. An integrity monitor watches key system objects for changes. For example, a basic integrity monitor uses system files, or registry keys, to track changes by an intruder. Although they have limited functionality, integrity monitors can add an additional layer of protection to other forms of **intrusion detection**.

A: Node tree view showing categories like Root Node Group, By Location, and By Service.

B: List of monitored system files with details such as Element, Change Type, Current Value, and Severity.

FIGURE 17.3: System Integrity Verifiers (SIV) Screenshot

General Indications of Intrusions



File System Intrusions

- The presence of new, unfamiliar files, or programs
- Changes in file permissions
- Unexplained changes in a file's size
- Rogue files on the system that do not correspond to your master list of signed files
- Unfamiliar file names in directories
- Missing files

Network Intrusions

- Repeated probes of the available services on your machines
- Connections from unusual locations
- Repeated login attempts from remote hosts
- Arbitrary data in log files, indicating attempts to cause a DoS or to crash a service

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



General Indications of Intrusions

Following are the general indications of intrusions:



File System Intrusions

By observing the system files, you can identify the presence of an intruder. The system files record the activities of the system. Any modification or deletion in the file attributes or the file itself is a sign that the system was a target of attack:

- If you find new, **unknown files/programs** on your system, then there is a possibility that your system has been intruded. The system can be compromised to the point that it can in turn **compromise other systems** in your network.
- When an intruder gains access to a system, he or she tries to escalate privileges to gain administrative access. When the intruder obtains the Administrator privilege, he or she changes the file permissions, for example, from **Read-Only to Write**.
- Unexplained modifications in file size are also an indication of an attack. Make sure you analyze all of your system files.
- Presence of rogue suid and sgid files on your Linux system that do not match your master list of suid and sgid files could indicate an attack.

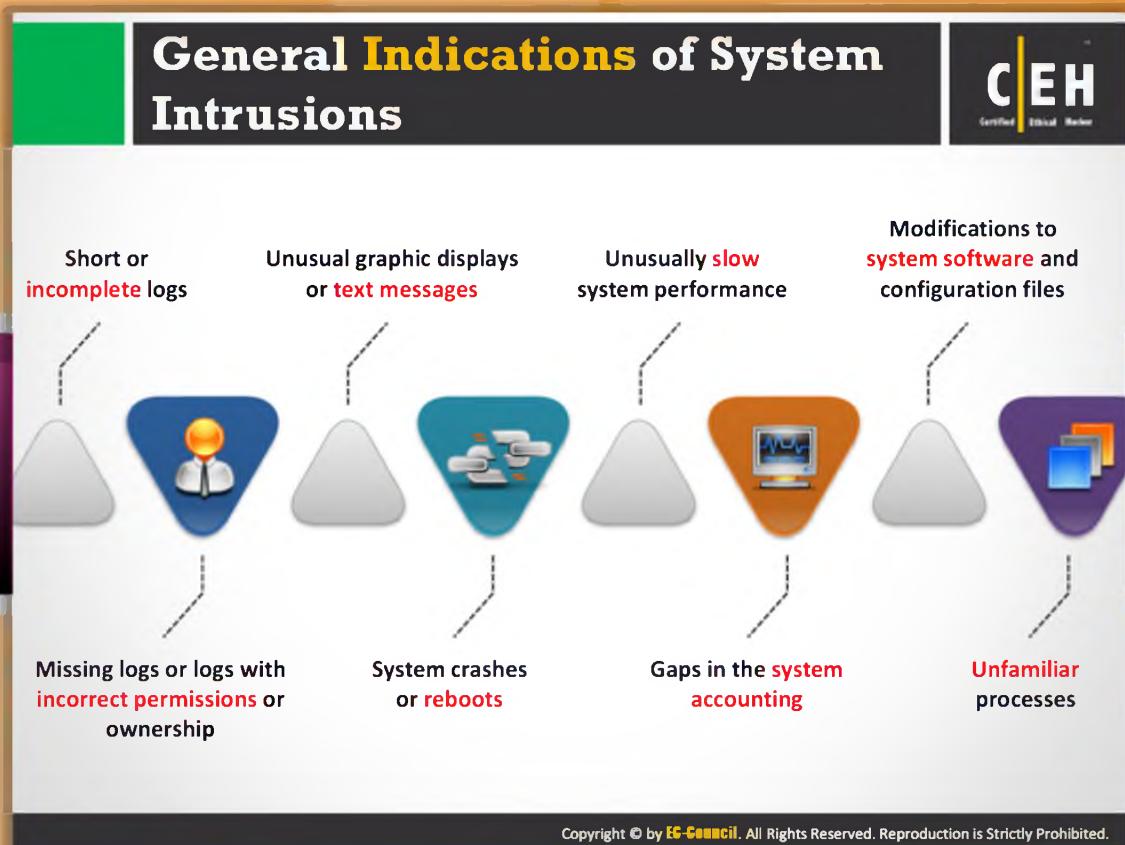
- ⌚ You can identify unfamiliar file names in directories, including executable files with strange extensions and double extensions.
- ⌚ Missing files are also sign of a probable intrusion/attack.



Network Intrusions

General indications of network intrusions:

- ⌚ Sudden increase in bandwidth consumption is an indication of intrusion.
- ⌚ Repeated probes of the available services on your machines.
- ⌚ Connection requests from IPs other than those in the network range are an indication that an **unauthenticated user (intruder)** is attempting to connect to the network.
- ⌚ You can identify repeated attempts to log in from remote machines.
- ⌚ Arbitrary log data in log files indicates attempts of **denial-of-service attacks**, bandwidth consumption, and distributed denial-of-service attacks.



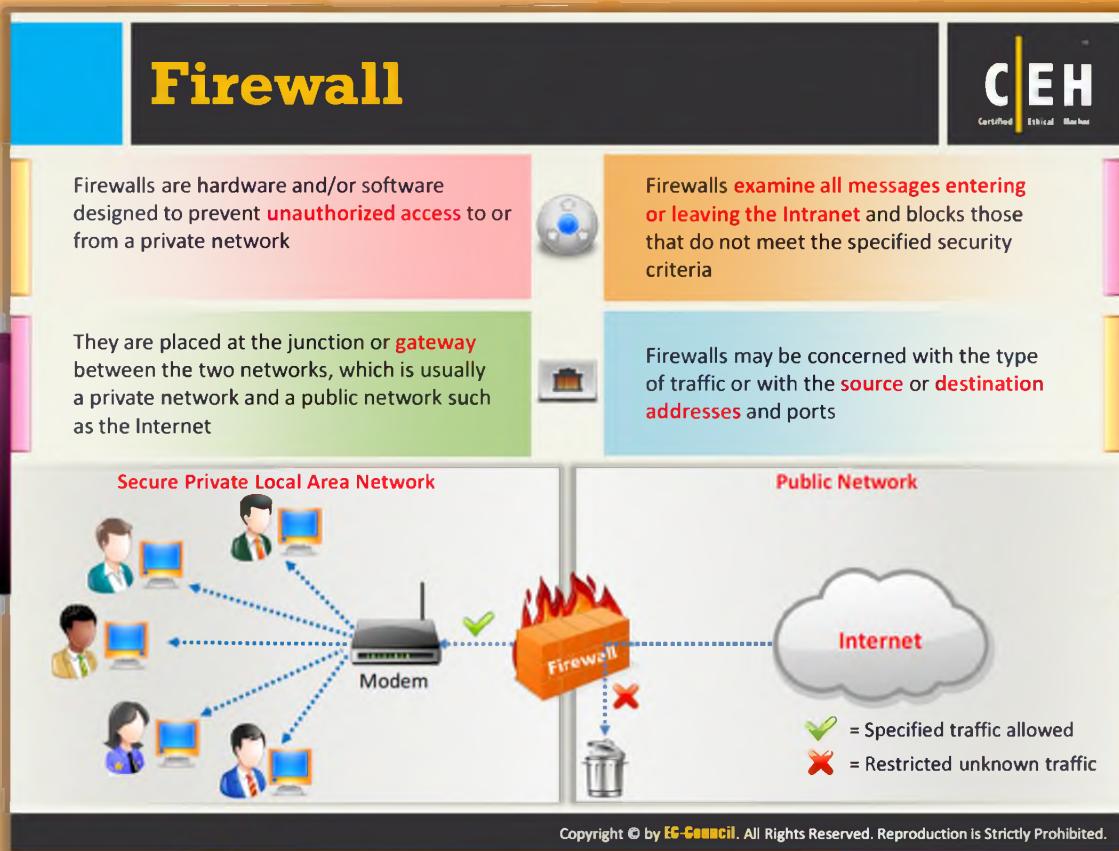
General Indications of System Intrusions

To check whether the system is **attacked**, you need to check certain parameters that clearly indicate the presence of an intruder on the system. When an intruder attempts to break into the system, he or she attempts to hide his or her presence by modifying certain **system files** and **configurations** that indicate intrusion.

Certain signs of intrusion include:

- ⌚ System's failure in identifying valid user
- ⌚ Active access to unused logins
- ⌚ Logins during non-working hours
- ⌚ New user accounts other than the accounts created
- ⌚ Modifications to system software and configuration files using Administrator access and the presence of hidden files
- ⌚ Gaps in system audit files, which indicate that the system was idle for that particular time; these gaps actually indicate that the intruder has attempted to erase the audit tracks
- ⌚ The system's performance decreases drastically, consuming CPU time
- ⌚ System crashes suddenly and reboots without user intervention

- ⌚ The system logs are too short and incomplete
- ⌚ Timestamps of system logs are modified to include strange inputs
- ⌚ Permissions on the logs are changed, including the ownership of the logs
- ⌚ System logs are deleted
- ⌚ Systems performance is abnormal, the system responds in unfamiliar ways
- ⌚ Unknown processes are identified on the system
- ⌚ Unusual display of graphics, pop-ups, and text messages observed on the system



Firewalls

A firewall is a set of related programs located at the **network gateway** server that protects the resources of a private network from users on other networks. Firewalls are a set of tools that monitor the flow of traffic between networks. A firewall, placed at the network level and working closely with a router, filters all network packets to determine whether or not to forward them toward their destinations. A firewall is often installed away from the rest of the network so that no incoming request can get directly to a private network resource. If configured properly, systems on one side of the firewall are protected from systems on the other side of the firewall.

- ➊ A firewall is an **intrusion detection mechanism**. Firewalls are specific to an organization's security policy. The settings of the firewalls can be changed to make appropriate changes to the firewall functionality.
- ➋ Firewalls can be configured to restrict incoming traffic to **POP** and **SNMP** and to enable email access. Certain firewalls block the email services to secure against spam.
- ➌ Firewalls can be configured to check inbound traffic at a point called the "**choke point**," where security audit is performed. The firewall can also act as an active "phone tap" tool in identifying the intruder's attempt to dial into the modems within the network

that is secured by firewall. The firewall logs consist of logging information that reports to the administrator on all the attempts of various incoming services.

- ➊ The firewall verifies the incoming and outgoing traffic against **firewall rules**. It acts as a router to move data between networks. Firewalls manage access of private networks to host applications.
- ➋ All the attempts to log in to the network are identified for auditing. Unauthorized attempts can be identified by embedding an alarm that is triggered when an unauthorized user attempts to login. Firewalls can filter packets based on address and types of traffic. They identify the source, destination addresses, and port numbers while address filtering, and they identify types of network traffic when protocol filtering. Firewalls can identify the state and attributes of the data packets.



FIGURE 17.4: Working of Firewall

Firewall Architecture

C|EH
Certified Ethical Hacker

Bastion Host:

- Bastion host is a computer system designed and configured to protect **network resources** from attack
- Traffic entering or leaving the network passes through the firewall, it has two interfaces:
 - **public interface** directly connected to the Internet
 - **private interface** connected to the Intranet

Screened Subnet:

- The screened subnet or DMZ (additional zone) contains **hosts** that offer public services
- The DMZ zone **responds to public requests**, and has no hosts accessed by the private network
- Private zone can not be accessed by **Internet users**

Multi-homed Firewall:

- In this case, a firewall with three or more interfaces is present that allows for further subdividing the systems based on the **specific security objectives** of the organization

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Firewall Architecture

Firewall architecture consists of the following elements:



Bastion host

The bastion host is designed for the purpose of **defending against attacks**. It acts as a mediator between inside and outside networks. A bastion host is a computer system designed and configured to protect **network resources from attack**.

Traffic entering or leaving the network passes through the firewall, it has two interfaces:

- Public interface directly connected to the Internet
- Private interface connected to the intranet



FIGURE 17.5: Bastion Host Architecture



Screened subnet

A screened subnet is a network architecture that uses a **single firewall** with three network interfaces. The first interface is used to connect the Internet, the second interface is used to connect the **DMZ**, the third interface is used to connect the intranet.

The main advantage with the screened subnet is it separates the DMZ and Internet from the intranet so that when the firewall is compromised access to the intranet won't be possible.

- The screened subnet or **DMZ (additional zone)** contains hosts that offer public services
- Public zone is directly connected to the Internet and has no hosts controlled by the organization
- Private zone has systems that Internet users have no business accessing



FIGURE 17.6: Screened Subnet Architecture



Multi-homed firewall

A multi-homed firewall generally refers to **two or more networks**. Each interface is connected to the **separate network segments** logically and physically. A multi-homed firewall is used to increase efficiency and reliability of an IP network. In this case, more than three interfaces are present that allow for further subdividing the systems based on the specific security objectives of the organization.



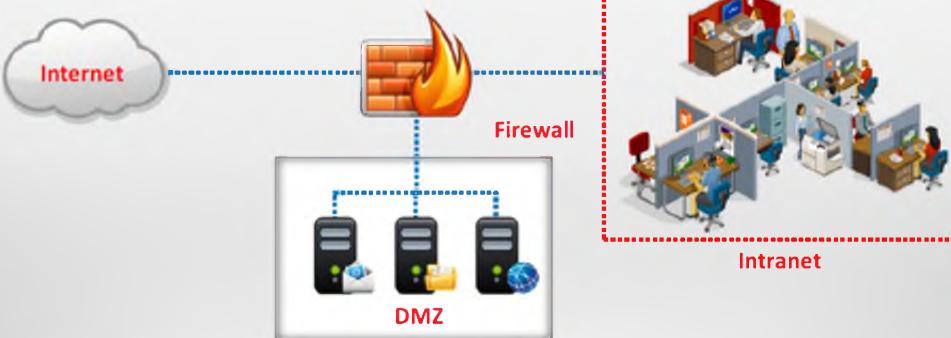
FIGURE 17.7: Multi-Homed Firewall Architecture

DeMilitarized Zone (DMZ)

C|EH
Certified Ethical Hacker

 DMZ is a network that **serves as a buffer** between the internal secure network and **insecure Internet**

 It can be created using **firewall with three or more network interfaces** assigned with specific roles such as Internal trusted network, DMZ network, and external un-trusted network (Internet)



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



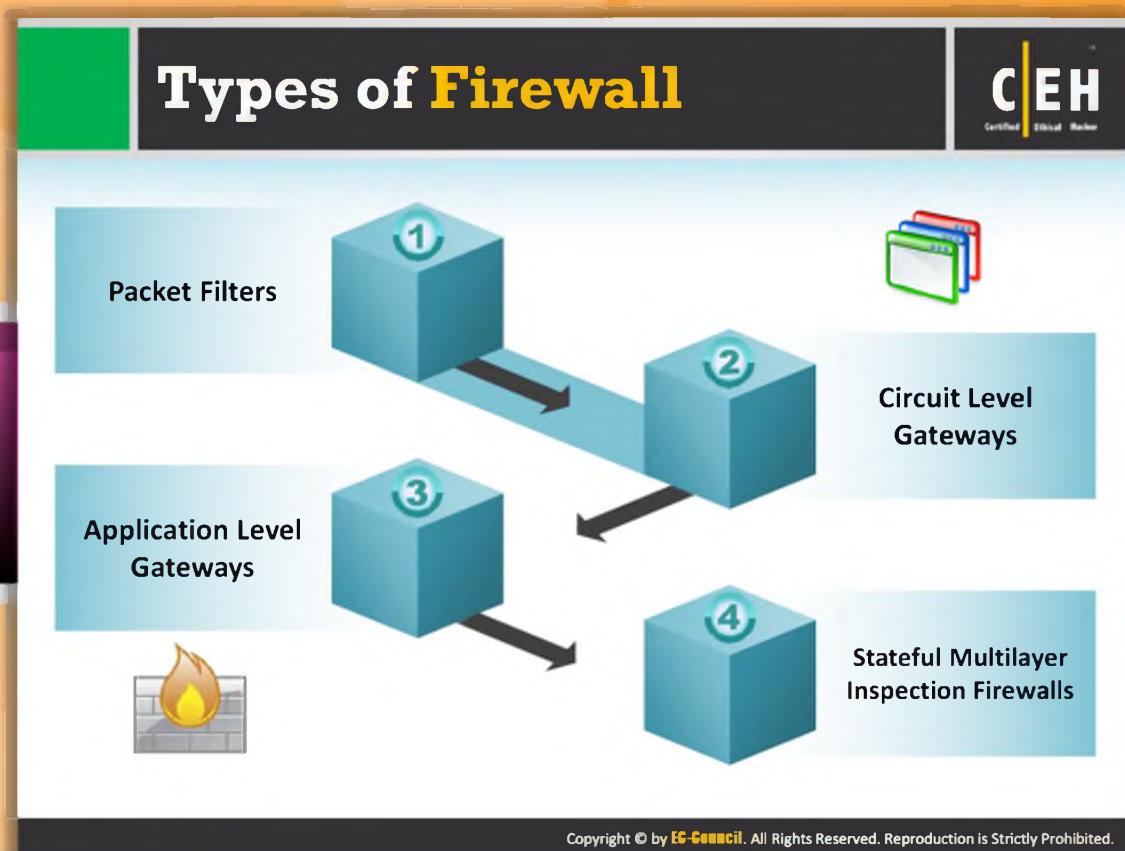
Demilitarized Zone (DMZ)

The DMZ is a **host computer** or a **network** placed as a neutral network between a particular firm's internal, or private, network and outside, or public, network to prevent the outside user from accessing the company's private data. DMZ is a network that serves as a buffer between the **internal secure network** and **insecure internet**.

It is created using a firewall with three or more network interfaces assigned with specific roles such as Internal trusted network, DMZ network, and External un-trusted network (Internet).



FIGURE 17.8: Demilitarized Zone (DMZ)



Types of Firewalls

A firewall refers to a **hardware device** or a **software program** used in a system to prevent malicious information from passing through and allowing only the approved information.

Firewalls are mainly categorized into four types:

- ⌚ Packet filters
- ⌚ Circuit-level gateways
- ⌚ Application-level gateways
- ⌚ Stateful multilayer inspection firewalls

Packet Filtering Firewall

C|EH
Certified Ethical Hacker

	Packet filtering firewalls work at the network level of the OSI model (or the IP layer of TCP/IP), they are usually a part of a router		Depending on the packet and the criteria , the firewall can drop the packet and forward it, or send a message to the originator
	In a packet filtering firewall, each packet is compared to a set of criteria before it is forwarded		Rules can include the source and the destination IP address , the source and the destination port number , and the protocol used

Internet → Firewall → **Corporate Network**

5 Application
4 TCP
3 Internet Protocol (IP)
2 Data Link
1 Physical

✓ = Traffic allowed based on source and destination IP address, packet type, and port number
✗ = Disallowed Traffic

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Packet Filtering Firewall

A packet filtering firewall investigates each **individual packet** passing through it and makes a decision whether to pass the packet or drop it. As you can tell from their name, **packet filter-based firewalls** concentrate on individual packets and analyze their header information and which way they are directed.

Traditional packet filters make the decision based on the following information:

- **Source IP address:** This is used to check if the packet is coming from a valid source or not. The information about the source IP address can be found from the **IP header** of the packet, which indicates the source system address.
- **Destination IP address:** This is used to check if the packet is going to the correct destination and to check if the destination accepts these types of packets. The information about the destination IP address can be found from the IP header of the packet, which has the destination address.
- **Source TCP/UDP port:** This is used to check the **source port** for the packet.
- **Destination TCP/UDP port:** This is used to check the destination port for the services to be allowed and the **services to be denied**.

- ➊ **TCP code bits:** Used to check whether the packet has a **SYN**, **ACK**, or other bits set for the connection to be made.
- ➋ **Protocol in use:** Used to check whether the protocol that the packet is carrying should be allowed. This is because some networks do not allow the **UDP protocol**.
- ➌ **Direction:** Used to check whether the packet is coming from the packet filter firewall or leaving it.
- ➍ **Interface:** Used to check whether or not the packet is coming from an unreliable site.

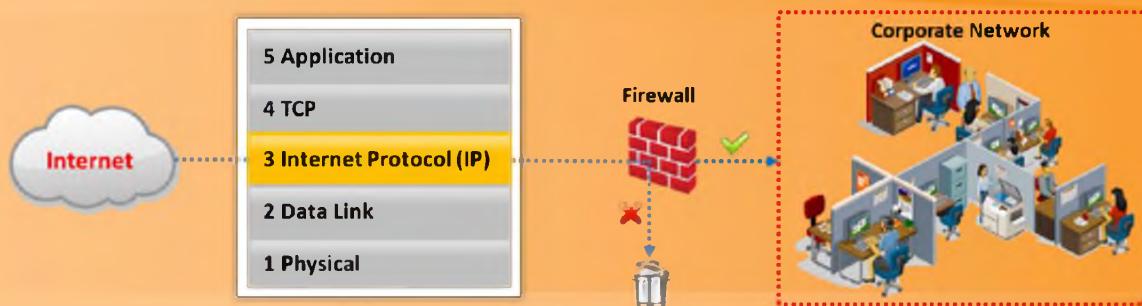


FIGURE 17.9: Packet Filtering Firewall

- ✓ = Traffic allowed based on source and destination **IP address, packet type, and port number**
✗ = Disallowed Traffic

Circuit-Level Gateway Firewall

C|EH
Certified Ethical Hacker

Circuit-level gateways work at the session layer of the OSI model or the TCP layer of TCP/IP	Information passed to a remote computer through a circuit-level gateway appears to have originated from the gateway
They monitor requests to create sessions, and determine if those sessions will be allowed	Circuit proxy firewalls allow or prevent data streams, they do not filter individual packets

The diagram illustrates the traffic flow between the Internet and a Corporate Network. On the left, a cloud icon labeled "Internet" is connected to a "Firewall" represented by a red brick wall. A dashed blue line representing data flow passes through the firewall. A green checkmark indicates the path is open, while a red X indicates it is closed. From the firewall, the dashed line continues to the right, leading into a "Corporate Network" area enclosed in a red dashed box. This network contains several office workers at desks with computers.

Legend:
✓ = Traffic allowed based on **session rules**, such as when a session is initiated by a recognized computer
✗ = Disallowed Traffic

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Circuit-level Gateway Firewall

Circuit-level gateways work at the session layer of the **OSI model** or the **TCP layer** of **P**. A circuit-level gateway forwards data between the networks without verifying it. It blocks incoming packets into the host, but allows the traffic to pass through itself. Information passed to remote computers through a circuit-level gateway appears to have originated from the gateway, as the incoming traffic carries the **IP address** of the proxy (**circuit-level gateway**).

A circuit-level gateway gives the controlled network connection to the network between the system, internal and external to it. For detecting whether or not a requested session is valid, it checks the **TCP handshaking** between the packets. Circuit-level gateways do not filter individual packets. Circuit-level gateways are relatively inexpensive and hide the information about the private network that they protect.

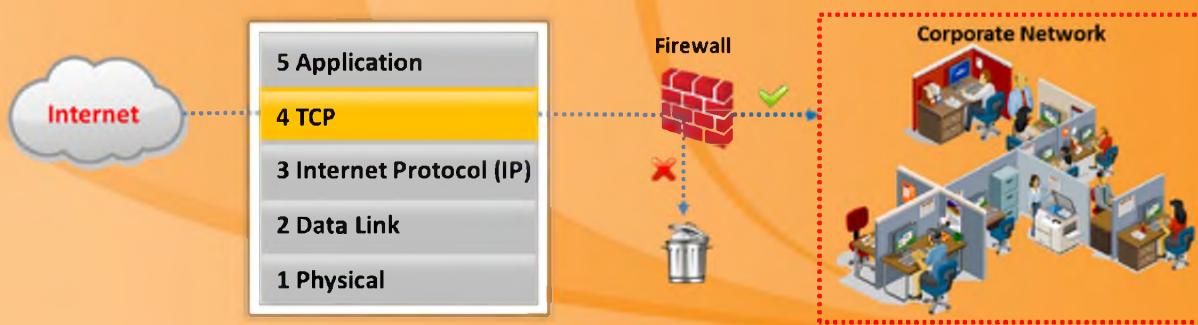
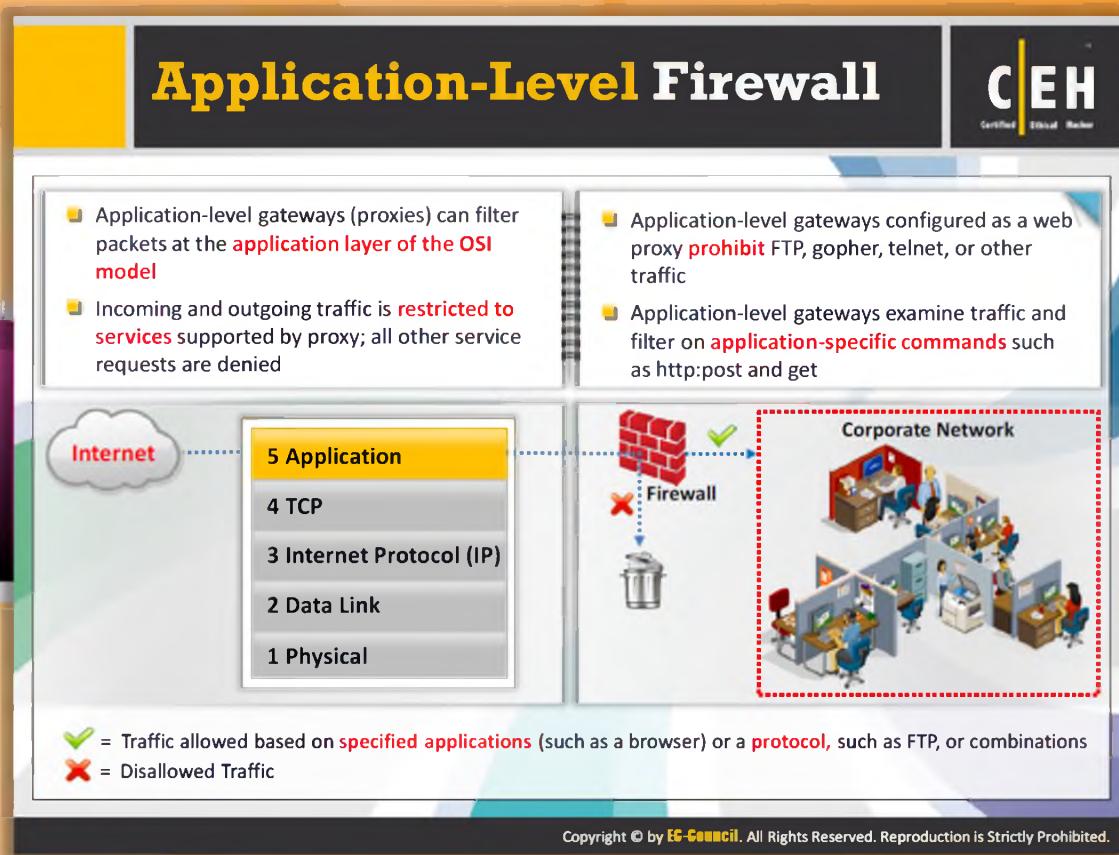


FIGURE 17.10: Circuit-level Gateway Firewall

✓ = Traffic allowed based on **session rules**, such as when a session is initiated by a recognized computer

✗ = Disallowed Traffic



Application-level Firewall

Proxy/application-based firewalls concentrate on the Application layer rather than just the packets.

- ➊ These firewalls analyze the application information to make decisions about whether or not to transmit the packets.
- ➋ A **proxy-based** firewall asks for authentication to pass the packets as it works at the Application layer.
- ➌ A content caching proxy optimizes performance by caching frequently accessed information instead of sending new requests for the same old data to the servers.

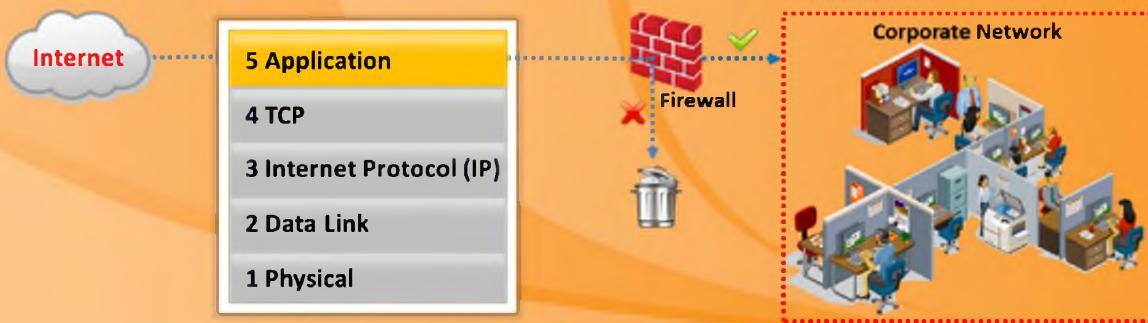


FIGURE 17.11: Application-level Firewall

Stateful Multilayer Inspection Firewall

C|EH
Certified Ethical Hacker

- Stateful multilayer inspection firewalls combine the aspects of the other three types of firewalls
- They filter packets at the network layer, to determine whether session packets are legitimate, and they evaluate the contents of packets at the application layer

Internet

5 Application
4 TCP
3 Internet Protocol (IP)
2 Data Link
1 Physical

Firewall

Corporate Network

Legend:
✓ = Traffic is filtered at three layers based on a wide range of the specified application, session, and packet filtering rules
✗ = Disallowed Traffic

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Stateful Multilayer Inspection Firewall

Stateful multilayer inspection firewalls combine the aspects of the other **three types of firewalls**. They filter packets at the network layer, to determine whether session packets are legitimate, and they evaluate the contents of packets at the application layer.

The inability of the packet filter firewall to check the header of the packets to allow the passing of packets is overcome by stateful packet filtering.

- This type of firewall can remember the packets that passed through it earlier and make decisions about future packets based on memory
- These firewalls provide the best of both **packet filtering** and **application-based filtering**
- Cisco PIX** firewalls are stateful
- These firewalls tracks and log slots or translations

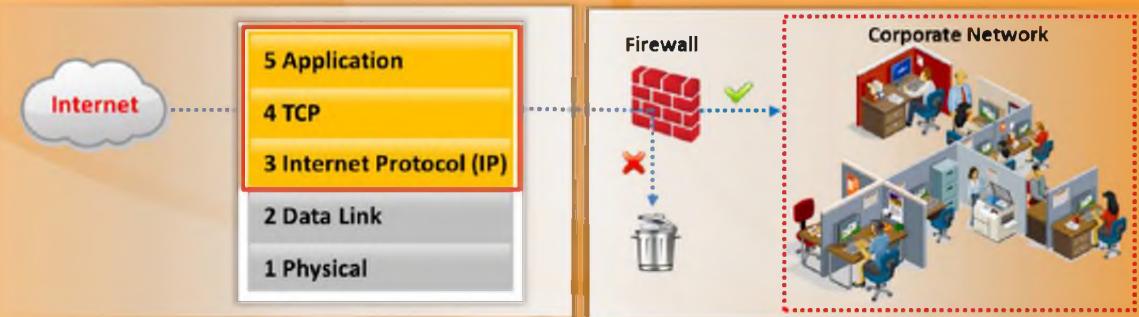


FIGURE 17.12: Stateful Multilayer Inspection Firewall

✓ = Traffic is filtered at three layers based on a wide range of the **specified application, session, and packet filtering rules**

✗ = Disallowed Traffic

Firewall Identification: Port Scanning



Port scanning is used to identify open ports and services running on these ports

Open ports can be further probed to identify the version of services, which helps in finding vulnerabilities in these services

Some firewalls will uniquely identify themselves in response to simple port scans

For example: Check Point's FireWall-1 listens on TCP ports 256, 257, 258, and 259, NetGuard GuardianPro firewall listens on TCP 1500 and UDP 1501

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Firewall Identification: Port Scanning

Systematically scanning the ports of a computer is known as port scanning. Attackers use such methods to identify the possible vulnerabilities in order to compromise a network. It is one of the most popular methods that attackers use for investigating the ports used by the victims. A tool that can be used for port scanning is **Nmap**.

A port scan helps the attacker find which ports are available (i.e., what service might be listening to a port); it consists of sending a message to each port, one at a time. The kind of response received indicates whether the port is used and can therefore be probed further for weakness. Some firewalls will uniquely identify themselves using simple port scans. For example: Check Point's FireWall-1 listens on **TCP ports 256, 257, 258, and 259** and Microsoft's Proxy Server usually listens on **TCP ports 1080 and 1745**.

Firewall Identification: Firewalking

C|EH
Certified Ethical Hacker

- A technique that uses TTL values to determine gateway **ACL filters** and map networks by analyzing IP packet responses
- Attackers send a TCP or UDP packet to the targeted firewall with a **TTL set to one hop greater** than that of the firewall
- If the packet makes it through the gateway, it is forwarded to the next hop where the TTL equals one and elicits an ICMP "**TTL exceeded in transit**" to be returned, as the original packet is discarded
- This method helps locate a firewall, additional probing permits **fingerprinting** and **identification of vulnerabilities**



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

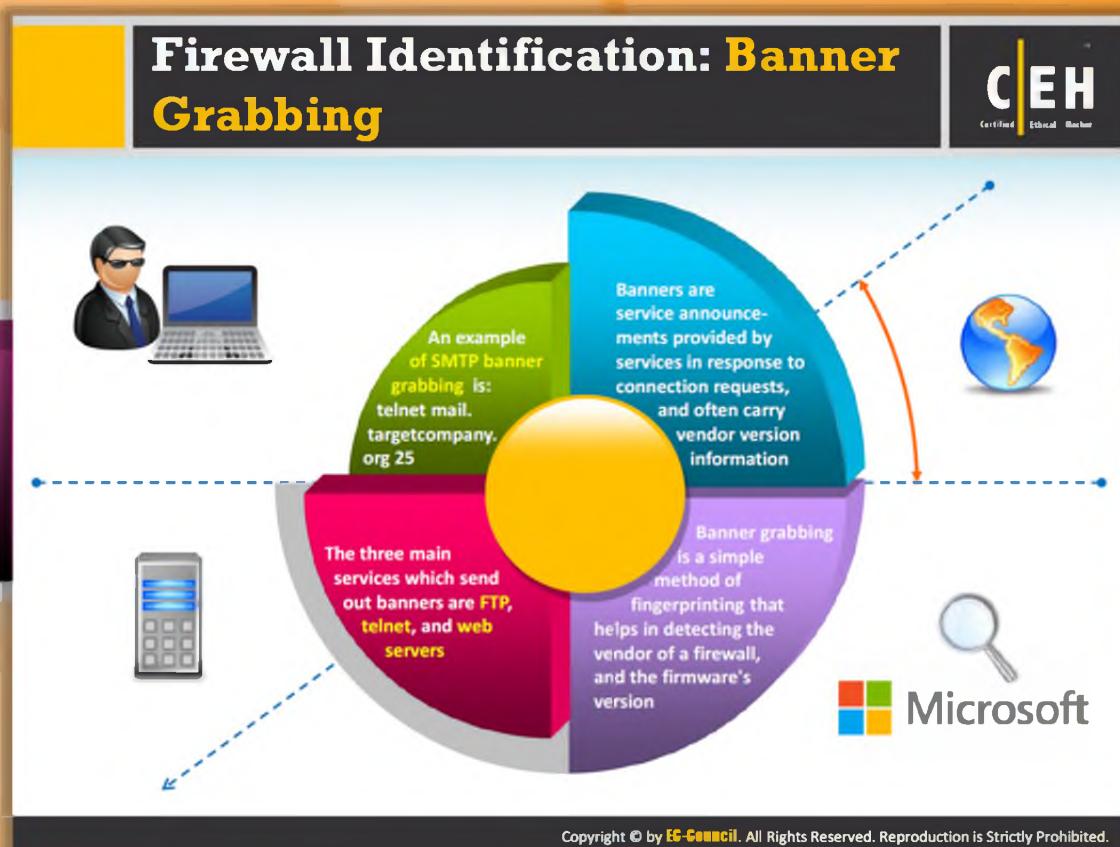


Firewall Identification: Firewalking

Firewalking is a method used to collect information about remote networks that are behind firewalls. It probes ACLs on packet filtering **routers/firewalls**. It is same as that of tracerouting and works by sending TCP or UDP packets into the firewall that have a TTL set at one hop greater than the targeted firewall. If the packet makes it through the gateway, it is forwarded to the next hop where the TTL equals zero and elicits a TTL "exceeded in transit" message, at which point the packet is discarded. Using this method, access information on the firewall can be determined if successive probe packets are sent.

Firewalk is the most well-known software used for firewalking. It has two phases: **a network discovery phase** and a **scanning phase**. It requires three hosts:

- **Firewalking host:** The firewalking host is the system, outside the target network, from which the data packets are sent, to the destination host, in order to gain more information about the target network.
- **Gateway host:** The gateway host is the system on the target network that is connected to the Internet, through which the data packet passes on its way to the target network.
- **Destination host:** The destination host is the target system on the target network that the data packets are addressed to.



Firewall Identification: Banner Grabbing

Banners are messages sent out by **network services** during the connection to the service. Banners announce which service is running on the system. Banner grabbing is a technique generally used by the attacker for **OS detection**. The attacker uses banner grabbing to discover services run by **firewalls**. The three main services that send out banners are FTP, Telnet, and web servers.

Ports of services such as FTP, Telnet, and web servers should not be kept open, as they are vulnerable to banner grabbing. A firewall does not **block banner grabbing** because the connection between the attacker's system and the target system looks legitimate.

An example of SMTP banner grabbing is: telnet mail.targetcompany.org 25. The syntax is: "<service name > <service running > <port number>"

Banner grabbing is a mechanism that is tried and true for specifying banners and application information. For example, when the user opens a telnet connection to a known port on the target server and presses Enter a few times, if required, the following result is displayed:

```
C:\>telnet www.corleone.com 80
```

```
HTTP/1.0 400 Bad Request
```

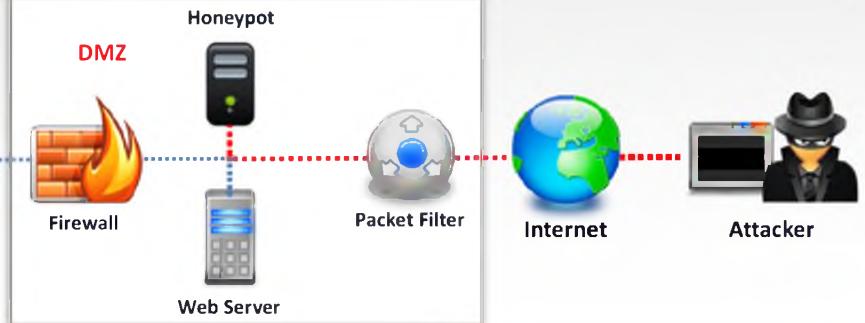
```
Server: Netscape – Commerce/1.12
```

This system works with many other common applications that respond on a set port. The information generated through banner grabbing can enhance the attacker's efforts to further compromise the system. With information about the version and the vendor of the web server, the attacker can further concentrate on employing platform-specific exploit techniques.

Honeypot

C|EH
Certified Ethical Hacker

-  A honeypot is an information system resource that is expressly **set up to attract and trap people** who attempt to penetrate an organization's network
-  It has no authorized activity, does not have any production value, and any traffic to it is **likely a probe, attack, or compromise**
-  A honeypot can **log port access attempts, or monitor an attacker's keystrokes**. These could be early warnings of a more concerted attack



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Honeypot

A honeypot is a system that is intended to **attract** and **trap** people who try unauthorized or **illicit utilization of the host system**. Whenever there is any interaction with a honeypot, it is most likely to be a malicious activity. Honeypots are unique; they do not solve a specific problem. Instead, they are a **highly flexible tool** with many different security applications. Some honeypots can be used to help prevent attacks; others can be used to detect attacks; while a few honeypots can be used for information gathering and research.

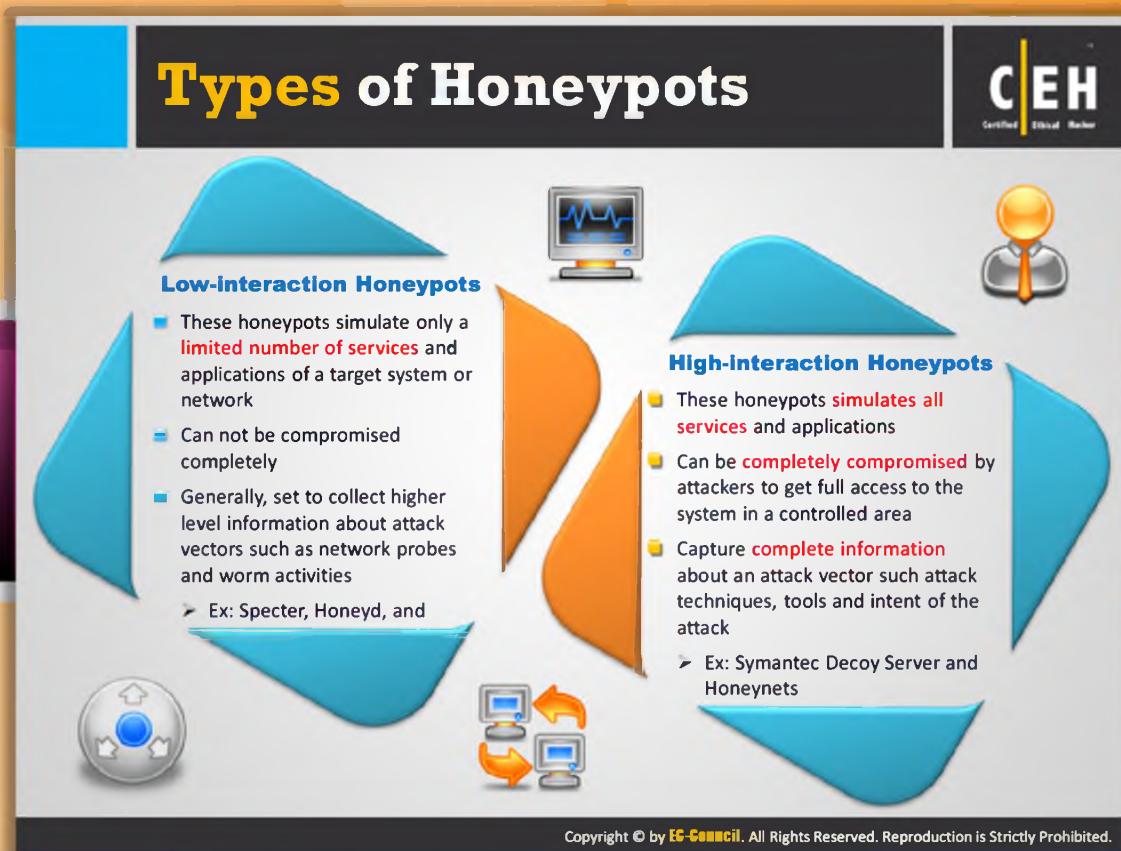
Examples:

- Installing a system on the network with no particular purpose other than to log all attempted access.
- Installing an older unpatched operating system on a network. For example, the default installation of WinNT 4 with IIS 4 can be hacked using several different techniques. A standard intrusion detection system can then be used to log hacks directed against the system and further track what the intruder attempts to do with the system once it is compromised. Install special software designed for this purpose. It has the advantage of making it look like the intruder is successful without really allowing him/her access to the network.

Any existing system can be “**honeypot-ized**.” For example, on WinNT, it is possible to rename the default administrator account and then create a dummy account called “administrator” with no password. WinNT allows extensive logging of a person's activities, so this honeypot tracks users who are attempting to gain administrator access and exploit that access.



FIGURE 17.13: Working of Honeypot



Types of Honeypots

Honeypots are mainly divided into two types:



Low-interaction Honeypot

They work by emulating services and programs that would be found on an individual's system. If the attacker does something that the emulation does not expect, the honeypot will simply **generate an error**. They capture limited amounts of information, mainly transactional data and some **limited interaction**

Ex: Specter, Honeyd, and KFSensor

Honeyd is a **low-interaction honeypot**. It is open source and designed to run primarily on UNIX systems. Honeyd works on the concept **of monitoring unused IP space**. Anytime it sees a connection attempt to an unused IP, it intercepts the connection and then interacts with the attacker, pretending to be the victim.

By default, Honeyd detects and logs connections to any **UDP or TCP port**. In addition, the user can configure emulated services to monitor specific ports, such as an emulated FTP server monitoring **port 21** (TCP). When an attacker connects to the emulated service, not only does the honeypot detect and log the activity, but also it captures all of the attacker's interaction with the emulated service.

In the case of the emulated **FTP server**, an attacker's login and password can be potentially captured; the commands that were issued, what they were looking for, or their identity can be tracked. Most emulated services work the same way. They expect a specific type of behavior, and then are programmed to react in a predetermined way.



High-interaction Honeypot

Honeynets are a prime example of a **high-interaction honeypot**. A honeynet is neither a product nor a software solution that the user installs. Instead, it is architecture, an entire network of computers designed to attack.

The idea is to have an architecture that creates a **highly controlled network**, one where all activity is controlled and captured. Within this network, intended victims are placed and the network has real computers running real applications.

The “**bad guys**” find, attack, and break into these systems on their own initiative. When they do, they do not realize they are within a honeynet. All of their activity, from **encrypted SSH** sessions to email and file uploads, is captured without them knowing it by inserting kernel modules on the victim’s systems, capturing all of the attacker’s actions.

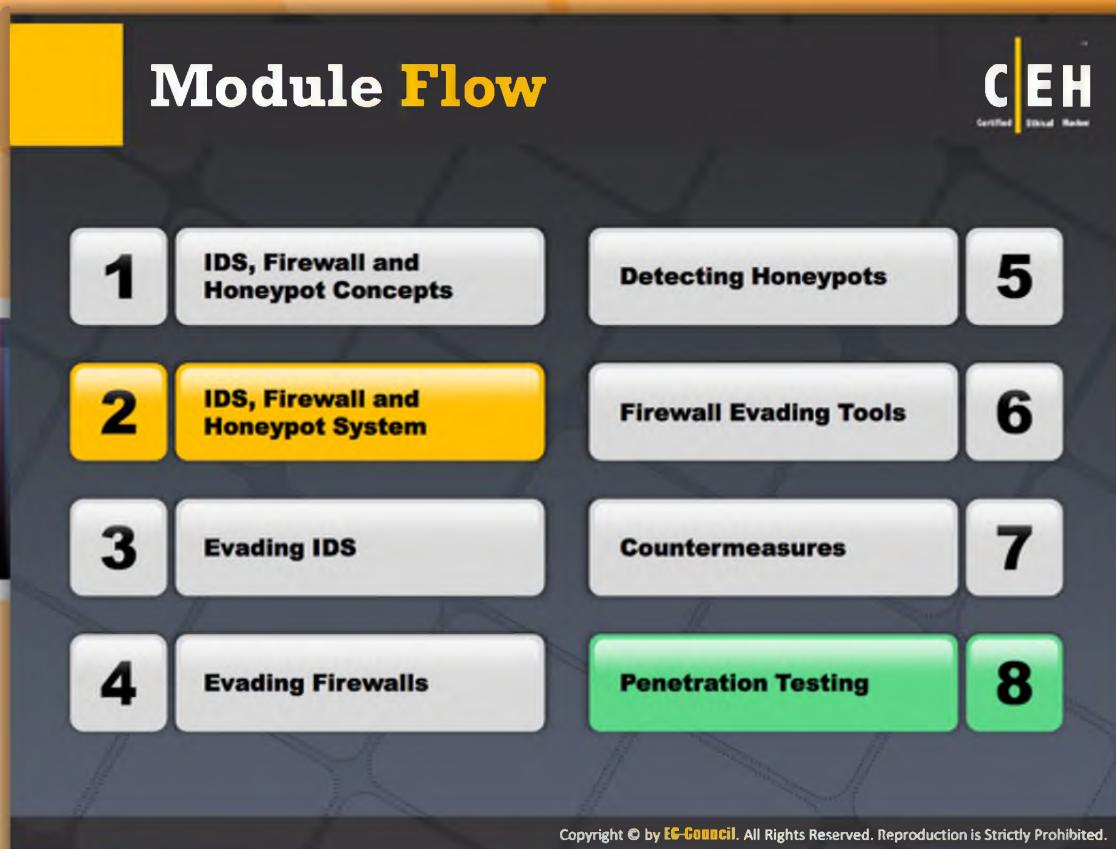
At the same time, the honeynet controls the attacker’s activity. Honeynets do this by using a honeywall gateway. This gateway allows inbound traffic to the victim’s systems, but controls the outbound traffic using intrusion prevention technologies. This gives the attacker the flexibility to interact with the victim’s systems, but prevents the attacker from harming other non-honeynet computers.



How to Set Up a Honeypot

Follow the steps here to set up a honeypot:

- ❶ Step 1: Download or purchase **honeypot software**. Tiny Honeypot, LaBrea, and Honeyd are some of the programs available for Linux systems. KFSensor is software that works with Windows.
- ❷ Step 2: Log in as an administrator on the computer to install a honeypot onto the computer.
- ❸ Step 3: Install the software on your computer. Choose the “**Full Version**” to make sure every feature of the program is installed.
- ❹ Step 4: Place the honeypot software in the Program Files folder. Once you have chosen the folder, click “**OK**” and the program will install.
- ❺ Step 5: Restart your computer for the honeypot to work.
- ❻ Step 6: Configure the honeypot to check the items that you want the honeypot to watch for, including services, applications, and Trojans, and name your domain.



Module Flow

Previously, we discussed the basic concepts of three security mechanisms: IDSes, firewalls, and honeypots. Now we will move on to detailed descriptions and functionalities of these security mechanisms.

	IDS, Firewall and Honeypot Concepts		Detecting Honeypots
	IDS, Firewall and Honeypot System		Firewall Evading Tools
	Evading IDS		Countermeasure
	Evading Firewall		Penetration Testing

This section describes the intrusion detection system Snort.

Intrusion Detection Tool: Snort

C|EH
Certified Ethical Hacker

Snort is an open source network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks

It can perform protocol analysis and content searching/matching, and is used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, and OS fingerprinting attempts

It uses a flexible rules language to describe traffic that it should collect or pass, as well as a detection engine that utilizes a modular plug-in architecture

Uses of Snort:

- Straight packet sniffer like tcpdump
- Packet logger (useful for network traffic debugging, etc.)
- Network intrusion prevention system

Command Prompt

```
c:\>snort -c c:\Snort\etc\snort.conf -l c:\Snort\log -i 2
-- Initialization Complete --
-> Snort! <-
Version 2.9.0.2-OIBC-MySON-PlexBNSP-WIN32 GDB (Build 92)
By Martin Roesch & The Snort Team: http://www.snort.org/snort-team
Copyright (C) 1998-2010 Sourcefire, Inc., et al.
Using PCRE version: 8.10 2010-06-25
Using XML version: 1.2.3
Rules Engine: SF SNORT DETECTION ENGINE Version 1.12 <Build 18>
Preprocessor: Object: SF SSL/TLS Version 1.1 <Build 4>
Preprocessor: Object: SF SSH Version 1.1 <Build 3>
Commencing packet processing (pid=589)
S5: Session exceeded configured max bytes to queue 1048576 using 1048576 bytes (client queue). 192.168.1.16 7 11616 --> 92.46.53.163 80 (0) : LMState Os1 LMFlags 0x2003
*** Caught Int-Signal

Run time for packet processing was 5985.944000 seconds
Snort processed 11774 packets.
Snort ran for 0 days 1 hours 39 minutes 45 seconds
Packets:
  Pkt/s: 11774
  Pkt/s: 118
  Pkt/s: 1
S5: Freed session from cache that was using 1098947 bytes (purge whole cache).
192.168.1.16 7 11616 --> 92.46.53.163 80 (0) : LMState Os1 LMFlags 0x22003

Packet I/O Totals:
  Received: 147490
  Analyzed: 11774 ( 7.983%)
  Dropped: 135707 ( 92.011%)
  Filtered: 0 ( 0.000%)
  Outstanding: 135716 ( 92.017%)
  Injected: 0
```

<http://www.snort.org>

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Intrusion Detection Tool: Snort

Source: <http://www.snort.org>

Snort is an open source network intrusion detection and prevention system capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis and content searching/matching. It can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting, attempts etc.

Snort uses a flexible rules language to describe traffic that it should collect or pass, as well as a detection engine that utilizes a modular plug-in architecture. Snort has a real-time alerting capability as well, incorporating alerting mechanisms for syslog, a user specified file, a UNIX socket, or WinPopup messages to Windows clients.

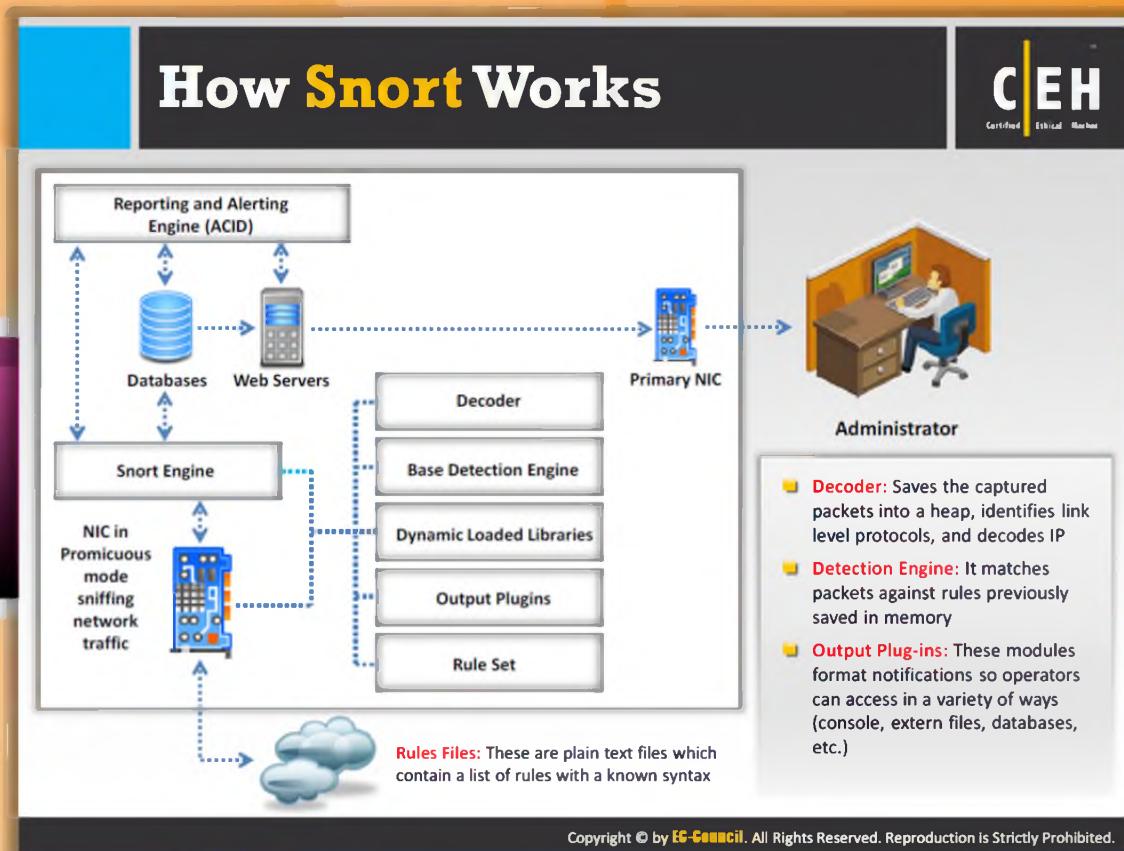
Snort has three primary uses: a straight packet sniffer like tcpdump, a packet logger (useful for network traffic debugging, etc.), or a full-blown network intrusion prevention system.

```
C:\ Command Prompt
Snort Commands
c:\Snort\bin>snort -c c:\Snort\etc\snort.conf -l c:\Snort\log -i 2
--> Snort! <*-
o" )~ Version 2.9.0.2-ODBC-MySQL-FlexRESP-WIN32 GRE (Build 92)
'*** By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-team
Copyright (C) 1998-2010 Sourcefire, Inc., et al.
Using PCRE version: 8.10 2010-06-25
Using ZLIB version: 1.2.3
Rules Engine: SF_SNORT_DETECTION_ENGINE Version 1.12 <Build 18>
Preprocessor Object: SF_SSLPP Version 1.1 <Build 4>
Preprocessor Object: SF_SSH Version 1.1 <Build 3>
Commencing packet processing (pid=5896)
S5: Session exceeded configured max bytes to queue 1048576 using 1048979 bytes (
client queue). 192.168.168.7 11616 --> 92.46.53.163 80 (0) : LWstate 0x1 LWFlags
0x2003
*** Caught Int-Signal

Run time for packet processing was 5985.944000 seconds
Snort processed 11774 packets.
Snort ran for 0 days 1 hours 39 minutes 45 seconds
Pkts/hr: 11774
Pkts/min: 118
Pkts/sec: 1
S5: Pruned session from cache that was using 1098947 bytes (purge whole cache).
192.168.168.7 11616 --> 92.46.53.163 80 (0) : LWstate 0x1 LWFlags 0x222003

Packet I/O Totals:
Received: 147490
Analyzed: 11774 ( 7.983%)
Dropped: 135707 ( 92.011%)
Filtered: 0 ( 0.000%)
Outstanding: 135716 ( 92.017%)
Injected: 0
```

FIGURE 17.14: Working of Snort in Command Promt



How Snort Works

The following are the three essential elements of the Snort tool:

- Decoder: Saves the **captured packets** into heap, identifies link level protocols, and decodes IP.
- Detection Engine: **Matches packets** against rules previously charged into memory since Snort initialization.
- Output Plug-ins: These modules format the notifications for the user to access them in different ways (console, extern files, databases, etc.).

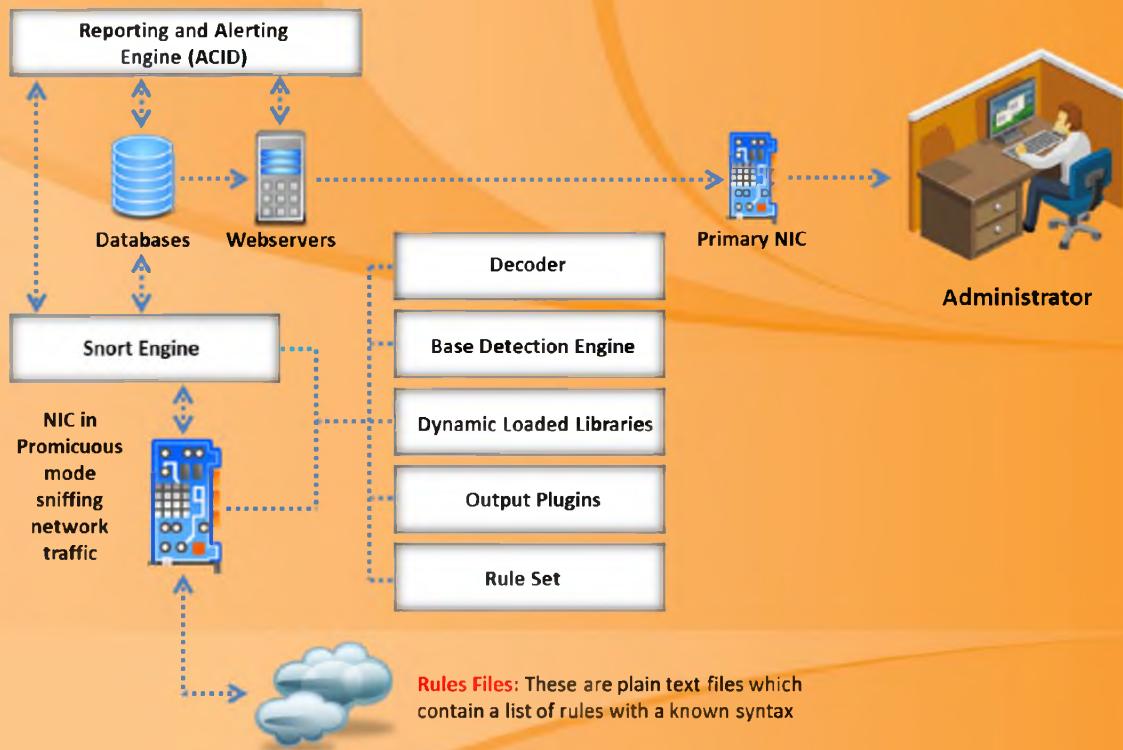


FIGURE 17.15: How Snort Works

Snort Rules

C|EH
Certified Ethical Hacker

- Snort's rule engine enables **custom rules** to meet the needs of the network
- Snort rules help in differentiating between **normal Internet activities** and **malicious activities**
- Snort rules must be contained on a **single line**, the Snort rule parser **does not handle rules on multiple lines**
- Snort rules come with two logical parts:
 - Rule header:** Identifies rule's **actions** such as alerts, log, pass, activate, dynamic, etc.
 - Rule options:** Identifies rule's **alert messages**

Example:

The diagram illustrates the structure of a Snort rule. It consists of several colored boxes connected by arrows indicating relationships:

- Rule Protocol:** A red box containing "alert" and "tcp".
- Rule Port:** A green box containing "any" and "->".
- Rule IP address:** A blue box containing "192.168.1.0/24".
- Rule Content:** A purple box containing "111" and "content::".
- Alert message:** A grey box containing the hex bytes "100 01 86 a5;" and the string "msg: \"mountd access\";")".

Arrows indicate the following relationships:

- A red arrow points from "alert" to "tcp".
- A green arrow points from "tcp" to "any".
- A blue arrow points from "any" to "192.168.1.0/24".
- A purple arrow points from "192.168.1.0/24" to "111".
- A grey arrow points from "111" to "content::".
- A grey arrow points from "content::" to the "Alert message" box.
- Red arrows point from "alert" to "Rule Action" and "Rule Format Direction".
- Green arrows point from "tcp" to "Rule Format Direction" and "Rule IP address".
- A blue arrow points from "any" to "Rule IP address".
- A purple arrow points from "192.168.1.0/24" to "Rule IP address".
- A grey arrow points from "content::" to "Alert message".

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Snort Rules

Snort uses the popular **libpcap library** (for UNIX/Linux) or **Winpcap** (for Windows), the same library that tcpdump uses to perform its packet sniffing. Snort decodes all the packets passing through the **network media** to which it is attached by entering promiscuous mode. Based on the content of the individual packets and rules defined in the configuration file, an alert is generated.

There are a number of rules that Snort allows the user to write. In addition, each of these Snort rules must describe the following:

- Any violation of the **security policy** of the company that might be a threat to the security of the company's network and other valuable information
- All the well-known and common attempts to exploit the **vulnerabilities** in the company's network
- The conditions in which a user thinks that a network packet(s) is unusual, i.e., if the identity of the packet is not authentic

Snort rules, written for both protocol analysis and content searching and matching, should be robust and flexible. The rules should be "**robust**"; it means the system should keep a rigid check on the activities taking place on the network and notify the administrator of any potential intrusion attempt. The rules should be "**flexible**"; it means that the system must be compatible

enough to act immediately and take necessary remedial measures, according to the nature of the intrusion.

Both flexibility and robustness can be achieved using an easy-to-understand and lightweight **rule-description language** that aids in writing simple Snort rules. There are two basic principles that must be kept in mind while writing Snort rules. They are as follows:

- ➊ No written rule must extend beyond a single line, so rules should be short, precise, and easy-to-understand.
- ➋ Each rule should be divided into two logical sections:
 - ➌ The rule header
 - ➍ The rule options

The rule header contains the rule's action, the protocol, the source and destination IP addresses the source and destination port information, and the **CIDR (Classless Inter-Domain Routing) block**.

The rule option section includes alert messages, in addition to information about which part of the packet should be inspected in order to determine whether the rule action should be taken.

The following illustrates a sample example of a Snort rule:



FIGURE 17.16: Rules for Snort

Snort Rules: Rule Actions and IP Protocols

Rule Actions

- The rule header stores the complete **set of rules** to identify a packet, and determines the action to be performed or what rule to be applied
- The rule action **alerts Snort** when it finds a packet that matches the rule criteria
- Three available actions in Snort:
 - Alert** - Generate an alert using the selected alert method, and then log the packet
 - Log** - Log the packet
 - Pass** - Drop (ignore) the packet

IP Protocols

Three available IP protocols that Snort supports for suspicious behavior:

- I TCP
- II UDP
- III ICMP

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Snort Rules: Rule Actions and IP Protocols

Source: <http://manual.snort.org>

The rule header contains the **information** that defines the who, where, and what of a packet, as well as what to do in the event that a packet with all the attributes indicated in the rule should show up. The first item in a rule is the rule action. The rule action tells Snort “**what to do**” when it finds a packet that matches the rule criteria. There are five available default actions in Snort: alert, log, pass, activate, and dynamic. In addition, if you are running Snort in inline mode, you have additional options which include drop, reject, and drop.

- Alert** - generate an alert using the selected alert method, and then log the packet
- Log** - log the packet
- Pass** - ignore the packet
- Activate** - alert and then turn on another dynamic rule
- Dynamic** - remain idle until activated by an activate rule, then act as a log rule
- Drop** - block and log the packet
- Reject** - block the packet, log it, and then send a **TCP reset** if the protocol is TCP or an ICMP port unreachable message if the protocol is UDP

- ⌚ **Sdrop** - block the packet but do not log it

The **Internet protocol (IP)** is used to send data from one system to another via the Internet. The IP supports unique addressing for every computer on a network. Data on the Internet protocol network is organized into packets. Each packet contains message data, source, destination, etc.

Three available IP protocols that Snort supports for suspicious behavior:

- ⌚ **TCP**: TCP (transmission control protocol) is a part of the Internet Protocol. TCP is used to connect two different hosts and exchanges data between them.
- ⌚ **UDP**: UDP, the acronym of User Datagram Protocol, is for broadcasting messages over a network.
- ⌚ **ICMP**: The Internet Control Message protocol (ICMP) is a part of the Internet protocol. It is used by the operating systems in a network to send error messages, etc.

Snort Rules: The Direction Operator and IP Addresses



The Direction Operator

- This operator indicates the direction of interest for the traffic; traffic can flow in either single direction or bi-directionally
- Example of a Snort rule using the Bidirectional Operator:

```
log !192.168.1.0/24 any <> 192.168.1.0/24 23
```

IP Addresses

- Identifies IP address and port that the rule applies to
- Use keyword "any" to define any IP address
- Use numeric IP addresses qualified with a CIDR netmask
- Example IP Address Negation Rule:

```
alert tcp !192.168.1.0/24 any -> 192.168.1.0/24 111 (content: "|00 01 86 a5|"; msg: "external mountd access";)
```

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Snort Rules: The Direction Operator and IP Addresses

The direction **operator -\$>\$** indicates the orientation, or direction, of the traffic that the rule applies to. The **IP address** and **port numbers** on the left side of the direction operator is considered to be the traffic coming from the source host, and the address and port information on the right side of the operator is the destination host. There is also a bidirectional operator, which is indicated with a **\$<>\$ symbol**. This tells Snort to consider the address/port pairs in either the source or destination orientation. This is handy for **recording/analyzing** both sides of a conversation, such as telnet or **POP3 sessions**.

Also, note that there is no \$<\$- operator. In Snort versions before 1.8.7, the direction operator did not have proper error checking and many people used an invalid token. The reason the \$<\$- does not exist is so that rules always read consistently.

The next fields in a Snort rule are used to specify the source and destination IP addresses and ports of the packet, as well as the direction in which the packet is traveling. Snort can accept a single IP address or a list of addresses. When specifying a list of IP address, you should separate each one with a comma and then enclose the list within square brackets, like this:

[192.168.1.1,192.168.1.45,10.1.1.24]

When doing this, be careful not to use any whitespace. You can also specify ranges of IP addresses using **CIDR notation**, or even include CIDR ranges within lists. Snort also allows you to apply the logical NOT operator (!) to an IP address or **CIDR range** to specify that the rule should match all but that address or range of addresses.

Snort Rules: Port Numbers

Port numbers can be listed in different ways, including "any" ports, static port definitions, port ranges, and by negation

Port ranges are indicated with the range operator ":"

Example of a Port Negation

```
log tcp any any -> 192.168.1.0/24 !6000:6010
```

Protocols	IP address	Action
Log UDP any any ->	92.168.1.0/24 1:1024	Log UDP traffic coming from any port and destination ports ranging from 1 to 1024
Log TCP any any ->	192.168.1.0/24 :5000	Log TCP traffic from any port going to ports less than or equal to 5000
Log TCP any :1024 ->	192.168.1.0/24 400:	Log TCP traffic from the well known ports and going to ports greater than or equal to 400

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Snort Rules: Port Numbers

Port numbers may be specified in a number of ways, including any ports, static port definitions, ranges, and by negation. Any ports are a wildcard value, meaning literally any port. Static ports are indicated by a single port number, such as **111** for **portmapper**, **23** for **telnet**, or **80** for **http**, etc. Port ranges are indicated with the range operator ":". The range operator may be applied in a number of ways to take on different meanings.

Example of Port Negation:

```
log tcp any any -> 192.168.1.0/24 !6000:6010
```

Protocols	IP address	Action
Log UDP any any ->	92.168.1.0/24 1:1024	Log UDP traffic coming from any port and destination ports ranging from 1 to 1024
Log TCP any any ->	192.168.1.0/24 :5000	Log TCP traffic from any port going to ports less than or equal to 5000
Log TCP any :1024 ->	192.168.1.0/24 400:	Log TCP traffic from the well known ports and going to ports greater than or equal to 400

TABLE 17.1: Port Numbers

Intrusion Detection System: Tipping Point

C|EH
Certified Ethical Hacker

- TippingPoint IPS is inserted **seamlessly** and **transparently** into the network; it is an **in-line** device
- Each packet is thoroughly inspected to determine whether it is **malicious** or **legitimate**
- It provides **performance**, **application**, and **infrastructure** protection at gigabit speeds through total packet inspection



XXXXXXX - Attacks Per Action

packets per 5 minutes

Mon 16:00 Mon 20:00 Tue 00:00 Tue 04:00 Tue 08:00 Tue 12:00
From 2009/09/21 12:22:52 To 2009/09/22 12:22:52

Action	Last	Avg	Max
Permitted	27.39 k	13.79 k	40.38 k
Blocked	0.00	0.00	0.00
Discarded Invalid	69.38	66.91	81.33

Graph Last Updated: Tue 22 Sep 12:20:02 CEST 2009

XXXXXXX - Attacks Per Protocol

attacks per 5 minutes

Mon 16:00 Mon 20:00 Tue 00:00 Tue 04:00 Tue 08:00 Tue 12:00
From 2009/09/21 12:22:52 To 2009/09/22 12:22:52

Protocol	Last	Avg	Max
ICMP	3.67 k	3.90 k	6.06 k
UDP	886.08	1.04 k	6.61 k
TCP	22.90 k	8.94 k	35.85 k
IP-Other	0.00	0.00	0.00

Graph Last Updated: Tue 22 Sep 12:20:02 CEST 2009

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Intrusion Detection System: Tipping Point

Source: <http://h10163.www1.hp.com>

TippingPoint IPS is inserted **seamlessly** and **transparently** into the network; it is an **in-line** device. Each packet is thoroughly inspected to determine whether it is **malicious** or **legitimate**. It provides **performance**, **application**, and **infrastructure** protection at gigabit speeds through total packet inspection.

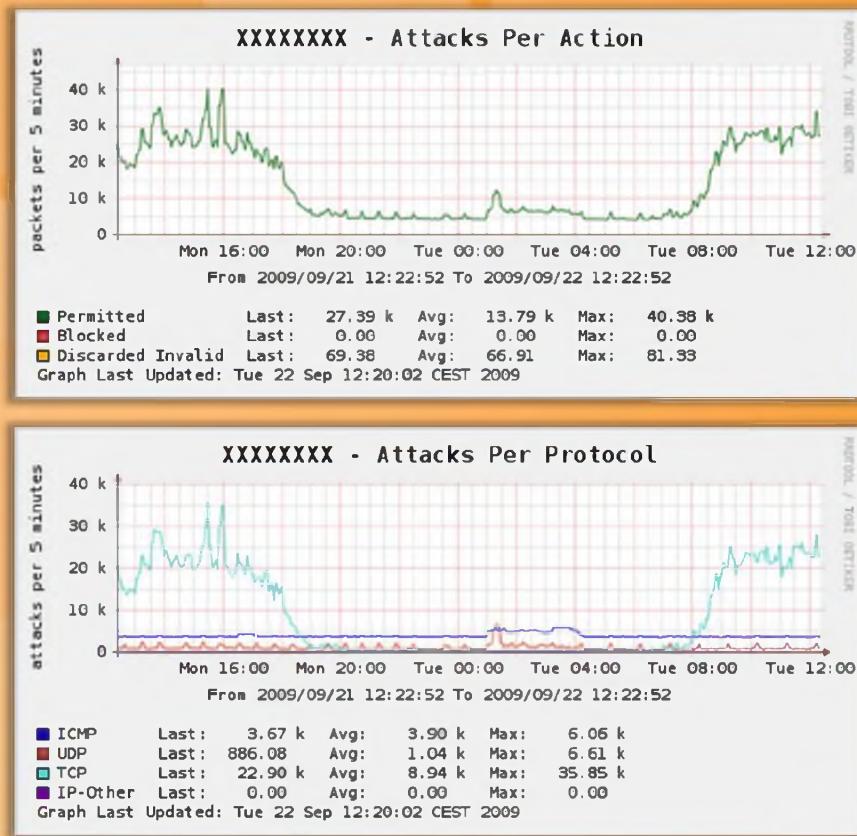


FIGURE 17.17: Tipping Point Screenshot

Intrusion Detection Tools

C|EH
Certified Ethical Hacker

 IBM Security Network Intrusion Prevention System http://www-01.ibm.com	 OSSEC http://www.ossec.net
 Peek & Spy http://networkingdynamics.com	 Cisco Intrusion Prevention Systems http://www.cisco.com
 INTOUCH INSA-Network Security Agent http://www.ttinet.com	 AIDE (Advanced Intrusion Detection Environment) http://aide.sourceforge.net
 Strata Guard http://www.stillsecure.com	 SNARE (System iNtrusion Analysis & Reporting Environment) http://www.intersectalliance.com
 IDP8200 Intrusion Detection and Prevention Appliances https://www.juniper.net	 Vanguard Enforcer http://www.go2vanguard.com

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Intrusion Detection Tools

Intrusion detection tools detect anomalies. These tools, when run on a dedicated workstation, read all network packets, reconstruct user sessions, and scan for possible intrusions by looking for attack signatures and **network traffic statistical anomalies**. In addition, these tools give real-time, zero-day protection from network attacks and malicious traffic, and prevent malware, spyware, port scans, viruses, and **DoS** and **DDoS** from compromising hosts. A few of intrusion detection tools are listed as follows:

- ➊ IBM Security Network Intrusion Prevention System available at <http://www-01.ibm.com>
- ➋ Peek & Spy available at <http://networkingdynamics.com>
- ➌ INTOUCH INSA-Network Security Agent available at <http://www.ttinet.com>
- ➍ Strata Guard available at <http://www.stillsecure.com>
- ➎ IDP8200 Intrusion Detection and Prevention Appliances available at <https://www.juniper.net>
- ➏ OSSEC available at <http://www.ossec.net>
- ➐ Cisco Intrusion Prevention Systems available at <http://www.cisco.com>

- ⌚ AIDE (Advanced Intrusion Detection Environment) available at <http://aide.sourceforge.net>
- ⌚ SNARE (System iNtrusion Analysis & Reporting Environment) available at <http://www.intersectalliance.com>
- ⌚ Vanguard Enforcer available at <http://www.go2vanguard.com>

Intrusion Detection Tools (Cont'd)



 Check Point Threat Prevention Appliance http://www.checkpoint.com	 FortiGate http://www.fortinet.com
 fragroute http://www.monkey.org	 Enterasys® Intrusion Prevention System http://www.enterasys.com
 Next-Generation Intrusion Prevention System (NGIPS) http://www.sourcefire.com	 StoneGate Virtual IPS Appliance http://www.stonesoft.com
 Outpost Network Security http://www.agnitum.com	 Cyberoam Intrusion Prevention System http://www.cyberoam.com
 Check Point IPS-1 http://www.checkpoint.com	 McAfee Host Intrusion Prevention for Desktops http://www.mcafee.com

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Intrusion Detection Tools (Cont'd)

In addition, to the previously mentioned intrusion detection tools, there are few more tools that can be used for detecting intrusions:

- ➊ Check Point Threat Prevention Appliance available at <http://www.checkpoint.com>
- ➋ Fragroute available at <http://www.monkey.org>
- ➌ Next-Generation Intrusion Prevention System (NGIPS) available at <http://www.sourcefire.com>
- ➍ Outpost Network Security available at <http://www.agnitum.com>
- ➎ Check Point IPS-1 available at <http://www.checkpoint.com>
- ➏ FortiGate available at <http://www.fortinet.com>
- ➐ Enterasys® Intrusion Prevention System available at <http://www.enterasys.com>
- ➑ StoneGate Virtual IPS Appliance available at <http://www.stonesoft.com>
- ➒ Cyberoam Intrusion Prevention System available at <http://www.cyberoam.com>
- ➓ McAfee Host Intrusion Prevention for Desktops available at <http://www.mcafee.com>



Firewall: ZoneAlarm PRO Firewall

Source: <http://www.zonealarm.com>

ZoneAlarm PRO Firewall **blocks attackers** and intruders from accessing your system. It monitors programs for suspicious behavior, spotting and stopping new attacks that bypass traditional antivirus protection. It prevents identity theft by guarding your personal data. It even erases your tracks allowing you to surf the web in complete privacy. Furthermore, it locks out attackers, blocks intrusions, and makes your **PC invisible online**. In addition, it filters out annoying and potentially dangerous email.

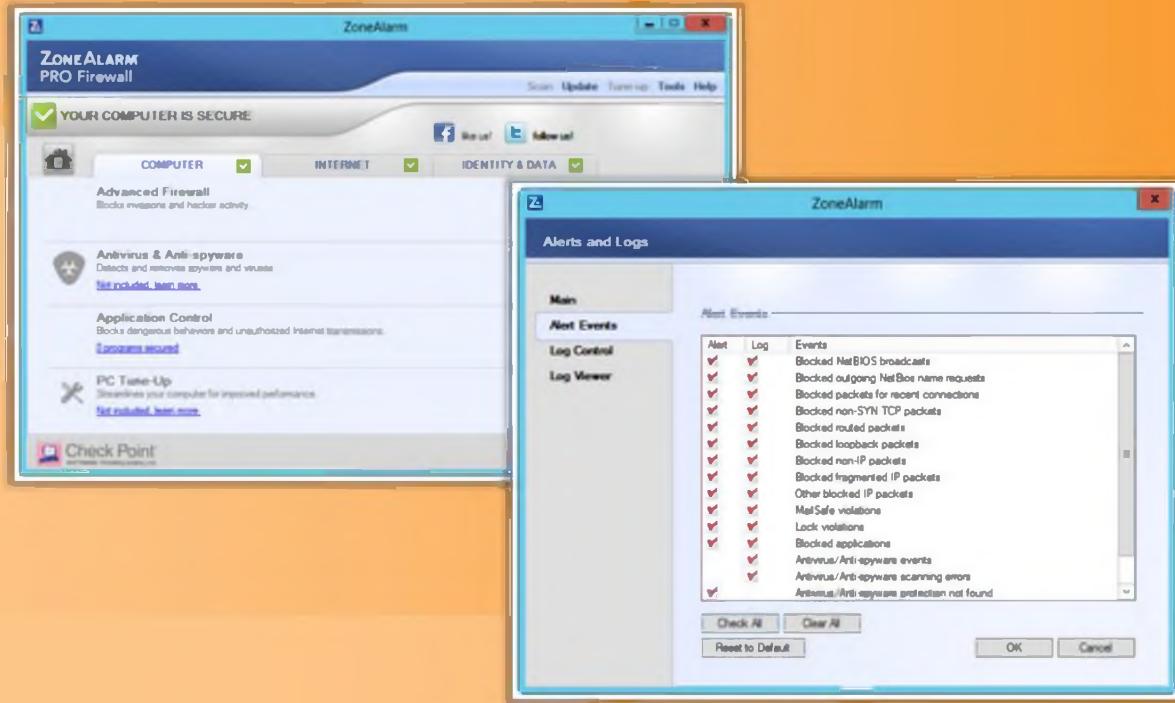


FIGURE 17.18: ZoneAlarm PRO Firewall Screenshot

Firewalls

C|EH
Certified Ethical Hacker

 Check Point Firewall Software Blade http://www.checkpoint.com	 Firewall UTM http://www.esoft.com
 eScan Enterprise Edition http://www.escanav.com	 Sonicwall http://www.tribecaexpress.com
 Jetico Personal Firewall http://www.jetico.com	 Comodo Firewall http://personalfirewall.comodo.com
 Outpost Security Suite http://free.agnitum.com	 Online Armor http://www.online-armor.com
 Novell BorderManager http://www.novell.com	 FortiGate-5101C http://www.fortinet.com

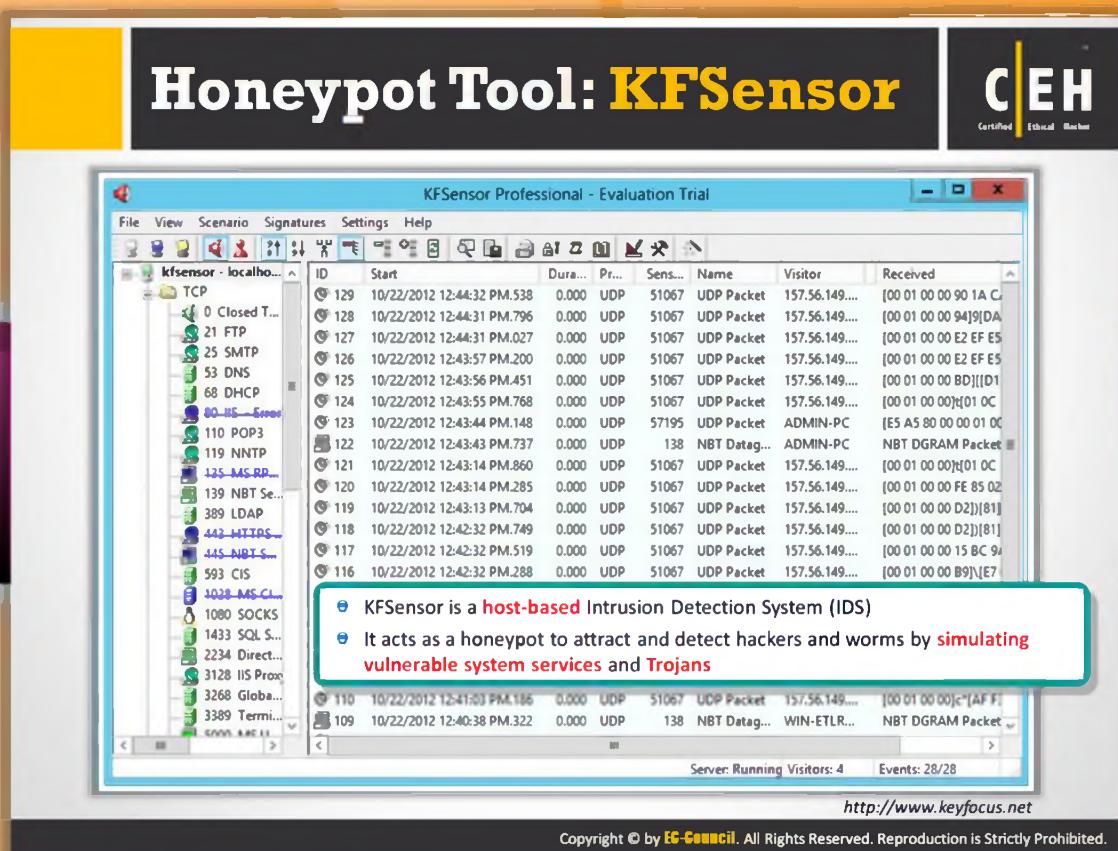
Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Firewalls

Firewalls provide essential protection to the **computers against viruses**, privacy threats, objectionable content, hackers, and malicious software when networked or connected to the Internet. A firewall **monitors running applications** that access the network. It analyzes downloads and warns you if downloading a malicious file, stops it from infecting your PC. A few of the firewalls that provide system protection are listed as follows:

- ➊ Check Point Firewall Software Blade available at <http://www.checkpoint.com>
- ➋ eScan Enterprise available at <http://www.escanav.com>
- ➌ Jetico Personal Firewall available at <http://www.jetico.com>
- ➍ Outpost Security Suite available at <http://free.agnitum.com>
- ➎ Novell BorderManager available at <http://www.novell.com>
- ➏ Firewall UTM available at <http://www.esoft.com>
- ➐ Sonicwall available at <http://www.tribecaexpress.com>
- ➑ Comodo Firewall available at <http://personalfirewall.comodo.com>
- ➒ Online Armor available at <http://www.online-armor.com>
- ➓ FortiGate-5101C available at <http://www.fortinet.com>



Honeypot Tool: KFSensor

Source: <http://www.keyfocus.net>

KFSensor is a **Windows-based honeypot intrusion detection system (IDS)**. It acts as a honeypot to attract and detect hackers and worms by simulating vulnerable system services and Trojans. By acting as a decoy server, it can divert attacks from critical systems and provide a higher level of information than can be achieved by using **firewalls** and **NIDS alone**.

KFSensor is designed for use in a Windows-based corporate environment and contains many innovative and unique features such as remote management, a Snort-compatible signature engine, and emulations of Windows networking protocols.

Features:

- GUI-based management console
- Remote management
- Snort compatible signature engine
- Emulations of Windows networking protocols
- Export logs in multiple formats

🕒 Denial-of-service (DOS) attack protection

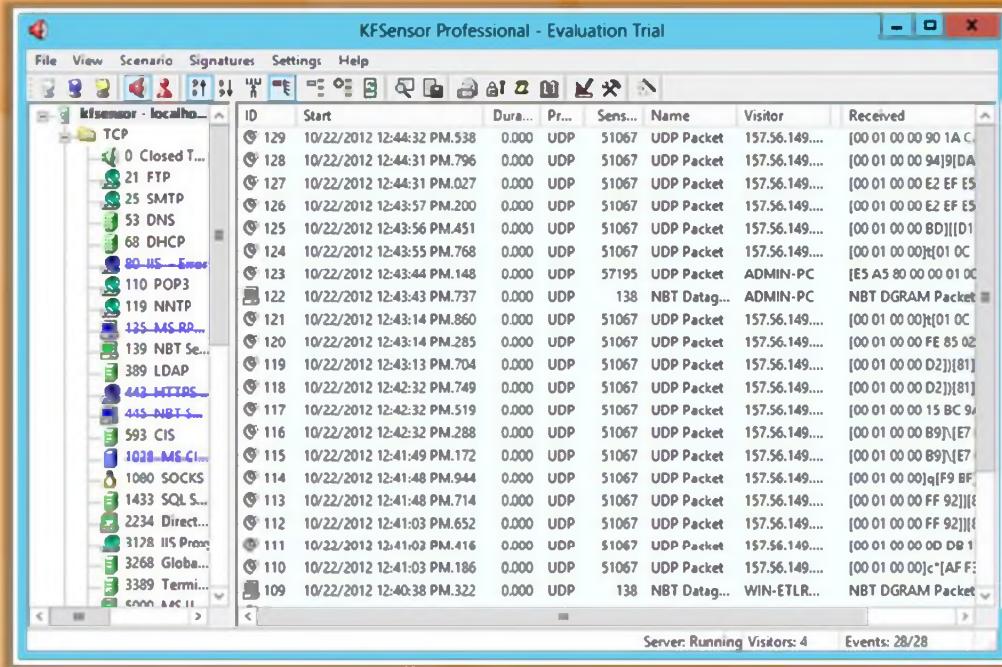


FIGURE 17.19: KFSensor Screenshot

Honeypot Tool: SPECTER

C|EH
Certified Ethical Hacker

- SPECTER is a smart honeypot-based intrusion detection system that offers common Internet services such as SMTP, FTP, POP3, HTTP, and TELNET which appear perfectly normal to the attackers but in fact are traps
- SPECTER provides massive amounts of decoy content including images, MP3 files, email messages, password files, documents, and all kinds of software

The screenshot shows the SPECTER Control interface. On the left, there's a sidebar with a magnifying glass icon containing a spider, and a blue ribbon-like graphic. The main window has several tabs: Services, Ports, Applications, Engine, and Engine Messages. Under Services, various operating systems like Windows 98, Windows NT, and Mac OS X are listed. Under Applications, there are checkboxes for DNS, TELNET, IMAP4, RINGER, SSH, SUB 7, and GENERIC. The Engine tab shows engine version 1.00, threads 1, and connection settings. The Engine Messages tab displays a log of captured traffic, including entries for TELNET, IMAP4, and POP3. Buttons for Start Engine, Stop Engine, and Log Analyzer are visible at the bottom.

<http://www.specter.com>

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Honeypot Tool: SPECTER

Source <http://www.specter.com>

SPECTER is a **honeypot** or **deception system**. It simulates a **complete system**, providing an **interesting target** to lure hackers away from production systems. It offers common Internet services such as SMTP, FTP, POP3, HTTP, and TELNET, which appear perfectly normal to attackers. However, they are traps so that traces are left without the attacker knowing that they are connected to a decoy system that does none of the things it appears to do; but instead, it logs everything and notifies the appropriate people.

Furthermore, SPECTER automatically **investigates attackers** while they are still trying to break in. It provides massive amounts of decoy content and it generates decoy programs that can't leave hidden marks on the attacker's computer. Automated weekly online updates of the honeypot's content and vulnerability databases allow the honeypot to change constantly without user interaction.

Advantages:

- Suspicious interest in the network, and computers, can be detected immediately.
 - Administrators are notified of hostile activity when it happens, so that they can immediately look at the problem and take action.

- ➊ The system is very easy to set up and configure while providing sophisticated features. Fully automated online updates of the honeypot's content and vulnerability databases allow the honeypot to change constantly without user interaction.
- ➋ There cannot be false alerts, as a legitimate user cannot connect to the honeypot.
- ➌ Specter simulates in **14 different operating systems**:
- ➍ Windows 98, Windows NT, Windows 2000, Windows XP, Linux, Solaris, Tru64, NeXTStep, Irix, Unisys Unix, AIX, MacOS, MacOS X, and FreeBSD.

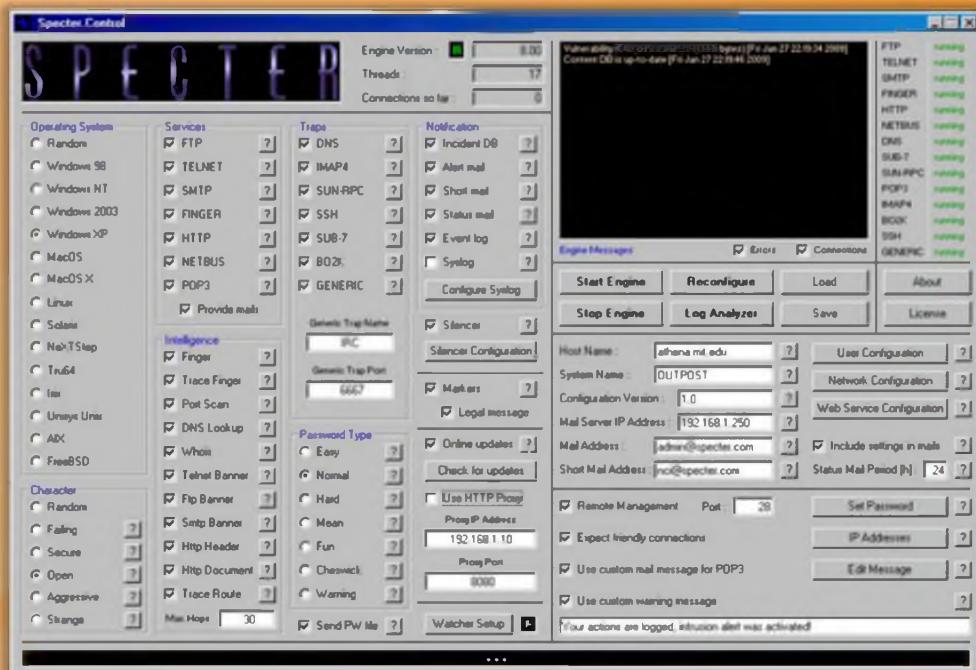


FIGURE 17.20: SPECTER Screenshot

Honeypot Tools



 LaBrea Tarpit http://labrea.sourceforge.net	 WinHoneyd http://www2.netvigilance.com
 PatriotBox http://www.alkasis.com	 HIHAT http://hihat.sourceforge.net
 Kojoney http://kojoney.sourceforge.net	 Argos http://www.few.vu.nl
 HoneyBOT http://www.atomicsoftwaresolutions.com	 Glastopf http://glastopf.org
 Google Hack Honeypot http://ghh.sourceforge.net	 Send-Safe Honeypot Hunter http://www.send-safe.com

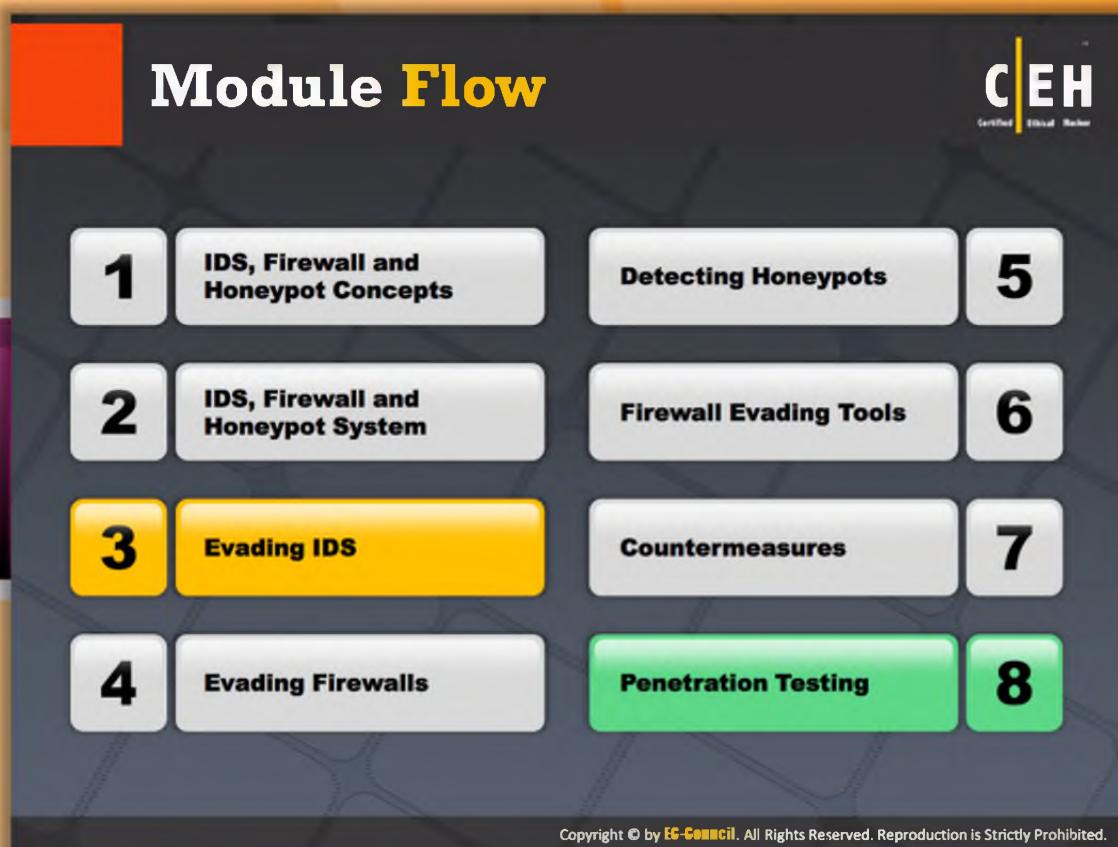
Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Honeypot Tools

Honeypots are the security tools that give the **security community** an opportunity to monitor attackers' tricks and exploits by logging their every activity, so that they can respond to these exploits quickly without attackers actually misusing and compromising systems. A few honeypot tools are listed as follows:

- ④ LaBrea Tarpit available at <http://labrea.sourceforge.net>
- ④ PatriotBox available at <http://www.alkasis.com>
- ④ Kojoney available at <http://kojoney.sourceforge.net>
- ④ HoneyBOT available at <http://www.atomicsoftwaresolutions.com>
- ④ Google Hack Honeypot available at <http://ghh.sourceforge.net>
- ④ WinHoneyd available at <http://www2.netvigilance.com>
- ④ HIHAT available at <http://hihat.sourceforge.net>
- ④ Argos available at <http://www.few.vu.nl>
- ④ Glastopf available at <http://glastopf.org>
- ④ Send-Safe Honeypot Hunter available at <http://www.send-safe.com>

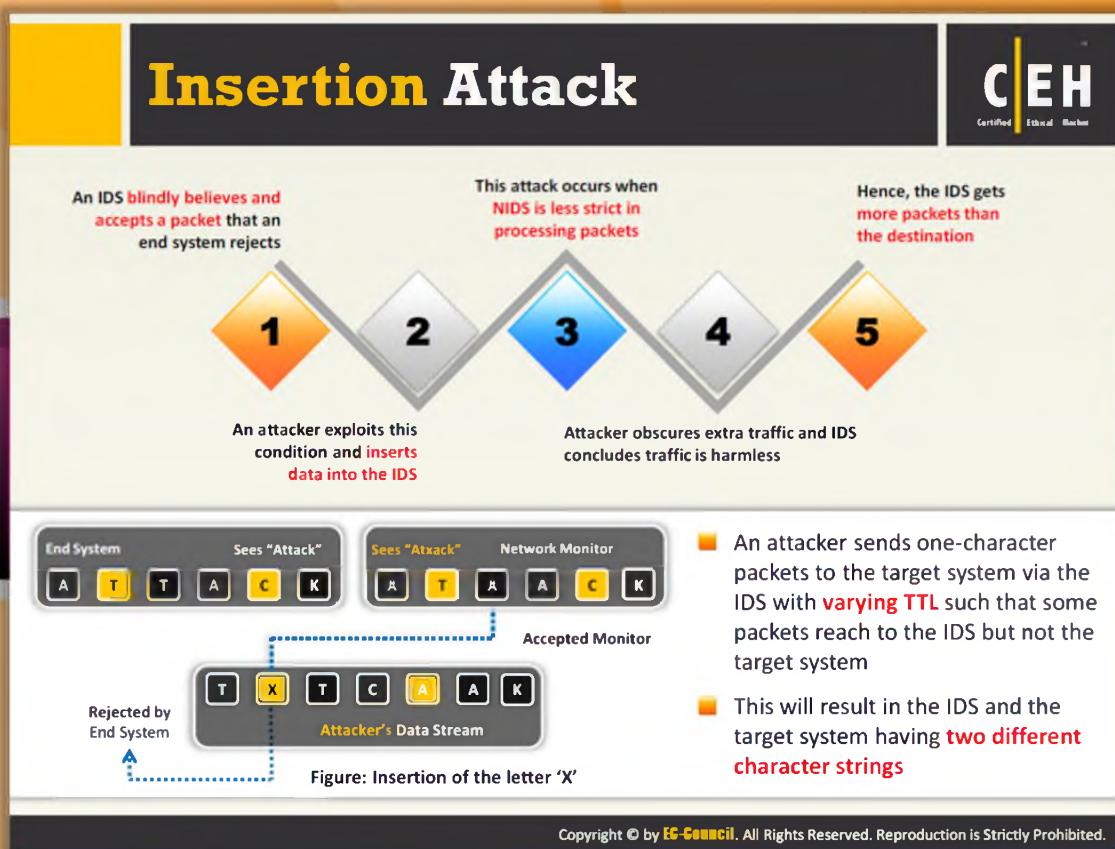


Module Flow

An IDS is the critical security mechanism implemented in order to prevent intrusions and at the same time, to alert the security personnel when an attacker attempts to intrude into the network. An IDS can detect the attacker's attempts of breaking into the network. In order to avoid being detected by the IDS, attackers try to evade IDSe.

IDS, Firewall and Honeypot Concepts	Detecting Honeypots
IDS, Firewall and Honeypot System	Firewall Evading Tools
Evading IDS	Countermeasure
Evading Firewall	Penetration Testing

This section describes the ways in which attackers try to evade IDSe.



Insertion Attack

The process where the **attacker confuses** the **IDS** by forcing it to read the invalid packets is known as insertion, that is, the packet would not be accepted by the system to which it is addressed. If a packet is malformed or if it does not reach its actual destination, the packet is invalid. If the IDS read an invalid packet, the IDS will become confused.

To understand how insertion becomes a problem for a network IDS, it is important to understand how **IDSes detect attacks**. The IDS employs pattern-matching algorithms to look for specific patterns of data in a packet or stream of packets. For example, IDSes might look for the string "phf" in an HTTP request to discover a PHF Common Gateway Interface (CGI) attack. An attacker who can insert packets into the IDS can prevent pattern matching from working. For instance, an attacker can send the string "phf" to a web server, attempting to exploit the CGI vulnerability, but force the IDS to read "phoneyf" (by "inserting" the string "oney") instead. One simple insertion attack involves intentionally corrupting the IP checksum. Every packet transmitted on an IP network has a checksum that is used to verify whether the packet was corrupted in transit. IP checksums are 16-bit numbers that are computed by examining information in the packet. If the checksum on an IP packet does not match the actual packet, the host to which it is addressed will not accept it, while the IDS might consider it as part of the effective stream.

For example, the attacker can send packets whose Time to live fields have been crafted to reach the IDS but not the target computers. An attacker confronts the IDS with a stream of one-character packets (the attacker-originated data stream), in which one of the characters (the letter 'X') will be accepted only by the IDS. As a result, the IDS and the end system reconstruct two different strings.

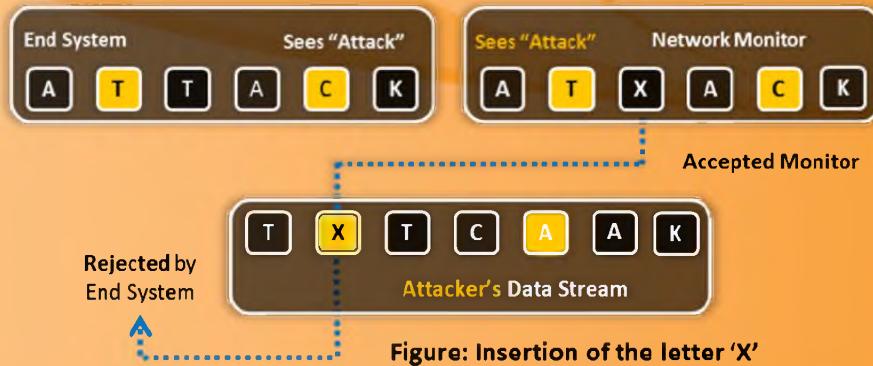


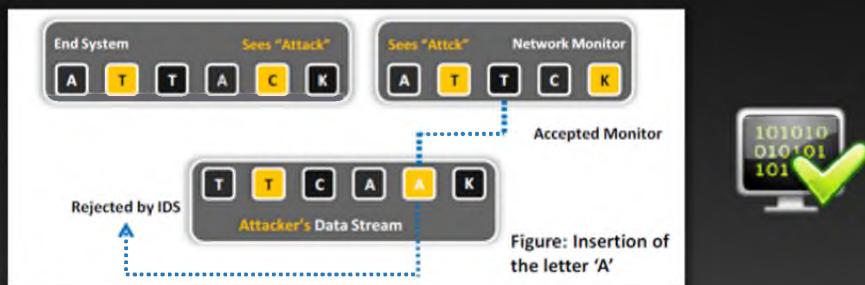
Figure: Insertion of the letter 'X'

FIGURE 17.21: Insertion Attack

Evasion



- 1** In this evasion technique, an end system **accepts a packet** that an IDS rejects
 - 2** Using this technique, an attacker **exploits** the host computer
 - 3** Attacker sends **portions of the request** in packets that the IDS mistakenly rejects, allowing the removal of parts of the stream from the IDS
 - 4** For example, if the malicious sequence is sent **byte-by-byte**, and one byte is rejected by the IDS, the IDS cannot detect the attack
 - 5** Here, the IDS gets fewer packets than the destination



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

An “evasion” attack occurs when the **IDS discards** a packet that the host to which it is addressed accepts. Evasion attacks are devastating to the accuracy of the IDS. An evasion attack at the IP layer allows an attacker to attempt arbitrary attacks against hosts on a network, without the IDS ever realizing it. The attacker sends portions of the request in packets that the IDS mistakenly rejects, allowing the removal of parts of the stream from the **ID system's** view. For example, if the malicious sequence is sent byte-by-byte, and one byte is rejected by the IDS, the IDS cannot detect the attack. Here, the IDS gets fewer packets than the destination.

One example of an evasion attack occurs when an attacker opens a **TCP connection** with a data packet. Before any TCP connection can be used, it must be “**opened**” with a handshake between the two endpoints of the connection. A fairly obscure fact about TCP is that the handshake packets can themselves bear data. **IDSes** that do not accept the data in these packets are vulnerable to an evasion attack.

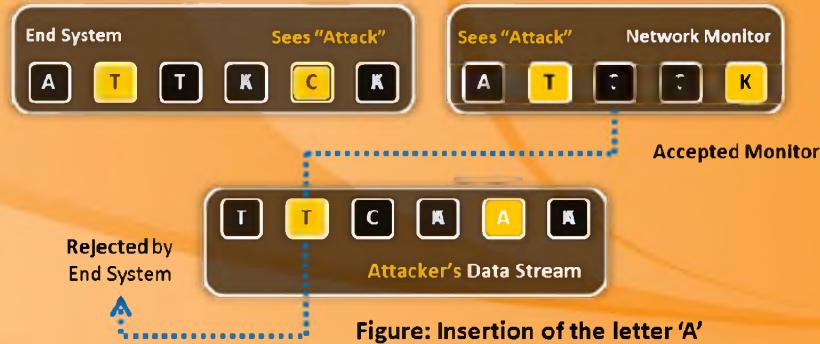


Figure: Insertion of the letter 'A'

FIGURE 17.22: Evasion

Denial-of-Service Attack (DoS)

C|EH
Certified Ethical Hacker

- Many IDSs use a **centralized server for logging** alerts
- If attackers know the IP address of the centralized server they can perform **DoS** or other hacks to slow down or crash the server
- As a result, attackers' intrusion attempts will not be logged

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Denial-of-Service Attack (DoS)

Multiple types of denial-of-service attacks are valid against **IDS systems**. The attacker identifies a point of network processing that requires the allocation of a resource, causing a condition to occur that consumes all of that resource. The resources that can be affected by the attacker are CPU cycles, memory, disk space, and network bandwidth. The **CPU capabilities** of the IDS can be monitored and affected. This is because IDS needs half of the CPU cycle to read the packets, detecting what the purpose of their existence is, and then comparing them with some location in the **saved network state**. An attacker can verify the most computationally expensive network processing operations and then compel the IDS to spend all its time carrying out useless work.

An IDS requires memory for a variety of things. For generating a match for the patterns, the TCP connections should be saved, the reassembly queues should be maintained, and the buffers of the data should be generated. In the initial phase, the system requires memory so that it can read the packets. Memory is allocated by the system. It is needed for network processing operations. An attacker can verify the processing operations that require the ID system to allocate memory and force the IDS to allocate all of its memory for meaningless information.

In certain circumstances, the ID systems store activity logs on the disk. The stored events occupy most of the disk space. Most computers have limited disk space. The attackers can

occupy a major part of the disk space on the IDS by creating and storing a large number of useless events. This renders the **IDS useless** in terms of storing real events.

Network IDS systems record the activity on the networks they monitor. They are competent because networks are hardly ever used to their full capacity; few monitoring systems can cope with an extremely busy network.

The IDS system, unlike an end system, must read everyone's packets, not just those sent specifically to it. An attacker can overload the network with meaningless information and prevent the IDS system from keeping up with what is actually happening on the network.

Many IDSEs today employ central logging servers that are used exclusively to store IDS alert logs. The central server's function is to centralize alert data so it can be viewed as a whole rather than on a system-by-system basis.

However, if attackers know the **central log server's IP address**, they could slow it down or even crash it using a DoS attack. After the server is shut down, attacks could go unnoticed because the alert data is no longer being logged.

Using this evasion technique, an attacker:

- ⌚ Consumes the device's processing power and allows attacks to sneak by
- ⌚ Fills up disk space causing attacks to not be logged
- ⌚ Causes more alarms than can be handled by management systems (such as databases, ticketing systems, etc.)
- ⌚ Causes personnel to be unable to investigate all the alarms
- ⌚ Causes the device to lock up

Obfuscating

An IDS can be evaded by obfuscating or encoding the attack payload in a way that the target computer understands but the IDS will not

Attackers can encode attack patterns in unicode to bypass IDS filters, but be understood by an IIS web server

Polymorphic code is another means to circumvent signature-based IDSes by creating unique attack patterns, so that the attack does not have a single detectable signature

Attackers manipulate the path referenced in the signature to fool the HIDS

Attacks on encrypted protocols such as HTTPS are obfuscated if the attack is encrypted

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Obfuscating

Obfuscation means to make **code harder** to understand or read, generally for privacy or **security purposes**. A tool called an **obfuscator** is sometimes used to convert a straightforward program into one that works the same way but is much harder to understand.

An IDS can be evaded by obfuscating or encoding the attack payload in a way that the target computer will reverse but the IDS will not. An **attacker manipulates** the path referenced in the signature to fool the **HIDS**. Using the Unicode character, an attacker could encode attack packets that the IDS would not recognize but that an IIS web server would decode and become attacked. Polymorphic code is another means to circumvent signature-based IDSes by creating unique attack patterns, so that the attack does not have a single detectable signature. Attacks on encrypted protocols such as **HTTPS are obfuscated** if the attack is encrypted. Polymorphic code is another means to circumvent signature-based IDSes by creating unique attack patterns, so that the attack does not have a single detectable signature.

False Positive Generation

CEH
Certified Ethical Hacker

Attackers with the knowledge of the target IDS, **craft malicious packets** just to generate alerts

These packets are sent to the IDS to generate a **large number of false positive** alerts

Attackers then use these false positive alerts to **hide real attack traffic**

Attackers can bypass IDS unnoticed as it is **difficult to differentiate** the attack traffic from the large volume of false positives

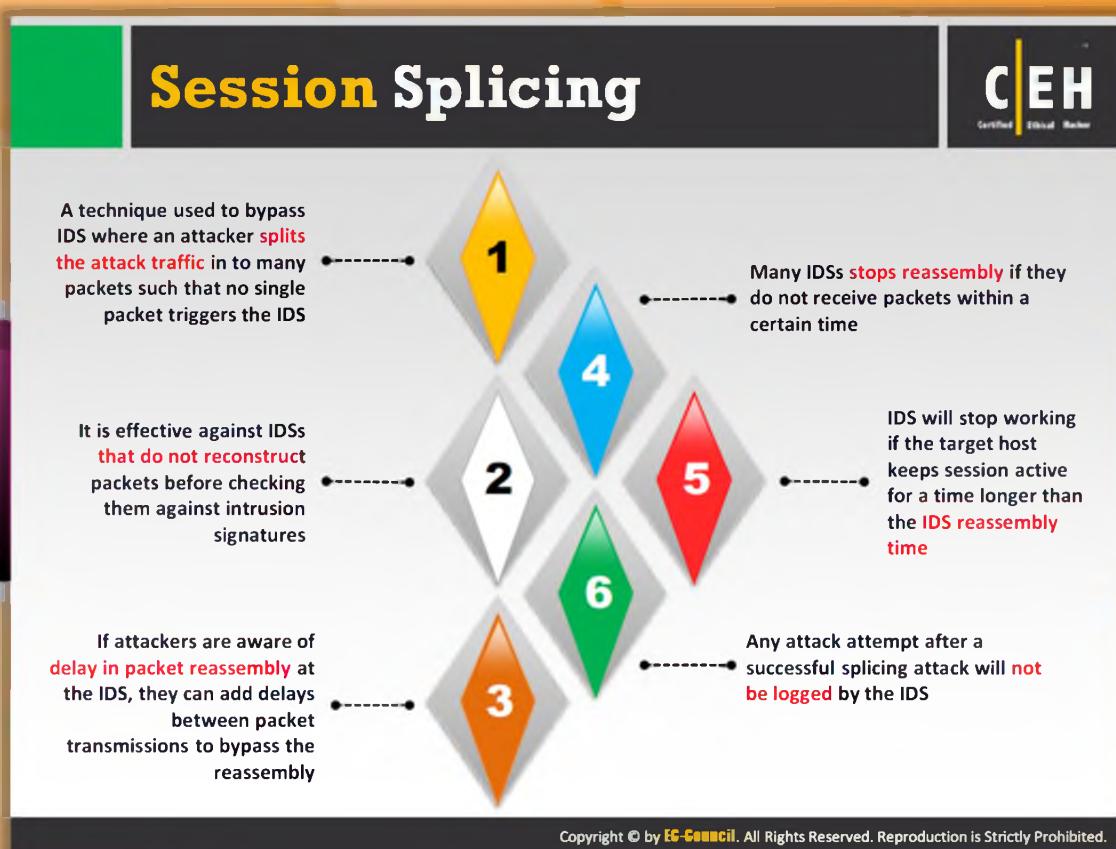
Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



False Positive Generation

This mode **does not attack** the target, but instead, it does something relatively normal. In this mode, an alarm is generated when no condition is present to warrant one. However, many IDSes falsely trigger on this.

Another attack similar to the **DoS method** is to generate a **large amount** of alert data that must be logged. Attackers craft packets known to trigger alerts within the IDS, forcing it to generate a large number of false reports. This type of attack is designed to create a great deal of log "noise" in an attempt to blend real attacks with the false. Attackers know all too well that when looking at log data, it can be very difficult to differentiate between legitimate attacks and false positives. If attackers have knowledge of the IDS system, they can even **generate false positives** specific to that IDS.



Session Splicing

Session splicing is an **IDS evasion technique** that exploits how some IDSSes do not reconstruct sessions before performing pattern matching on the data. It is a network-level evasion method that divides the string across **several packets**. The data in the packets is divided into small portions of bytes and while delivering the string match is evaded. It is used by an attacker to deliver the data into several small sized packets. IDS can't handle too many small sized packets and fails to detect the **attack signatures**. If attackers know what IDS system is in use, they could add delays between packets to bypass reassembly checking. Many IDSSes reassemble **communication streams**, so if a packet is not received within a reasonable amount of time, many IDSSes stop reassembling and handling that stream. If the application under attack keeps a session active longer than an IDS will spend on reassembling it, the IDS will stop. As a result, any session after the IDS stops reassembling the sessions will be susceptible to malicious data theft by the attacker. Different tools such as Nessus, Whisker, etc. are used for session splicing attacks.

Unicode Evasion Technique

C|EH
Certified Ethical Hacker

- ➊ Unicode is a character coding system to support the worldwide interchange, processing, and display of the written texts
- ➋ For Example, / → %u2215, e → %u00e9 (UTF-16) and © → %c2%a9, ≠ → %e2%89%a0 (UTF-8)
- ➌ Attackers can convert attack strings to Unicode characters to avoid pattern and signature matching at the IDS
- ➍ Attackers can encode URLs in HTTP requests using Unicode characters to bypass HTTP-based attack detection at the IDS



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Unicode Evasion Technique

Unicode is a character representation that gives each character a **unique identifier** for each written language to facilitate the uniform computer representation of each language. This is problematic for IDS technology because it is possible to have multiple representations of a single character.

For example, '\' can be represented as **5C**, **C19C** and **E0819C**, which makes writing pattern matching signatures very difficult.

Example for how Unicode affects IDS:

- ➊ **Microsoft IIS 4.0/5.0** Directory Traversal vulnerability released in October 2000 by Rain Forrest Puppy
- ➋ This IIS vulnerability improperly restricts directory listings that were Unicode encoded within the URL request
- ➌ This allowed remote attackers to view files on the IIS server that they normally would not be permitted to see

Fragmentation Attack

Fragmentation can be used as an attack vector when **fragmentation timeouts** vary between IDS and host

- If fragment reassembly timeout is **10 seconds** at the IDS and **20 seconds** at the target system, attackers will send the second fragment after **15 seconds** of sending the first fragment
- In this scenario, the IDS will **drop the fragment** as the second fragment is received after its reassembly time but the target system will reassemble the fragments
- Attackers will keep sending the fragments with **15 second delays** until all the attack payload is reassembled at the target system

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

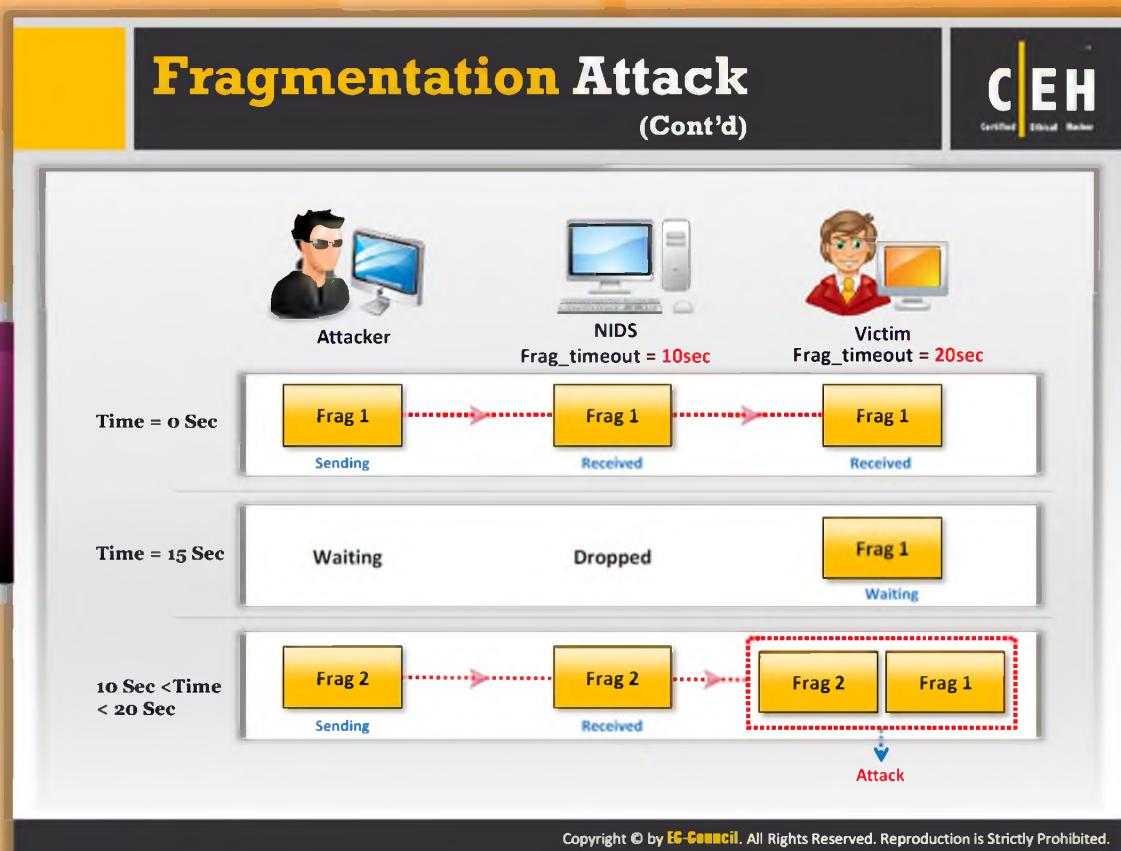


Fragmentation Attack

Attackers break the single Internet protocol datagram into multiple packets of smaller size. IDS fragmentation reassembly timeout is less than fragmentation reassembly timeout of the victim.

Attack Scenario:

Assume the IDS fragmentation reassembly timeout is 15 seconds and the system is monitoring Linux hosts that have default fragmentation reassembly timeout of 30 seconds. After sending the first fragment, the attacker can send the second fragment with a delay of 15 seconds but still within 30 seconds. Now, the victim reassembles the fragments whereas at the IDS the fragmentation reassembly timeout parameter kicks in and the time out occurs. The second fragment received by the IDS will be dropped as the IDS has already lost the first fragment, due to time out. Thus, the victim will reassemble the fragments and will receive the attack whereas the IDS will not make any noise or generate alerts.



Fragmentation Attack (Cont'd)

The following figure illustrates the attack where the NIDS fragmentation re-assemble timeout is less than the victim's fragmentation reassembly timeout.

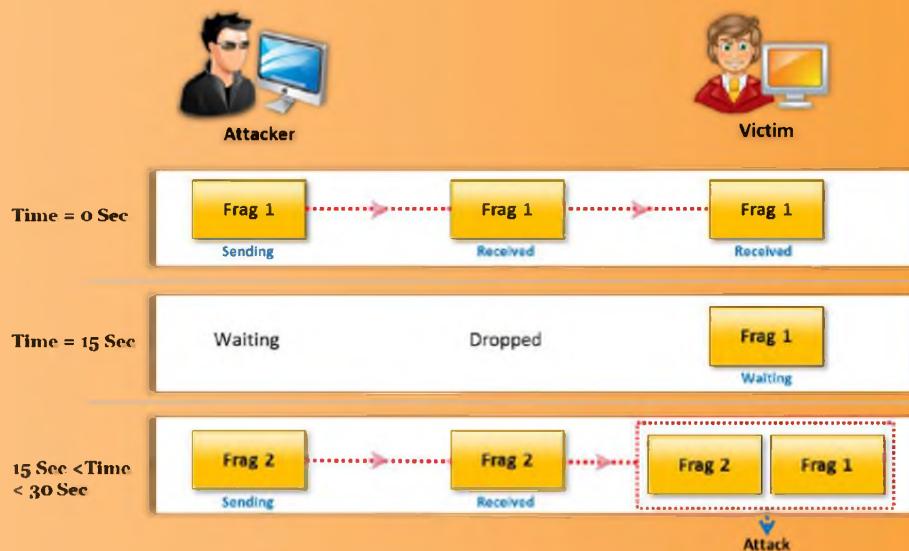
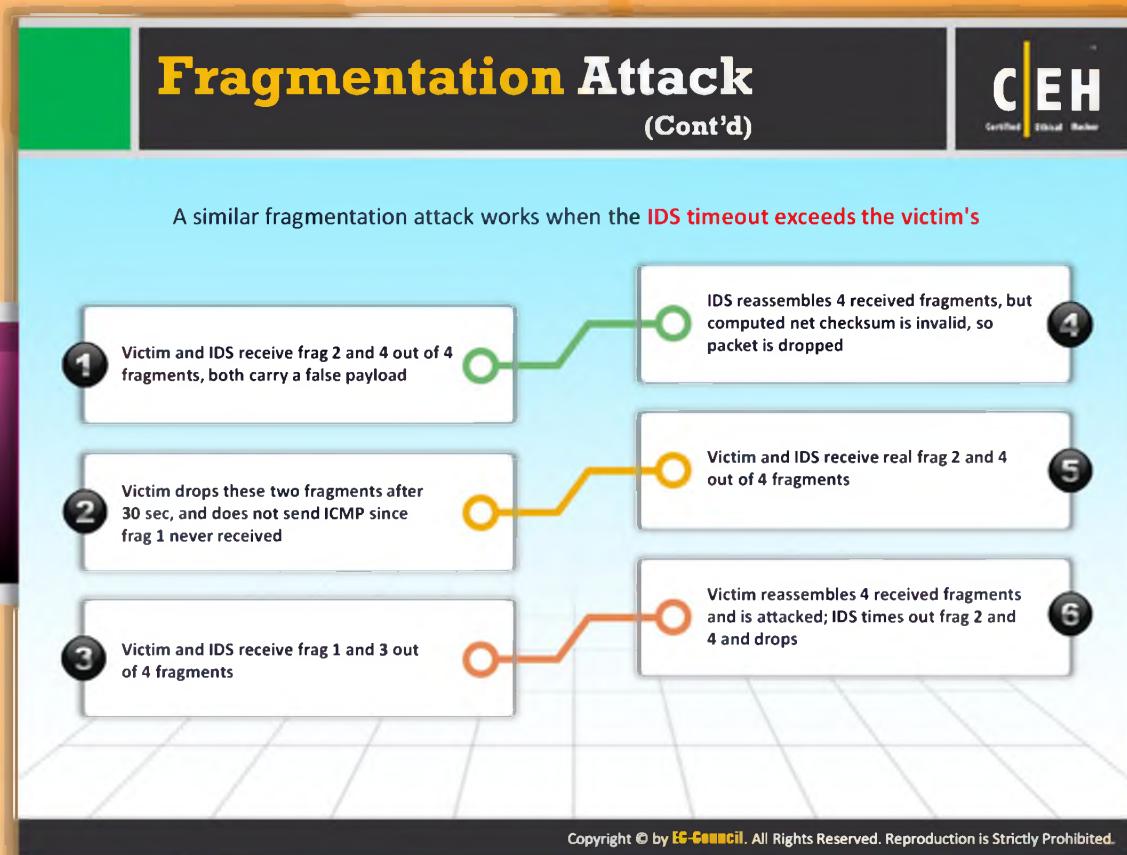
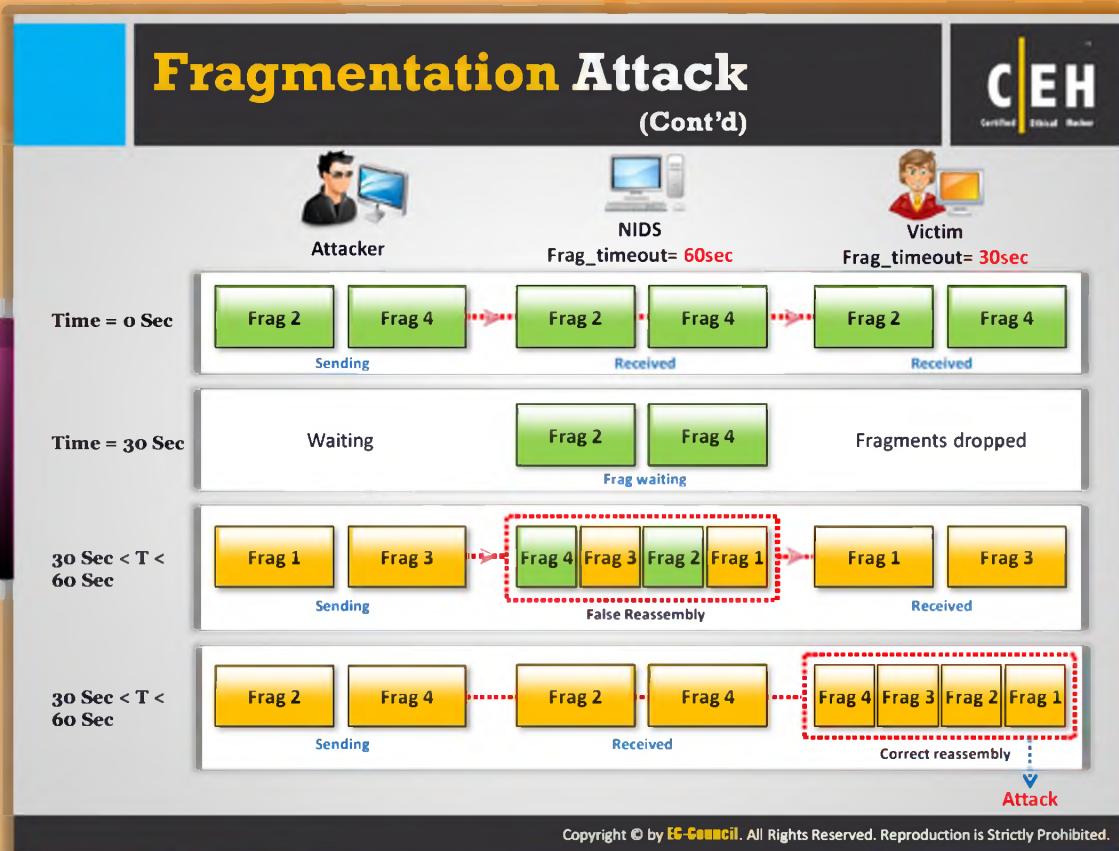


FIGURE 17.23: NIDS Fragmentation Re-assemble Part 1



Fragmentation Attack (Cont'd)

An attacker has fragmented the attack packet into four segments: 1, 2, 3, and 4, and sends frag2 and frag4 with a false payload (referred as 2', 4'), which are received by both the victim and the IDS. The victim waits until the fragments' reassembly timeout occurs at the victim's end and it drops the initial fragments (30 seconds in this case). The victim still has not received fragment 1, so it will quietly drop the fragments and no ICMP error message will be thrown by the victim. The attacker then sends packets (1, 3) with legitimate payloads. At this stage, the victim has only fragments (1, 3), whereas the IDS has fragments (1, 2', 3, 4') in that 2, 4 fragments sent by attacker have a false payload. Since the IDS has all the four fragments it will do a TCP reassembly. Also, since fragments 2 and 4 have false payloads, the net checksum computed will be invalid. So, the IDS will drop the packet. If the attacker now sends fragments 2, 4 again with valid payload, the IDS will have only these two fragments, whereas the victim will have all (1, 3, 2, 4) fragments all with a valid payload, and it will do a reassembly and read the packet as an attack.



Fragmentation Attack (Cont'd)

The following figure illustrates an attack where the NIDS fragmentation reassembly timeout is more than the victim's fragmentation reassembly timeout.

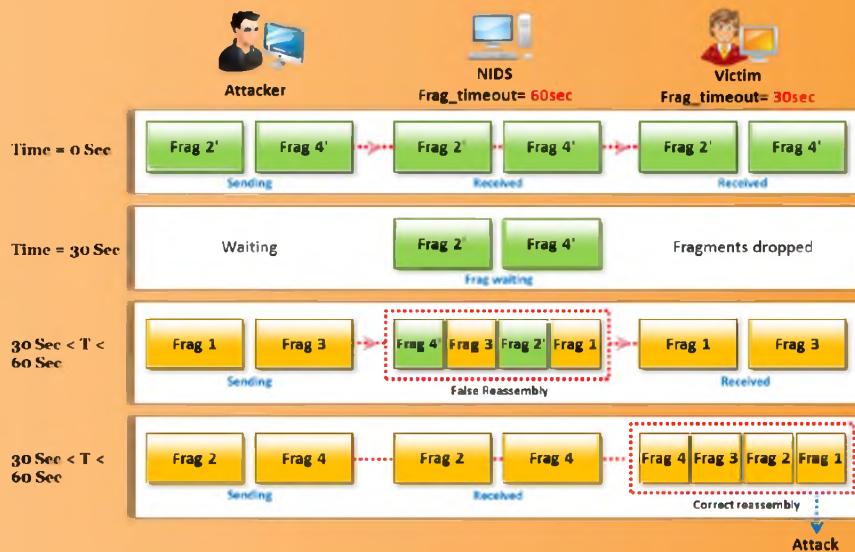


FIGURE 17.24: NIDS Fragmentation Re-assembly Part 2

Overlapping Fragments

C|EH
Certified Ethical Hacker

- An IDS evasion technique is to **craft a series of packets** with TCP sequence numbers configured to overlap
- For example, the first packet will include **80 bytes** of payload, but the second packet's sequence number will be **76 bytes** after the start of the first packet
- When the target computer **reassembles the TCP stream**, it must decide how to handle the four overlapping bytes
- Some operating systems will take the **original fragments with a given offset** (e.g., Windows W2K/XP/2003) and some operating systems will take the subsequent fragments with a given offset (e.g., Cisco IOS)

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Overlapping Fragments

Source: <http://books.google.co.in>

An IDS evasion technique is to craft a series of packets with **TCP sequence** numbers configured to overlap. In an **overlapping fragment attack**, the packets start in the middle of another packet. For example, the first packet can include **80 bytes** of payload, but the second packet's sequence number can be **76 bytes** after the start of the first packet. When the target computer reassembles the TCP stream, it must decide how to handle the four overlapping bytes. Some operating systems can take the original fragments with a given offset (e.g., Windows W2K/XP/2003) and some operating systems can take the subsequent fragments with a given offset (e.g., Cisco IOS).

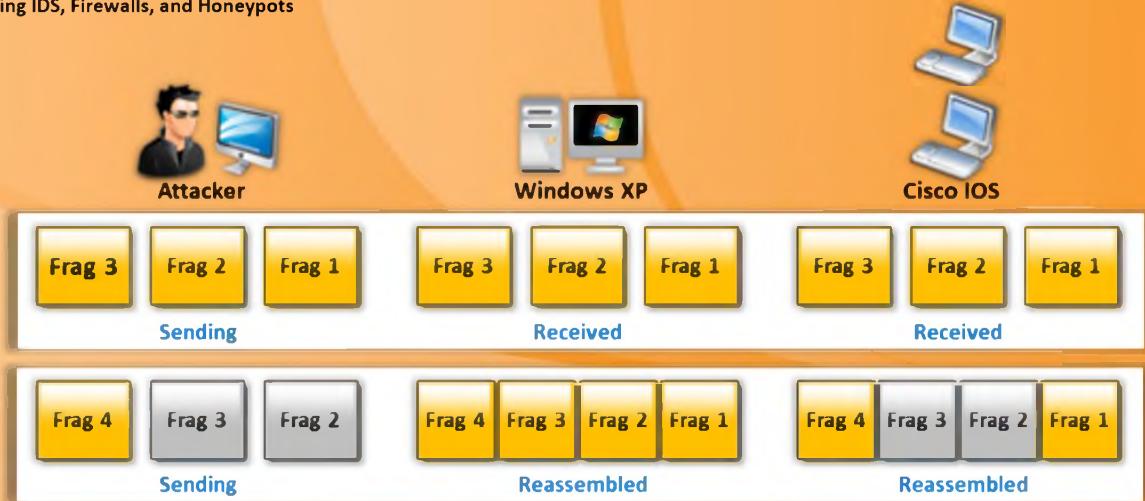


FIGURE 17.25: Working of Overlapping Fragments

Time-To-Live Attacks

C|EH
Certified Ethical Hacker

- These attacks require the attacker to have a **prior knowledge of the topology** of the victim's network
- This information can be obtained using tools such as **traceroute** which gives information on the **number of routers between the attacker and the victim**

- Attacker breaks malicious traffic into **3 fragments**
- Attacker sends frag 1 with **high TTL**, false frag 2 with low TTL
- IDS receives both fragments, victim receives **first fragment only**
- Attacker sends **frag 3 with high TTL**
- IDS reassembles 3 fragments into meaningless packet and **drops**
- Victim receives real frag 2, and **suffers attack**, while no log entry created

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



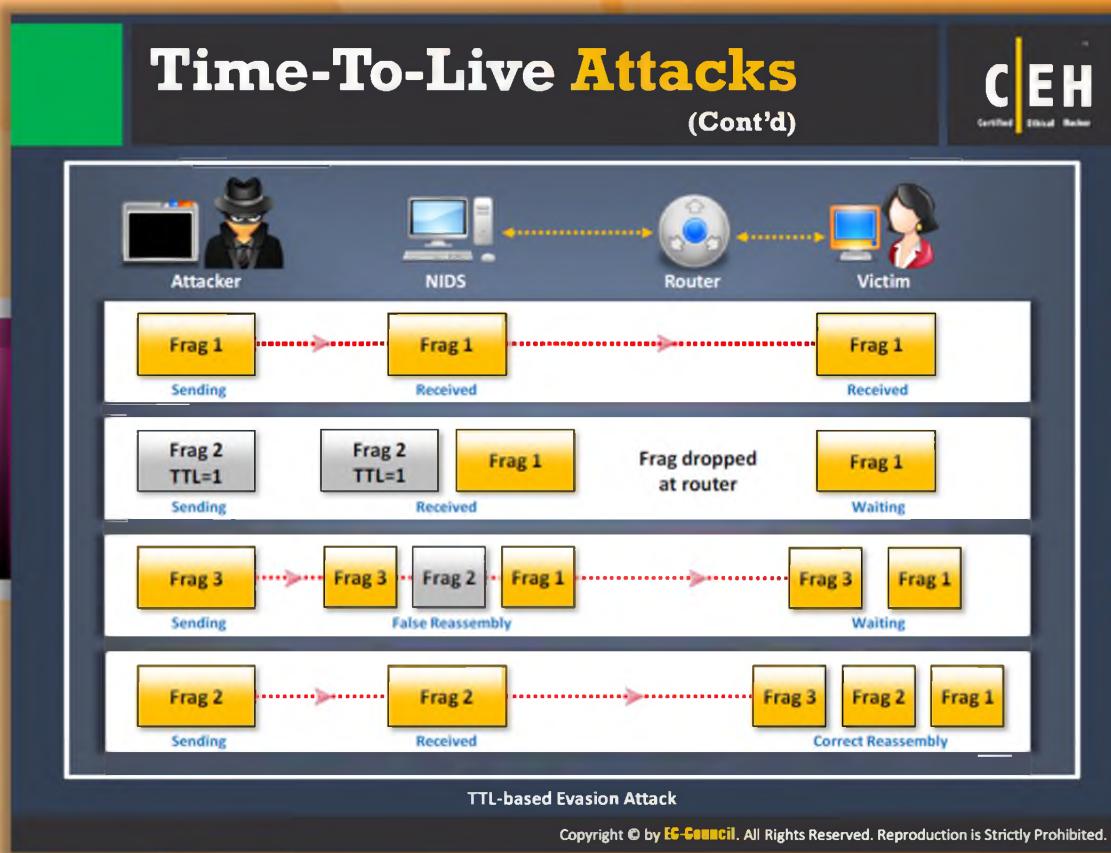
Time-To-Live Attacks

Source: <http://www.scribd.com>

Each IP packet has a field called **Time to Live (TTL)**, which indicates how many more hops the packet should be allowed to make before being discarded or returned. Each router along a data path decrements this value, by one. When a router decrements this value to zero, it drops the packet and sends an ICMP alert notification. Typically, when a host sends a packet, it sets the TTL to a value high enough that the packet can reach its destination under normal circumstances. Different operating systems use different default initial values for the TTL. Because of this an attacker can guess the number of routers between itself and a sending machine, and make assumptions on what the initial TTL was, thereby guessing which OS a host is running, as prelude to an attack. In order to prevent such detection, SmartDefense can change the TTL field of all packets (or all outgoing packets) to a given number.

A router is present between the IDS and a victim - and the attacker is assumed to have this prior information and carries out the attack by breaking it into three fragments. Attacker sends fragment 1 with a large TTL value, which is received by both the IDS and the victim and then sends second fragment (frag2') with the TTL value of 1 and false payload. This fragment is received by the IDS, whereas the router (which is situated between the IDS and the victim) discards it as the TTL value is now reduced to zero. At this stage, the IDS has only fragment 2 as

it has already performed a reassembly and the stream has been flushed. The attacker finally sends the second fragment with a valid payload and the victim performs a reassembly on fragments (1, 2, 3) and gets the attack. The attacker then sends fragment 3 with a valid TTL. This makes the IDS perform a TCP-reassembly on fragments (1, 2', 3), whereas the victim still waits for the second fragment.



Time-To-Live Attacks (Cont'd)

The following figure illustrates the Time-to-Live attack, a TTL-based evasion attack:

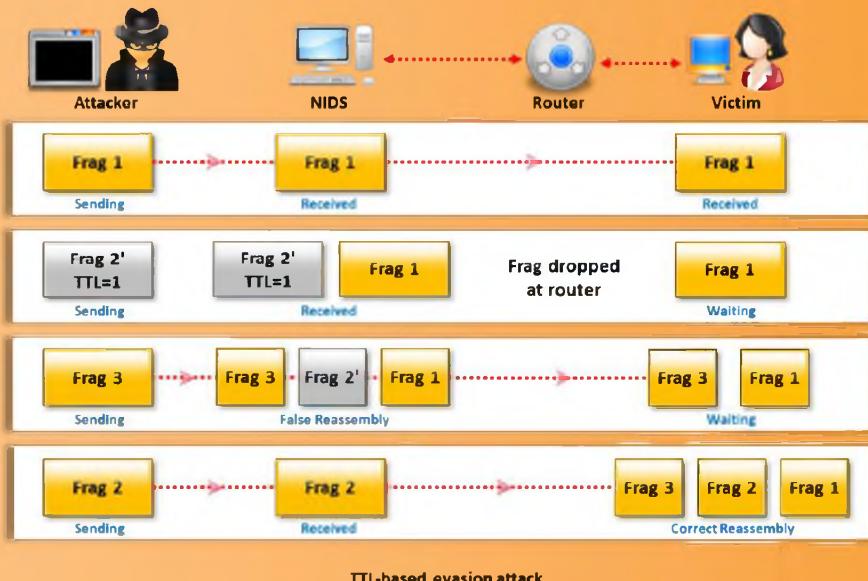


FIGURE 17.26: Time-To-Live Attacks

Invalid RST Packets

C|EH
Certified Ethical Hacker

- 1 TCP uses 16-bit checksum field for **error-checking** of the header and data
- 2 **Reset (RST)** flag in a TCP header is used to close a TCP connection
- 3 In invalid reset attack, attackers **send RST packet** to the IDS with an invalid checksum
- 4 IDS stop processing the packet thinking that the **TCP communication session** has ended but the target system will receive the packet
- 5 The target system **checks the RST packet's checksum** and drops it
- 6 The attack enables **attackers to communicate** with the target system while the IDS thinks that the communication has ended



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Invalid RST Packets

The **TCP protocol** uses **checksums** to ensure that communication is reliable. A checksum is added to every **transmitted segment** and it is checked at the receiving end. When a checksum differs from the checksum expected by the receiving host, the packet is dropped at the receiver's end. The TCP protocol also uses an **RST packet** to end two-way communications. Attackers can use this feature to elude detection by sending RST packets with an invalid checksum, which causes the IDS to stop processing the stream because the **IDS** thinks the communication session has ended. However, the end host sees this packet and verifies the checksum value, then drops the packet if it is invalid.

Some IDS systems might interpret this packet as an actual termination of the communication and stop reassembling the communication. Such instances allow attackers to continue to communicate with the end host while confusing the IDS because the end host accepts the packets that follow the RST packet with an invalid checksum value.

Urgency Flag

The Urgent (URG) flag in the TCP header is used to mark the data that require urgent processing at the receiving end.

If the URG flag is set, the TCP protocol sets the Urgent Pointer field to a 16-bit offset value that points to the last byte of urgent data in the segment.

Many IDSs do not consider the urgent pointer and process all the packets in the traffic whereas the target system processes only the urgent data.

This results in the IDS and the target systems having different set of packets, which can be exploited by attackers to pass the attack traffic.

Urgency flag attack example:
"1 Byte data, next to Urgent data, will be lost, when Urgent data and normal data are combined."
Packet 1: ABC
Packet 2: DEF Urgency Pointer: 3
Packet 3: GHI
End result: ABCDEFHI

This example illustrates how the urgency flag works in conjunction with the urgency pointer.
According to the RFC 1122, the urgency pointer causes one byte of data next to the urgent data to be lost when urgent data is combined with normal data.



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Urgency Flag

The urgency flag is used within the **TCP protocol** to mark data as urgent. TCP uses an **urgency pointer**. That points to the beginning of urgent data within a packet. When the urgency flag is set, all data before the urgency pointer is ignored, and the data to which the urgency pointer points is processed. Some IDSes do not take into account the TCP protocol's urgency feature, which could allow attackers to evade the IDS, as seen in other **evasion techniques**. Attackers can place garbage data before the urgency. The pointer and the IDS read that data without consideration for the end host's **urgency flag handling**. This means the IDSes have more data than the end host actually processed.

Urgency flag attack example:

"1 Byte data, next to Urgent data, can be lost, when Urgent data and normal data are combined."

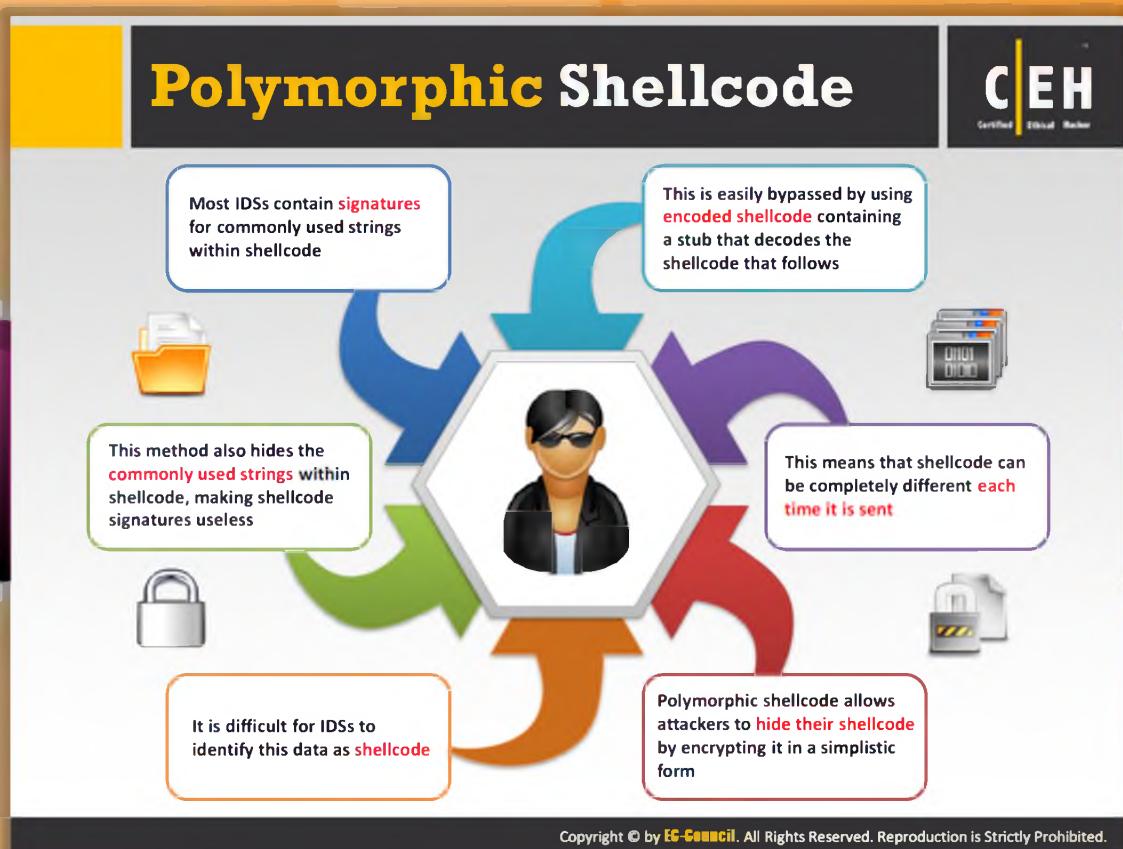
Packet 1: ABC

Packet 2: DEF Urgency Pointer: 3

Packet 3: GHI

End result: ABCDEFHI

This example illustrates how the urgency flag works in conjunction with the urgency pointer. According to the [1122 RFC](#), the urgency pointer causes one byte of data next to the urgent data to be lost when urgent data is combined with normal data.



Polymorphic Shellcode

Most IDSes contain **signatures** for commonly used strings within shellcode. This is easily bypassed by using **encoded shellcode** containing a stub that decodes the shellcode that follows. This means that shellcode can be completely different each time it is sent. Polymorphic shellcode allows attackers to **hide their shellcode** by encrypting it in a simplistic form. It is difficult for IDSs to identify this data as shellcode. This method also hides the commonly used strings within shellcode, making shellcode signatures useless.

ASCII Shellcode



The following is an ASCII shellcode example:

```
char shellcode[] =  
"LLLLYhb0pLX5b0pLHSSPPWQPPaPWSUTBRDJfh5tDS"  
"RajYX0Dka0TkafhN9fYf1Lkb0Tkdfjfy0Lkf0Tkgfh"  
"6rfYf1Lki0tkkh95h8Y1LkmjpY0Lkq0tkrh2wnuX1"  
"Dks0tkwjfx0Dkx0tkx0tkyCjnY0LkzC0TkzCCjtX0"  
"DkzC0tkzCj3X0Dkz0TkzC0tkzChjG3IY1LkzCC  
CCC"  
"tkzChpfcMX1DkzCCCC0tkzCh4pCnY1Lkz1TkzC  
CCC"  
"fhJGfxF1Dkzf1tkzCCjHX0DkzCCCCjvY0LkzCC  
Cjd"  
"X0DkzC0TkzCjWX0Dkz0TkzCjdX0DkzCjXY0Lkz  
0tk"  
"zMdgvvn9F1r8F55h8pg9wnuvjrNfrVx2LGkG3I  
Dpf"  
"cM2KgmnJGgbinYshdvD9d";
```

When executed, the shellcode above executes a "/bin/sh" shell. 'bin' and 'sh' are contained in the last few bytes of the shellcode.

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



ASCII Shellcode

ASCII shellcode contains **only characters** contained within the ASCII standard. This form of shellcode allows attackers to bypass commonly enforced character restrictions within string input code. It also helps attackers bypass IDS pattern matching signatures because strings are hidden within the shellcode in a similar fashion to polymorphic shellcode.

Using ASCII for shellcode is **very restrictive** in that it limits what the shellcode can do under some circumstances because not all assembly instructions convert directly to ASCII values. This restriction can be **bypassed** using other instructions or a combination of instructions that convert to ASCII character representation, which serves the same purpose of the instructions that improperly convert.

The following is an ASCII shellcode example:

```
char shellcode[] =  
"LLLLYhb0pLX5b0pLHSSPPWQPPaPWSUTBRDJfh5tDS"  
"RajYX0Dka0TkafhN9fYf1Lkb0Tkdfjfy0Lkf0Tkgfh"  
"6rfYf1Lki0tkkh95h8Y1LkmjpY0Lkq0tkrh2wnuX1"  
"Dks0tkwjfx0Dkx0tkx0tkyCjnY0LkzC0TkzCCjtX0"  
"DkzC0tkzCj3X0Dkz0TkzC0tkzChjG3IY1LkzCCCC0"
```

```
"tkzChpfMX1DkzCCCC0tkzCh4pCnY1Lkz1TkzCCCC"  
"fhJGfXf1Dkzf1tkzCCjHX0DkzCCCCjvY0LkzCCCjd"  
"X0DkzC0TkzCjWX0Dkz0TkzCjdX0DkzCjXY0Lkz0tk"  
"zMdgvvn9F1r8F55h8pG9wnuvjrNfrVx2LGkG3IDpf"  
"cM2KgmnJGgbinYshdvD9d";
```

When executed, the shellcode above executes a "/bin/sh" shell. 'bin' and 'sh' are contained in the last few bytes of the shellcode.

Application-Layer Attacks

C|EH
Certified Ethical Hacker

	Applications accessing media files (audio, video and images) compress them to smaller size for maximizing data transfer rate
	IDS cannot verify the signature of compressed file format
	This enables an attacker to exploit the vulnerabilities in compressed data
	IDS can recognize particular conditions favorable for attack but other alternative forms of attack are also possible, for example, various integer values can be used to exploit integer overflow vulnerabilities
	This makes the detection of attack traffic extremely difficult at the IDS

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Application-layer Attacks

In order to **transfer media files** speedily, such as images, audios, videos, the files can be compressed and transferred in smaller parts. Attackers find flaws in this **compressed data** and perform attacks and even IDSees cannot identify the signatures within the compressed data.

Many applications that deal with media such as images, video, and, audio employ some form of compression to be sent in a form much smaller than the original, which **increases data transfer speeds**. When a flaw is found in these applications, the entire attack can occur within compressed data, and the IDS can have no way to check the **compressed file format** for signatures. Many IDSees look for specific conditions that allow for an attack. However, there are times when the attack can take many different forms. For example, integer overflow vulnerabilities could be exploited using several different integer values. This fact combined with compressed data makes signature detection extremely difficult.

Desynchronization - Pre Connection SYN

If a SYN packet is received **after the TCP control block is opened**, the IDS resets the appropriate sequence number to match that of the newly received SYN packet

Attackers send **fake SYN packets** with a completely invalid sequence number to desynchronize the IDS

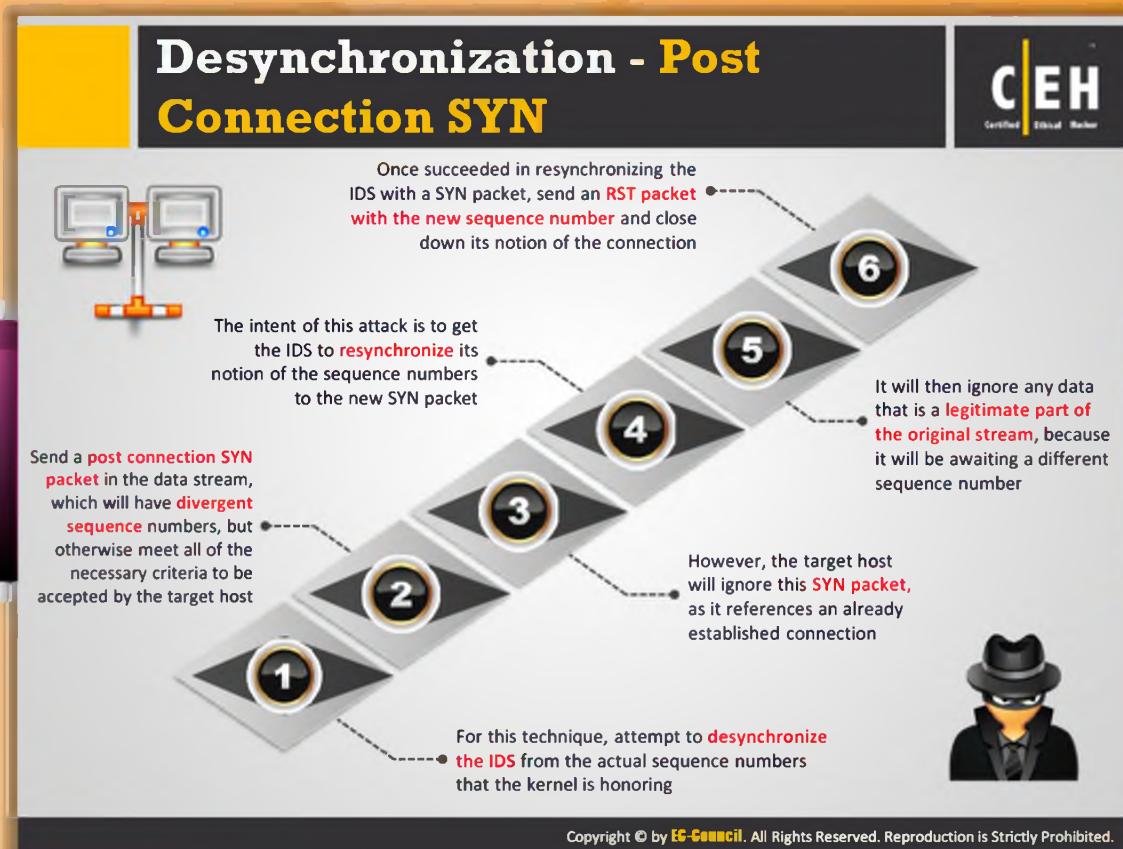
This **stops IDS** from monitoring all, legitimate and attack, traffic

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Desynchronization – Pre Connection SYN

This attack calls bind to get the kernel to assign a local port to the socket before calling connect. This is another attack that an attacker performs and sends an initial SYN before the real connection is established, but with an invalid **TCP checksum**. The sniffer can ignore or accept subsequent SYNs in a connection. If the sniffer is smart, it does not check the TCP checksum; otherwise it checks the **TCP checksum**. If the sniffer checks the checksum, then the attack is synchronized and a bogus sequence number is sent to the **sniffer/IDS** before the real connection occurs.



Desynchronization - Post Connection SYN

To deceive an intelligent sniffer or an ID system, attackers do not directly try to deceive it, for it keeps track of the **TCP sequence numbers**. For this technique to work efficiently, attackers first desynchronize the sniffer or IDS. The attack on the sniffer or IDS can be implemented by sending a post connection **SYN packet** in the data stream. The data stream can have all the necessary sequence numbers (all different) and meet the criteria so that the stream is accepted by the target. After transmitting the data stream, the host ignores the SYN packet, because the reference of the SYN packet has already established connection. The motive behind this attack is to resynchronize the **sniffer/IDS**. If the attacker succeeds in resynchronizing the IDS with a SYN packet, attacker then sends an RST packet with the new sequence number.

Other Types of Evasion

C|EH
Certified Ethical Hacker

Encryption

When the attacker has already established an **encrypted session with the victim**, it results in the most effective evasion attack



Flooding

The attacker sends loads of **unnecessary traffic to produce noise**, and if IDS does not analyze the noise traffic well, then the true attack traffic may go undetected



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Other Types of Evasion

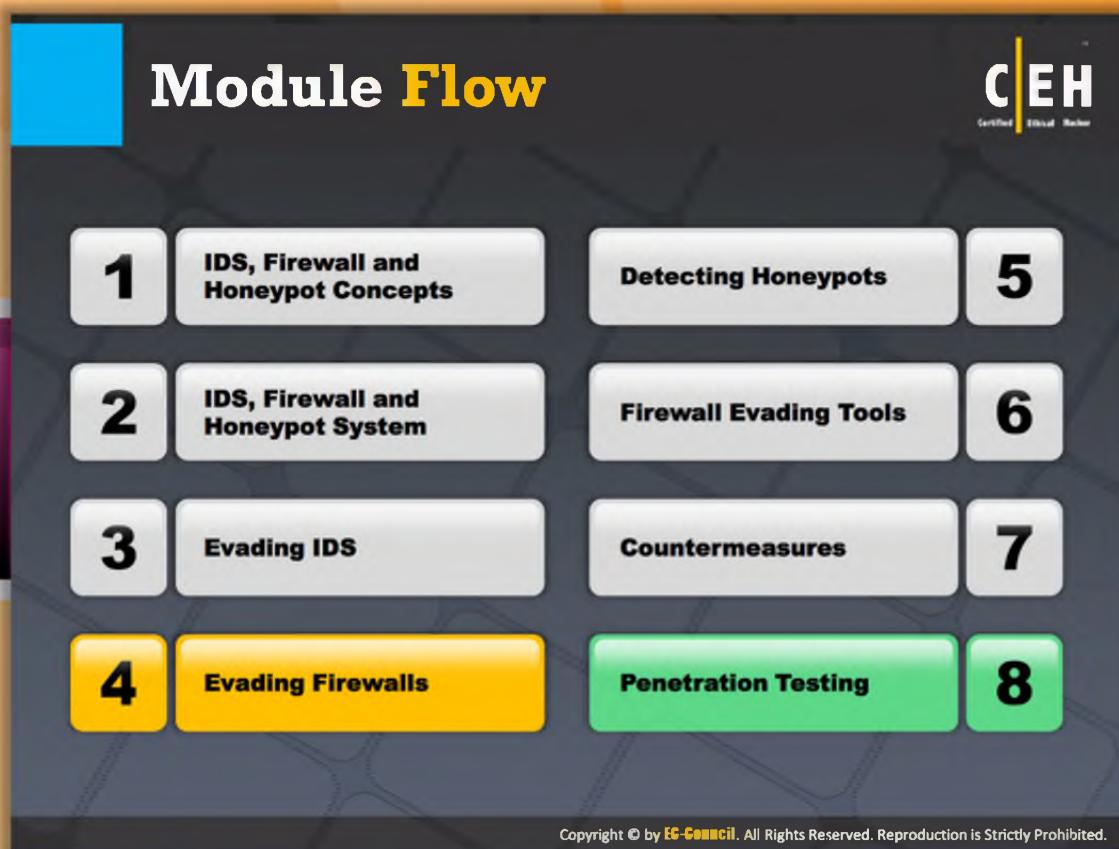
There are two more types of evasion:

Encryption

When the attacker has already established an encrypted session with the victim, it results in the most effective evasion attack.

Flooding

The attacker sends loads of unnecessary traffic to produce noise, and if the IDS do not analyze the noise traffic, the true attack traffic may go undetected.



Module Flow

Firewalls are the security mechanisms implemented by a network or a system to protect itself from being attacked. Attackers try to bypass firewalls so that they can break the security mechanisms and gain access to the legitimate system or network.

 IDS, Firewall and Honeypot Concepts	 Detecting Honeypots
 IDS, Firewall and Honeypot System	 Firewall Evading Tools
 Evading IDS	 Countermeasure
 Evading Firewall	 Penetration Testing

This section describes various ways in which an attacker can evade the firewall.

IP Address Spoofing

C|EH
Certified Ethical Hacker

IP address spoofing is a hijacking technique in which an attacker **masquerades as a trusted host** to conceal his identity, spoof a Web site, hijack browsers, or gain unauthorized access to a network

Attackers modify the **addressing information** in the IP packet header and the source address bits field in order to bypass the firewall

- For example, let's consider **three hosts**: A, B and C
- Host **C is a trusted machine** of host B
- Host A masquerades to be as host C by **modifying the IP address** of the malicious packets that he intends to send to the host B
- When the **packets are received**, host B thinks that they are from host C, but are actually from host A

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



IP Address Spoofing

IP address spoofing or IP spoofing is one of the ways that an attacker tries to evade **firewall restrictions**. IP spoofing is a technique where the **attacker creates Internet protocol** packets by using a forged IP address and gains access over the system or network without any **authorization**. The attacker spoofs the messages and they appear to be sent from a reliable source. Thus, the attacker succeeds in impersonating others' identities with help of IP spoofing. Hackers generally use this technique for not getting caught while spamming and various other activities.

The following scenario shows how an attacker bypasses a firewall by impersonating a different identity with the help of the IP spoofing technique:

- Let's consider three hosts: A, B, and C
- Host C is a trusted machine of host B
- Host A wants to send some packets to host B and A impersonates itself to be C by changing the IP address of these packets
- When these packets are received, B thinks that these packets are from C, but actually they are from A

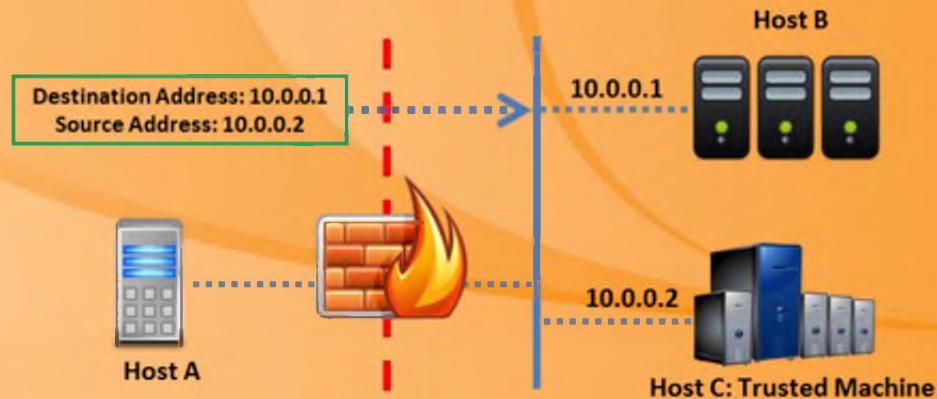


FIGURE 17.27: Working of IP Address Spoofing

Source Routing

The figure shows source routing, where the originator dictates eventual route of traffic

Source routing allows the sender of a packet to partially or completely **specify the route**, the packet takes through the network

As the packet travels through the nodes in the network, each **router examines** the destination IP address and **chooses the next hop** to direct the packet to the destination

In source routing, the **sender** makes some or all of these decisions on the router

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Source Routing

Using this technique, the sender of the **packet designates** the route that a packet should take through the network in such a way that the designated route should bypass the firewall node. Using this technique the attacker can evade the **firewall restrictions**.

When these packets travel through the nodes in the network, each router will check the IP address of the destination and choose the next node to forward them. In source routing, the sender makes some or all of these decisions on the router.

The figure shows the principle of the source routing but it is an optimal way, which makes the decision of the next hop.

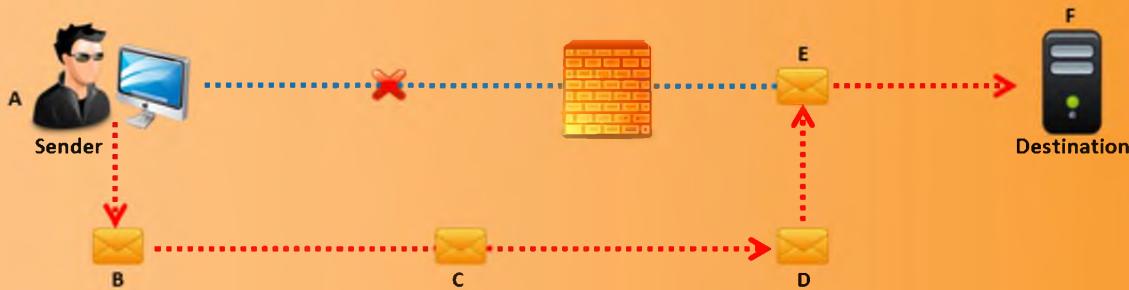


FIGURE 17.28:

Tiny Fragments

The CEH logo is visible in the top right corner.

Attackers create tiny fragments of outgoing packets forcing some of the TCP packet's header information into the next fragment	The attack will succeed if the filtering router examines only the first fragment and allow all the other fragments to pass through
The IDS filter rules that specify patterns will not match with the fragmented packets due to broken header information	This attack is used to avoid user defined filtering rules and works when the firewall checks only for the TCP header information

A detailed diagram of a TCP header is shown:

IP-3ar0J10BOK		MK=1, Fragment Offset=0						
Source Port	Destination Port							
Sequence Number								
Acknowledgement Sequence Number								
Data Offset	Reserved	-	ACK	-	-	-	Window	
Checksum		Urgent Pointer=0						
0								

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

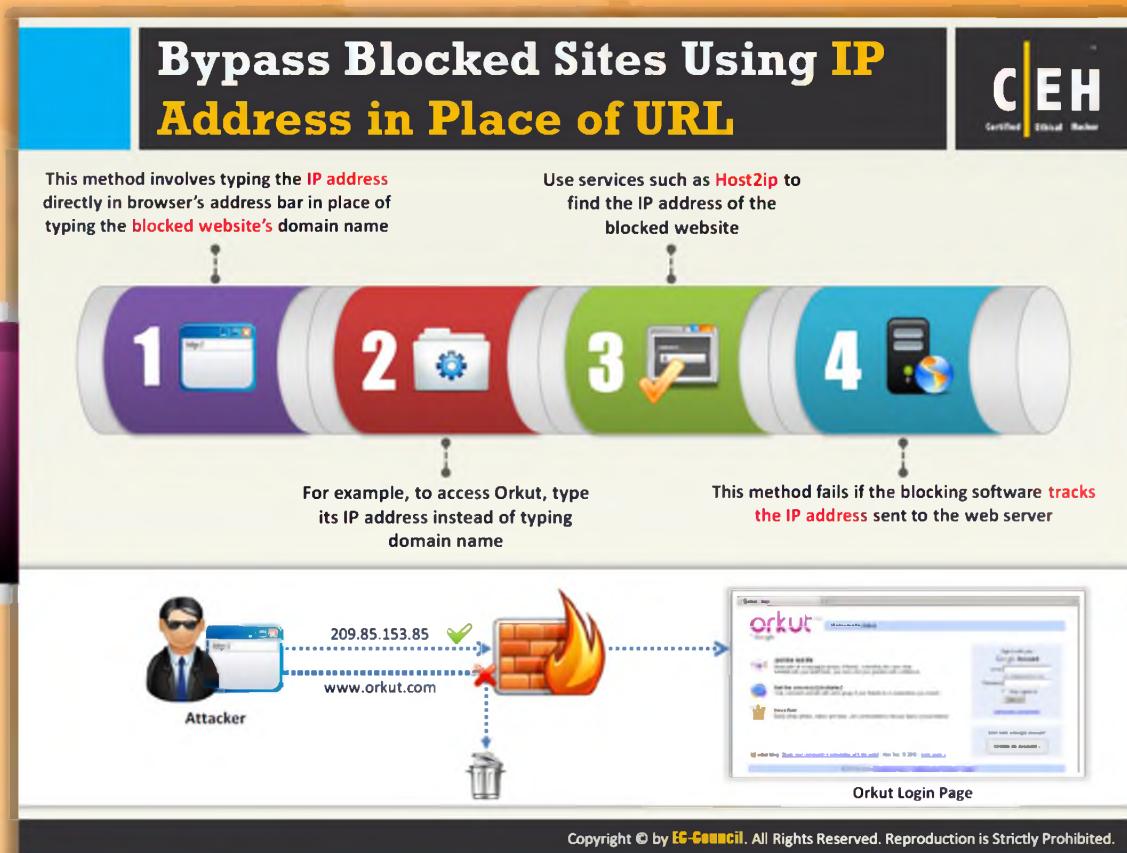
Tiny Fragments

The attacker uses the **IP fragmentation technique** to create extremely small fragments and force the **TCP header** information into the next fragment. This may result in a case whereby the TCP flags field is forced into the second fragment, and filters will be unable to check these flags in the first octet thus ignoring them in **subsequent fragments**.

Attackers hope that only the first fragment is examined by the **filtering router (firewall)** and the remaining fragments are passed through. This attack is used to avoid user defined filtering rules and works when the firewall checks only for the TCP header information.

IP-3ar0J10B0K		MK=1, Fragment Offset=0													
Source Port		Destination Port													
Sequence Number															
Acknowledgement Sequence Number															
Data Offset	Reserved	-	ACK	-	-	-	-	Window							
Checksum						Urgent Pointer=0									
0															

FIGURE 17.29: Tiny Fragments Diagram



Bypass Blocked Sites Using IP Address in Place of URL

You can also evade **firewall restrictions** by typing the IP address of the **blocked site** instead of its domain names. This allows you to access the restricted or blocked sites. You need to use some tools to convert the **target domain name** into its IP address.

For example:

- Instead of typing www.Orkut.com, type its IP address to access Orkut
- Host2ip can help you to find the IP address of that blocked website
- If the blocking software can track the IP address sent to the web server, the website could not be unblocked or accessed by using this method

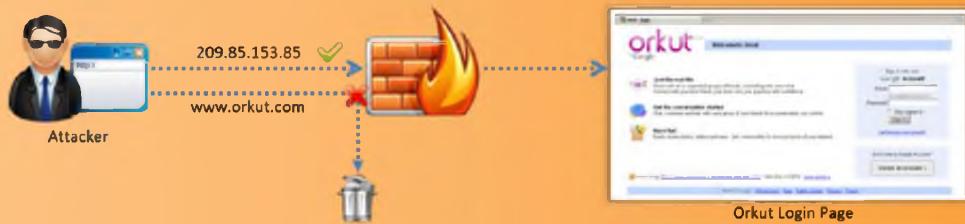
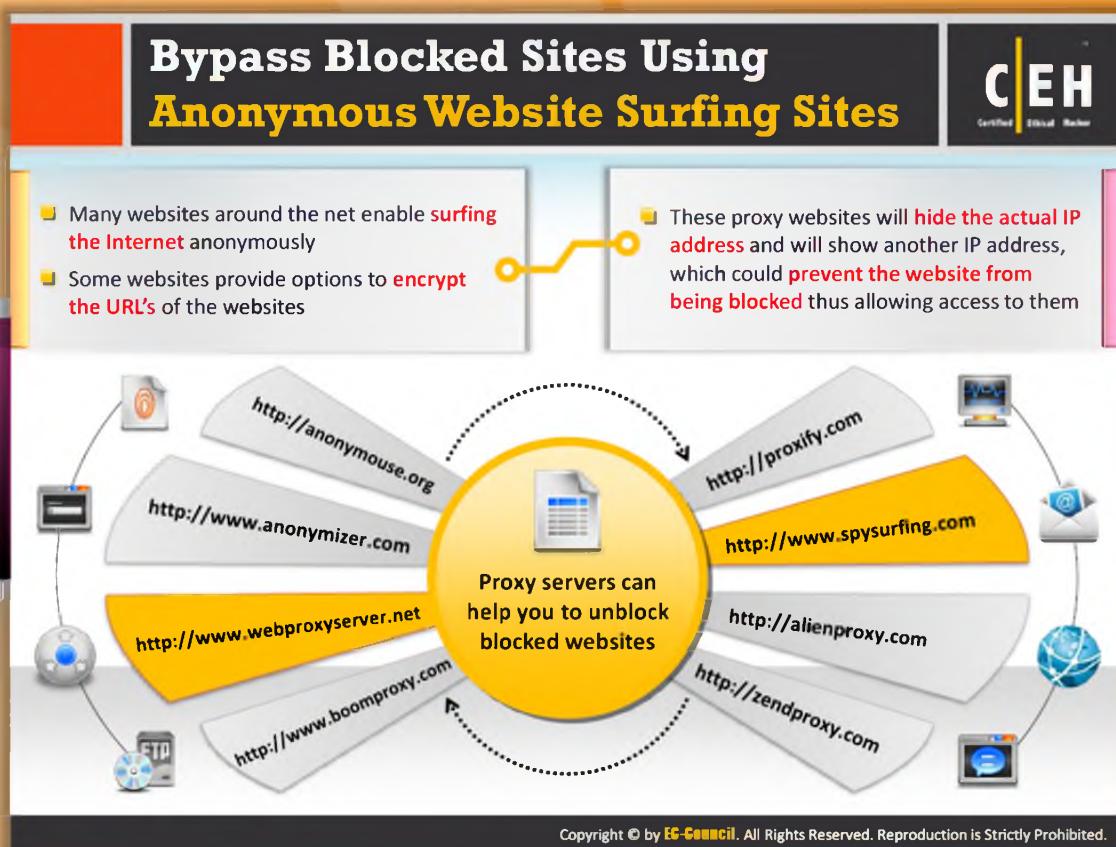


FIGURE 17.30: Bypass Blocked Sites Using IP Address in Place of URL



Bypass Blocked Sites Using Anonymous Website Surfing Sites

Anonymous website surfing sites help you to surf the Internet anonymously and to unblock blocked sites. i.e., evade **firewall restrictions**. By using these sites, you can surf restricted sites anonymously, i.e., without using your IP address on the Internet. There are a number of anonymous **website surfing sites** available on the Internet. Some websites provide options to encrypt the URLs of websites.

Here is a list of some of the proxy servers that can help you to unblock blocked websites:

- ④ <http://anonymouse.org>
- ④ <http://www.anonymizer.com>
- ④ <http://www.webproxyserver.net>
- ④ <http://www.boomproxy.com>
- ④ <http://proxy.com>
- ④ <http://www.spysurfing.com>
- ④ <http://alienproxy.com>
- ④ <http://zendproxy.com>

Bypass a Firewall Using Proxy Server

C|EH
Certified Ethical Hacker

 Find an appropriate proxy server	 In the Port box, type the port number that is used by the proxy server for client connections (by default, 8080)
 On the Tools menu of any Internet browser, go to LAN or Network Connections tab, and then click LAN/Network Settings	 Click to select the bypass proxy server for local addresses check box if you do not want the proxy server computer to be used when connected to a computer on the local network
 Under Proxy server settings, select the use a proxy server for LAN	 Click OK to close the LAN Settings dialog box
 In the Address box, type the IP address of the proxy server	 Click OK again to close the Internet Options dialog box

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Bypass a Firewall Using a Proxy Server

By using a proxy server, you can also bypass the firewall restriction imposed by a particular organization. To evade the firewall restrictions using a proxy server, follow these steps:

1. Find an appropriate proxy server.
2. On the **Tools** menu of any Internet browser, go to LAN or Network Connections tab, and then click **LAN/Network Settings**.
3. Under **Proxy server settings**, select the use a proxy server for the LAN.
4. In the **Address** text box, type the IP address of the proxy server.
5. In the **Port** text box, type the port number that is used by the proxy server for client connections (by default, 8080).
6. Click to select the bypass proxy server for local addresses check box if you do not want the proxy server computer to be used when connected to a computer on the local network.
7. Click **OK** to close the **LAN Settings** dialog box.
8. Click **OK** again to close the **Internet Options** dialog box.

Bypassing Firewall through ICMP Tunneling Method

C|EH
Certified Ethical Hacker

The diagram shows three main components: an Attacker (represented by a person in a suit and sunglasses), a Firewall (represented by a brick wall with a flame), and an Internet Client (represented by a woman at a laptop). A blue dashed arrow points from the Attacker to the Firewall, labeled "Wraps evil client command in ICMP Echo packet". Another blue dashed arrow points from the Firewall to the Internet Client, labeled "Unwraps command, executes it, locally wraps output in ICMP Echo Packet, and resends back to attacker".

It allows tunneling a backdoor shell in the data portion of ICMP Echo packets

RFC 792, which delineates ICMP operation, does not define what should go in the data portion

The payload portion is arbitrary and is not examined by most of the firewalls, thus any data can be inserted in the payload portion of the ICMP packet, including a backdoor application

Some administrators keep ICMP open on their firewall because it is useful for tools like ping and traceroute

Assuming that ICMP is allowed through a firewall, use Loki ICMP tunneling to execute commands of choice by tunneling them inside the payload of ICMP echo packets

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Bypassing a Firewall through the ICMP Tunneling Method

ICMP tunneling allows tunneling a **backdoor shell** in the data portion of ICMP Echo packets. **RFC 792**, which delineates ICMP operation, does not define what should go in the data portion. The payload portion is arbitrary and is not examined by most of the firewalls, thus any data can be inserted in the payload portion of the ICMP packet, including a **backdoor application**. Some administrators keep ICMP open on their firewall because it is useful for tools like ping and traceroute. Assuming that ICMP is allowed through a firewall, use **Loki ICMP tunneling** to execute commands of choice by tunneling them inside the payload of ICMP echo packets



FIGURE 17.31: Bypassing a Firewall through the ICMP Tunneling Method

Bypassing Firewall through ACK Tunneling Method

C|EH
Certified Ethical Hacker

- It allows tunneling a backdoor application with TCP packets with the ACK bit set
- ACK bit is used to acknowledge receipt of a packet
- Some firewalls do not check packets with the ACK bit set because ACK bits are supposed to be used in response to legitimate traffic that is already being allowed through
- Tools such as AckCmd (<http://ntsecurity.nu>) can be used to implement ACK tunneling

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Bypassing a Firewall through the ACK Tunneling Method

ACK tunneling allows tunneling a backdoor application with TCP packets with the ACK bit set. The ACK bit is used to acknowledge receipt of a packet. Some **firewalls** do not check packets with the ACK bit set because **ACK bits** are supposed to be used in response to legitimate traffic that is already being allowed through. Attackers use this as an advantage to perform ACK tunneling. Tools such as **AckCmd (<http://ntsecurity.nu>)** can be used to implement ACK tunneling.



FIGURE 17.32: Bypassing a Firewall through the ACK Tunneling Method

Bypassing Firewall through HTTP Tunneling Method

This method can be implemented if the target company has a public web server with port 80 used for HTTP traffic, that is unfiltered on its firewall

Many firewalls do not examine the payload of an HTTP packet to confirm that it is legitimate HTTP traffic, thus it is possible to tunnel traffic inside TCP port 80 because it is already allowed

Tools such as [HTTP Tunnel](http://www.nocrew.org) (<http://www.nocrew.org>) use this technique of tunneling traffic across TCP port 80

HTTP Tunnel is a client/server application, the client application is called **htc**, and the server is **hts**

Upload the server onto the target system and tell it which port is to be redirected through TCP port 80

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Bypassing a Firewall through the HTTP Tunneling Method

This method can be implemented if the target company has a public web server with port 80 used for HTTP traffic, that is unfiltered on its firewall. Many firewalls do not examine the payload of an HTTP packet to confirm that it is legitimate HTTP traffic, thus it is possible to tunnel traffic inside **TCP port 80** because it is already allowed.

Tools such as [HTTP Tunnel](http://www.nocrew.org) (<http://www.nocrew.org>) use this technique of tunneling traffic across TCP port 80. HTTP Tunnel is a client/server application, the client application is called **htc**, and the server is **hts**. Upload the server onto the target system and tell it which port is to be redirected through TCP port 80.

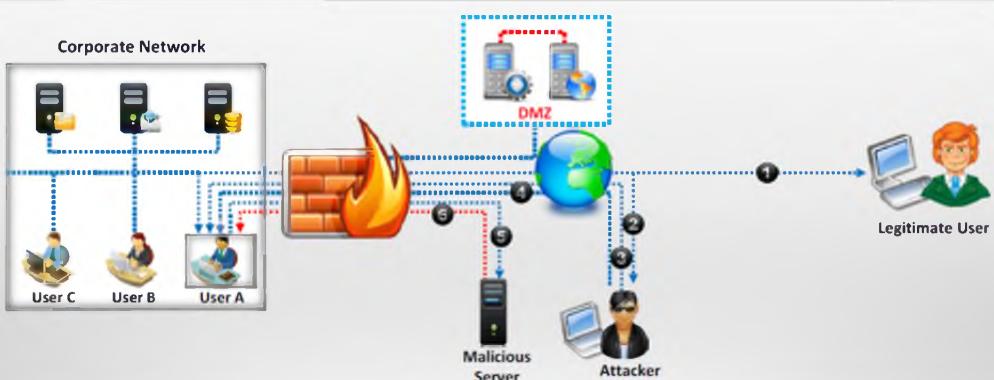


FIGURE 17.33: Bypassing a Firewall through the HTTP Tunneling Method

Bypassing Firewall through External Systems



- Legitimate user works with some **external system** to access the corporate network
- Attacker sniffs the **user traffic**, steals the **session ID** and **cookies**
- Attacker **accesses the corporate network** bypassing the firewall and gets **Windows ID** of the running **Netscape 4.x/ Mozilla** process on user's system
- Attacker then issues an **openURL()** command to the found window
- User's web browser is redirected to the **attacker's Web server**
- The malicious codes embedded in the attacker's web page are **downloaded and executed** on the user's machine



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Bypassing a Firewall through External Systems

Attackers can bypass firewall restrictions through external systems as follows:

1. Legitimate user works with some external system to access the **corporate network**.
2. Attacker sniffs the user traffic, and steals the session ID and cookies.
3. Attacker accesses the corporate network bypassing the firewall and gets Windows ID of the running **Netscape 4.x/ Mozilla** process on user's system.
4. Attacker then issues an **openURL()** command to the found window.
5. User's web browser connects with the attacker's WWW server.
6. Attacker inserts malicious payload into the requested web page (Java applet) and thus the attacker's code gets executed on the user's machine.

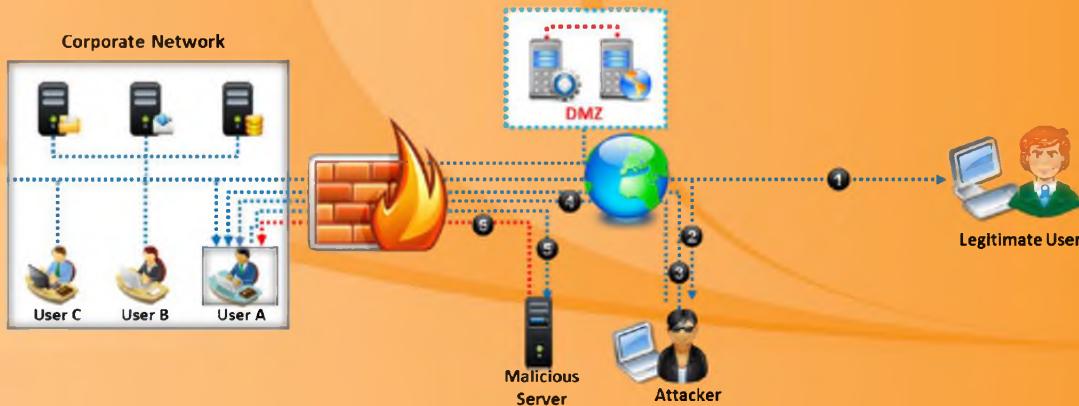


FIGURE 17.34: Bypassing a Firewall through External Systems

Bypassing Firewall through MITM Attack

C|EH
Certified Ethical Hacker

- Attacker performs **DNS server poisoning**
- User A requests for **WWW.juggyboy.com** to the **corporate DNS server**
- Corporate DNS server sends the **IP address (127.22.16.64) of the attacker**
- User A accesses the **attacker's malicious server**
- Attacker connects with the **real host and tunnels the user's HHTP traffic**
- The malicious codes embedded in the attacker's web page are **downloaded and executed** on the user's machine

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Bypassing a Firewall through a MITM Attack

The following steps illustrate an example scenario of how an attacker bypasses a firewall through an MITM attack:

1. Attacker performs **DNS server poisoning**.
2. User A requests **WWW.juggyboy.com** to the corporate DNS server.
3. Corporate DNS server sends the IP address (**127.22.16.64**) of the attacker.
4. User A accesses the attacker's malicious server.
5. Attacker connects with the real host and tunnels the user's HHTP traffic.
6. Attacker inserts malicious payload into the requested web page (Java applet), and thus the attacker's code is executed on the user's machine.

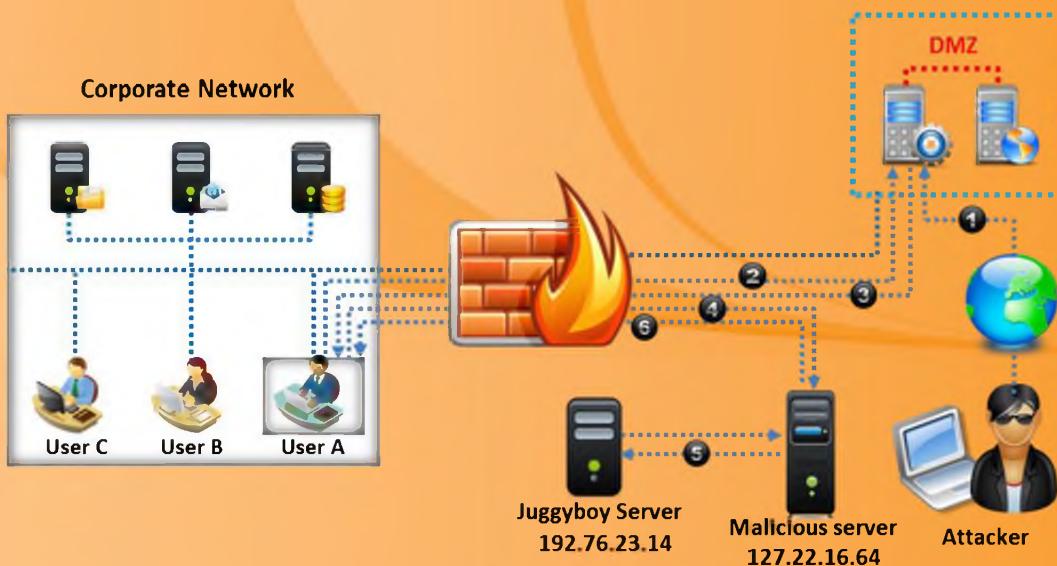
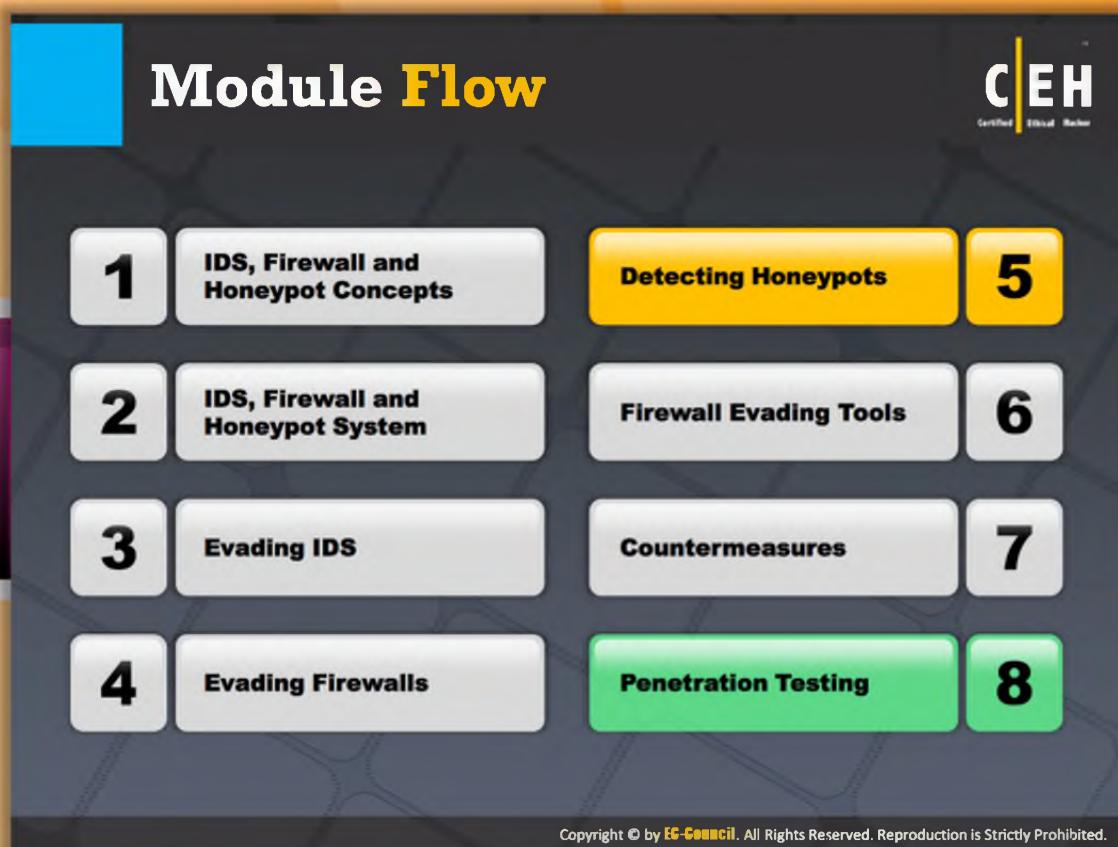


FIGURE 17.35: Bypassing a Firewall through a MITM Attack



Module Flow

Honeypots are the mechanisms intended to track or divert attackers from entering into a genuine network without adequate permissions. Attackers in an attempt to break into the target network first check for honeypots, if any are installed on the target network. Attackers perform honeypot detection to check whether the target network has a honeypot or not.

	IDS, Firewall and Honeypot Concepts		Detecting Honeypots
	IDS, Firewall and Honeypot System		Firewall Evading Tools
	Evading IDS		Countermeasure
	Evading Firewall		Penetration Testing

This section provides insight into honeypot detection and the tools that can be used for detecting honeypots.

Detecting Honeypots

C|EH
Certified Ethical Hacker

The diagram shows a central target icon surrounded by four colored circles (blue, red, orange, purple) connected by curved arrows forming a loop. Step 1 (blue) is 'Attackers can determine the presence of honeypots by probing the services running on the system'. Step 2 (red) is 'Attackers craft malicious probe packets to scan for services such as HTTP over SSL (HTTPS), SMTP over SSL (SMPTS), and IMAP over SSL (IMAPS)'. Step 3 (orange) is 'Ports that show a particular service running but deny a three-way handshake connection indicate the presence of a honeypot'. Step 4 (purple) is 'Tools to probe honeypots:

- Send-safe Honeypot
- Hunter
- Nessus
- Hping

'.

Note: Attackers can also defeat the purpose of honeypots by using multi-proxies (TORs) and hiding their conversation using encryption and steganography techniques

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Detecting Honeypots

A honeypot is a system used on the **Internet designed** especially for diverting the attacker by tricking or attracting him or her when he or she attempts to gain **unauthorized access** to the information system in an organization. Just as honeypots are intended to divert the attackers from actual network, attackers use **honeypot detection systems** or methods to identify the honeypots installed on the target network. Once they detect honeypots, attackers try to bypass them so that they can focus on **targeting the actual network**. Detecting honeypots involves three basic steps:

- Attackers can determine the presence of honeypots by probing the services running on the system.
- Attackers craft malicious probe packets to scan for services such as HTTP over SSL (HTTPS), SMTP over SSL (SMPTS), and IMAP over SSL (IMAPS).
- Ports that show a particular service running but deny a three-way handshake connection indicate the presence of a honeypot.

Different tools such as Send-safe Honeypot, Hunter, Nessus, and Hping can be used for probing honeypots.

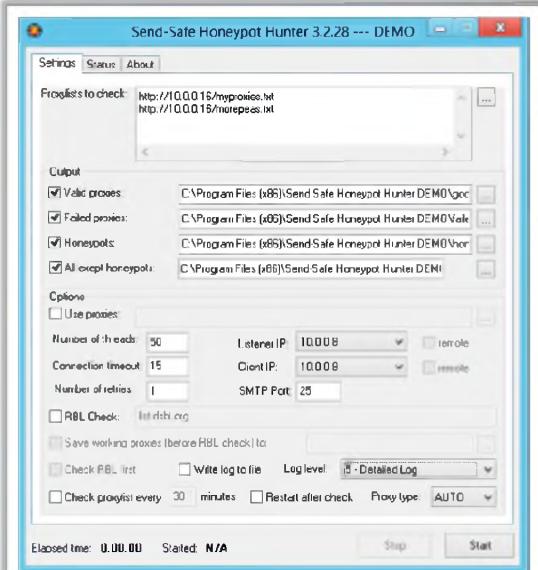
Note: Attackers can also defeat the purpose of honeypots by using **multi-proxies (TORs)** and hiding their conversation using encryption and steganography techniques.

Honeypot Detecting Tool: Send-Safe Honeypot Hunter

Send-Safe Honeypot Hunter is a tool designed for checking lists of HTTPS and SOCKS proxies for "honeypots"

Features:

- Checks lists of **HTTPS, SOCKS4, and SOCKS5 proxies** with any ports
- Checks several remote or local proxylists at once
- Can upload "Valid proxies" and "All except honeypots" files to FTP
- Can process proxylists automatically every specified period of time
- May be used for usual proxylist validating as well



<http://www.send-safe.com>

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Honeypot Detecting Tool: Send-Safe Honeypot Hunter

Source: <http://www.send-safe.com>

Send-Safe Honeypot Hunter is a honeypot detection tool designed for checking lists of HTTPS and **SOCKS proxies** for honeypots.

Some of the Send-Safe Honeypot Hunter features include:

- Checks lists of HTTPS, SOCKS4, and SOCKS5 proxies with any ports
- Can check several remote or local proxylists at once
- Can upload "Valid proxies" and "All except honeypots" files to FTP
- Can process proxylists automatically every specified period of time
- May be used for usual proxylist validating as well

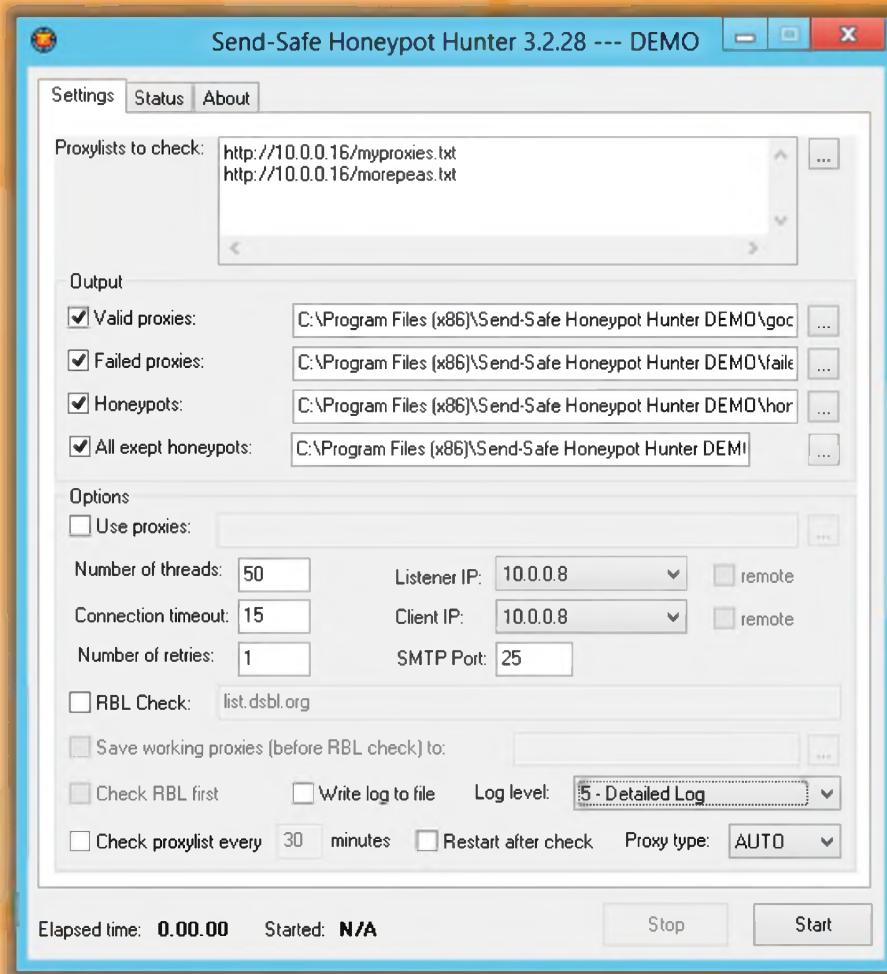
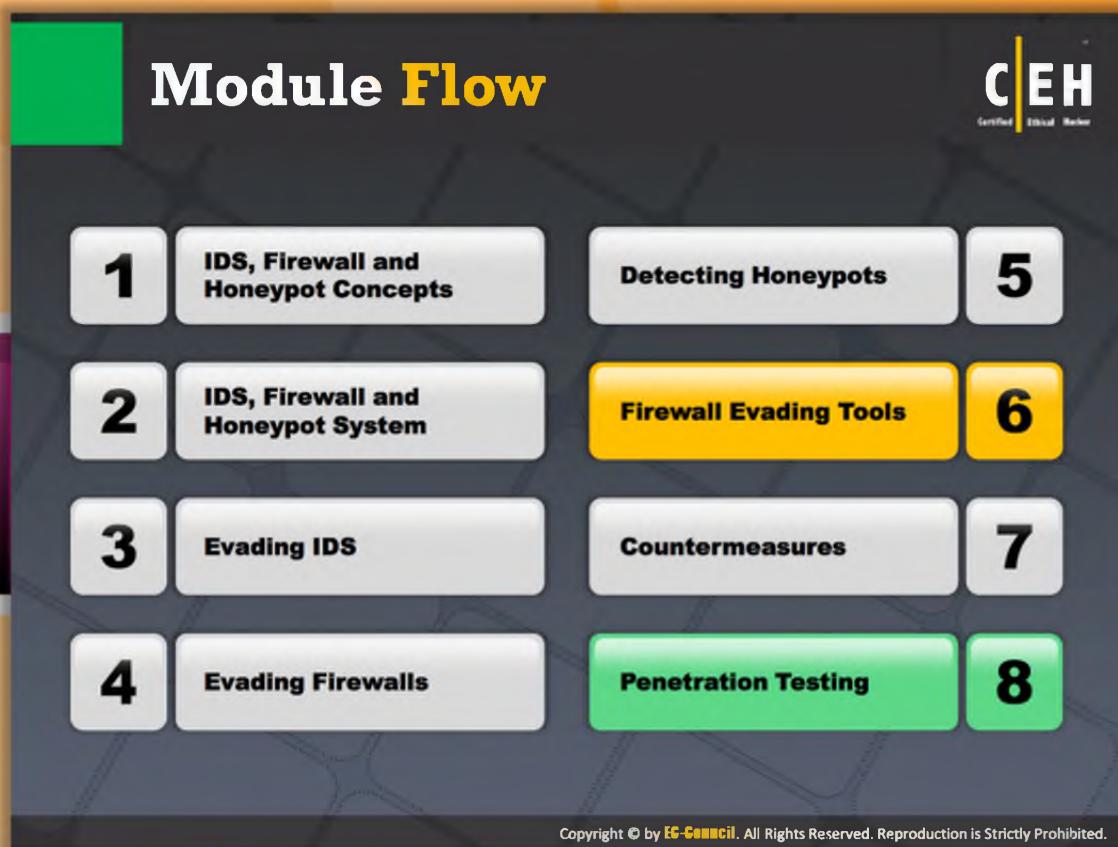


FIGURE 17.36: Honeypot Detecting Tool: Send-Safe Honeypot Hunter Screenshot



Module Flow

Firewall evasion can be accomplished with the help of tools. These tools help an attacker in evading the firewall and thus breaking into the network. With the help of tools, an attacker can evade a firewall easily and also in less time.

 IDS, Firewall and Honeypot Concepts	 Detecting Honeypots
 IDS, Firewall and Honeypot System	 Firewall Evading Tools
 Evading IDS	 Countermeasure
 Evading Firewall	 Penetration Testing

This section is dedicated to firewall evasion tools.

The screenshot shows the Traffic IQ Professional software interface. At the top, there's a banner with the title "Firewall Evasion Tool: Traffic IQ Professional" and the CEH logo. Below the banner, a red text box contains the following message: "Traffic IQ Professional enables security professionals to audit and validate the behavior of security devices by generating the standard application traffic or attack traffic between two virtual machines". To the left, there's a sidebar with a list of what the tool can be used for: Application layer firewalls, Intrusion detection systems, Intrusion prevention systems, and Routers and switches. Below this list is a small icon of a network device with yellow cables. The main window has tabs for Import, Play, Pause, Stop, and a toolbar with icons for File, Help, Traffic, Groups, Scan, Prompt, Editor, Script, Reports, and Settings. The central pane displays a list of attack types under "Program Files (x86)" and "Internal Machine". The list includes various HTTP and HTTPS attack types such as "HTTPPIE DHTML Script Injection S.kar", "HTTPPIE IE Form Dump S.kar", "HTTPPIE IE WebDAV Overflow S.kar", "HTTPPIE Object data remote execution S.kar", "HTTPPIE Object tag overflow S.kar", "HTTPPIE Popup Blocker bypass S.kar", "HTTPPIE SQLi Oracle S.kar", "HTTPPIE Telnet Spoof S.kar", "HTTPPIE WebDAV S.kar", "HTTPPIE 4.0 HTR Overflow (win32_bind) S.kar", "HTTPPIE 4.0 HTR Overflow (win32_bind, httpd) S.kar", "HTTPPIE 4.0 HTR Overflow (win32_bind, httpd, httpd) S.kar", "HTTPPIE 4.0 HTR Overflow (win32_bind, httpd_upnpoc) S.kar", "HTTPPIE 4.0 HTR Overflow (win32_bind, vncclient) S.kar", "HTTPPIE 4.0 HTR Overflow (win32_bind, vncviewer) S.kar", "HTTPPIE 4.0 SQLi Code Disclosure (CodeDisc_vsp) S.kar", "HTTPPIE 5.0 ISAPI POST Overflow (win32_bind) S.kar", "HTTPPIE 5.0 ISAPI POST Overflow (win32_bind_msvncat) S.kar", "HTTPPIE 5.0 ISAPI POST Overflow (win32_bind_msi) S.kar", and "HTTPPIE 5.0 IIS-RAR POST Overflow (win32_bind) S.kar". Below this list is a table titled "Adapter Status" with columns for Internal Machine, External Machine, and Adapter Status. The table shows "Internal Machine: Ethernet" and "External Machine: Ethernet" with "Adapter Status: Packets sent: 21". At the bottom of the window, it says "Ready". The URL "http://www.idappcom.com" is visible at the bottom right.

Firewall Evasion Tool: Traffic IQ Professional

Source: <http://www.idappcom.com>

Traffic IQ Professional **enables security professionals** to audit and validate the behavior of security devices by generating the standard application traffic or attack traffic between two virtual machines. The **unique features** and packet transmission capabilities of Traffic IQ Professional make the task of reliably auditing, validating, and proving **security compliance** easy and quick to complete. It can be used to assess, audit, and test the behavioral characteristics of any **non-proxy packet-filtering** device including **Application layer firewalls**, intrusion detection systems, intrusion prevention systems, and routers and switches.

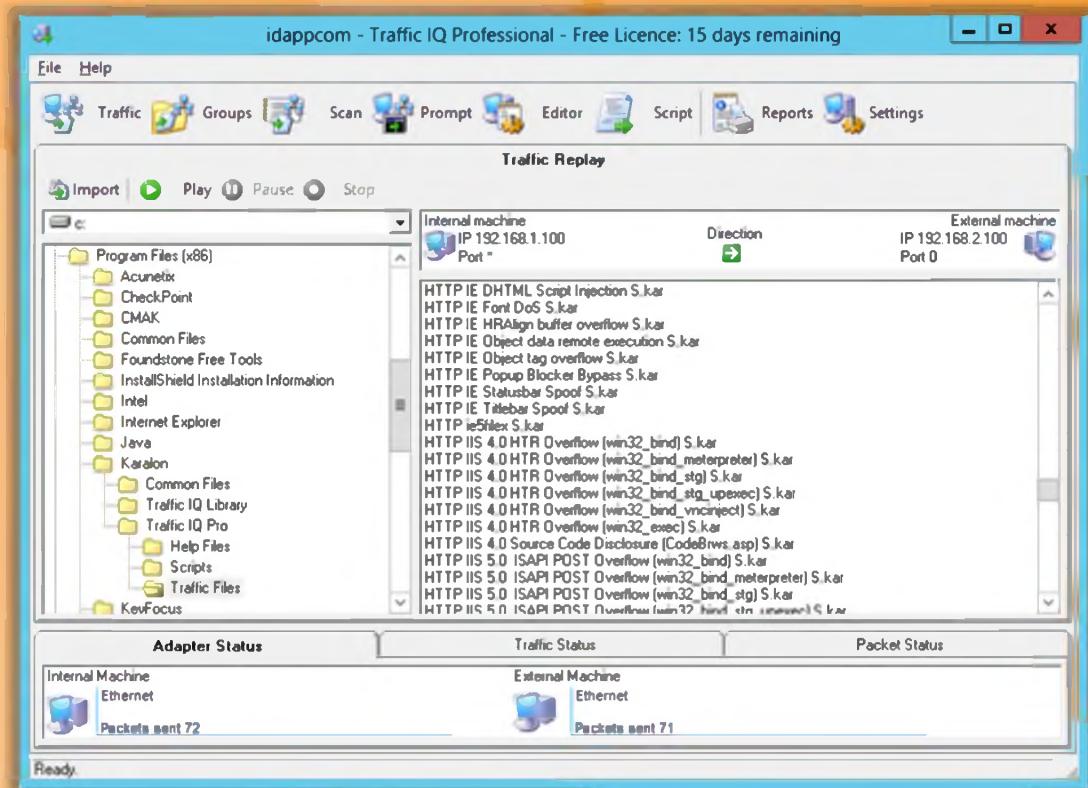


FIGURE 17.37: Traffic IQ Professional Screenshot

Firewall Evasion Tool: **tcp-over-dns**

C|EH
Certified Ethical Hacker

tcp-over-dns

- tcp-over-dns contains a special **dns server** and a special **dns client**
- The client and server work in tandem to provide a **TCP (and UDP!) tunnel** through the standard DNS protocol

```
C:\>java -jar tcp-over-dns-1.0>java -jar tcp-over-dns-server.jar --domain dnstunnel --forward-port 22
dnstunnel.B main: tcp-over-dns-server starting up
dnstunnel.B main: Hosting domain: dnstunnel
dnstunnel.B main: DNS listening on: /0.0.0.0:53
dnstunnel.B main: Forwarding to: /127.0.0.1:22
dnstunnel.B main: MTU: 1500
dnstunnel.B main: Log level: 3
```

<http://analogbit.com>

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



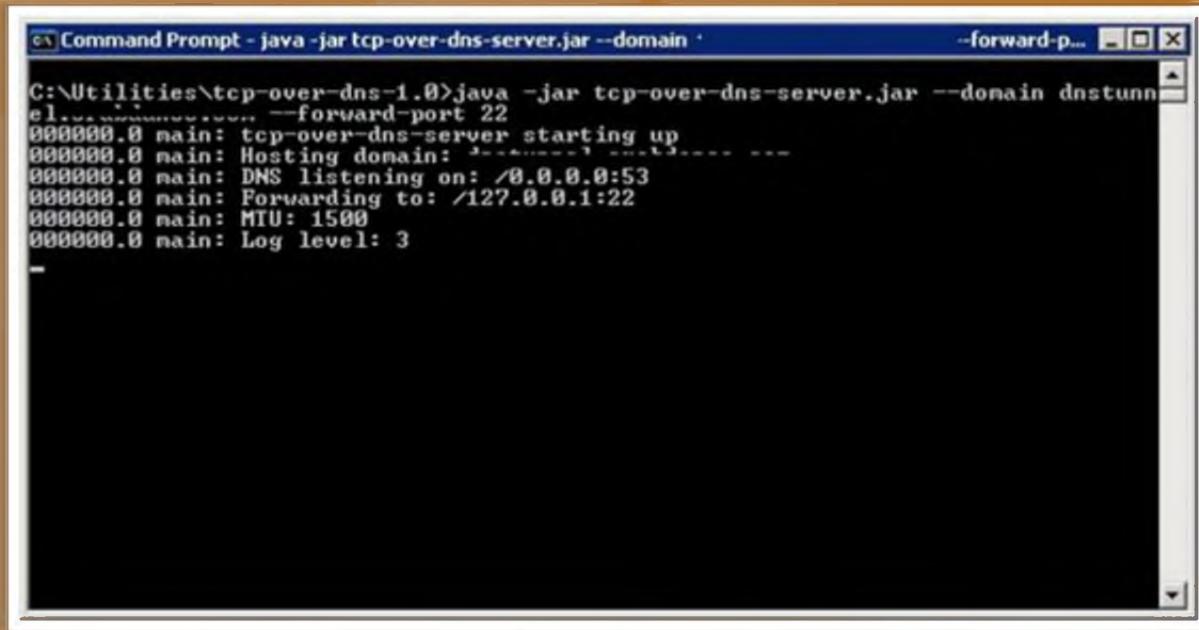
Firewall Evasion Tool: **tcp-over-dns**

Source: <http://analogbit.com>

tcp-over-dns contains a special dns server and a special **dns client**. The client and server work in tandem to provide a **TCP (and UDP!) tunnel** through the standard DNS protocol. It is similar to the defunct **NSTX dns tunneling software**. The purpose of this software is to succeed where **NSTX failed**. All NSTX tunnels disconnect within tens of seconds in real-world situations. tcp-over-dns is written to be quite robust while at the same time providing acceptable bandwidth speeds.

Its features include:

- Windows, Linux, Solaris compatibility
- Sliding window packet transfers for increased speed and reliability
- Runtime selective LZMA compression
- TCP and UDP traffic tunneling



```
C:\Utilities\tcp-over-dns-1.0>java -jar tcp-over-dns-server.jar --domain dnstunnel
--forward-port 22
main: tcp-over-dns-server starting up
main: Hosting domain: dnstunnel
main: DNS listening on: /0.0.0.0:53
main: Forwarding to: /127.0.0.1:22
main: MTU: 1500
main: Log level: 3
```

FIGURE 17.38: tcp-over-dns in command prompt

Firewall Evasion Tools

C|EH
Certified Ethical Hacker

 Snare Agent for Windows http://www.intersectalliance.com	 Freenet https://freenetproject.org
 AckCmd http://ntsecurity.nu	 GTunnel http://gardenetworks.org
 Tomahawk http://tomahawk.sourceforge.net	 Hotspot Shield http://www.anchorfree.com
 Your Freedom http://www.your-freedom.net	 Proxifier http://www.proxifier.com
 Atelier Web Firewall Tester http://www.atelierweb.com	 Vpn One Click http://www.vpnoneclick.com

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Firewall Evasion Tools

Firewall evasion tools helps in **breaching a firewall** from inside as well as exporting data with innocent-looking packets that contain insufficient data for sniffers or firewalls to analyze. A few firewall evasion tools are listed as follows:

- ② Snare Agent for Windows available at <http://www.intersectalliance.com>
- ② AckCmd available at <http://ntsecurity.nu>
- ② Tomahawk available at <http://tomahawk.sourceforge.net>
- ② Your Freedom available at <http://www.your-freedom.net>
- ② Atelier Web Firewall Tester available at <http://www.atelierweb.com>
- ② Freenet available at <https://freenetproject.org>
- ② GTunnel available at <http://gardenetworks.org>
- ② Hotspot Shield available at <http://www.anchorfree.com>
- ② Proxifier available at <http://www.proxifier.com>
- ② Vpn One Click available at <http://www.vpnoneclick.com>

Packet Fragment Generators



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

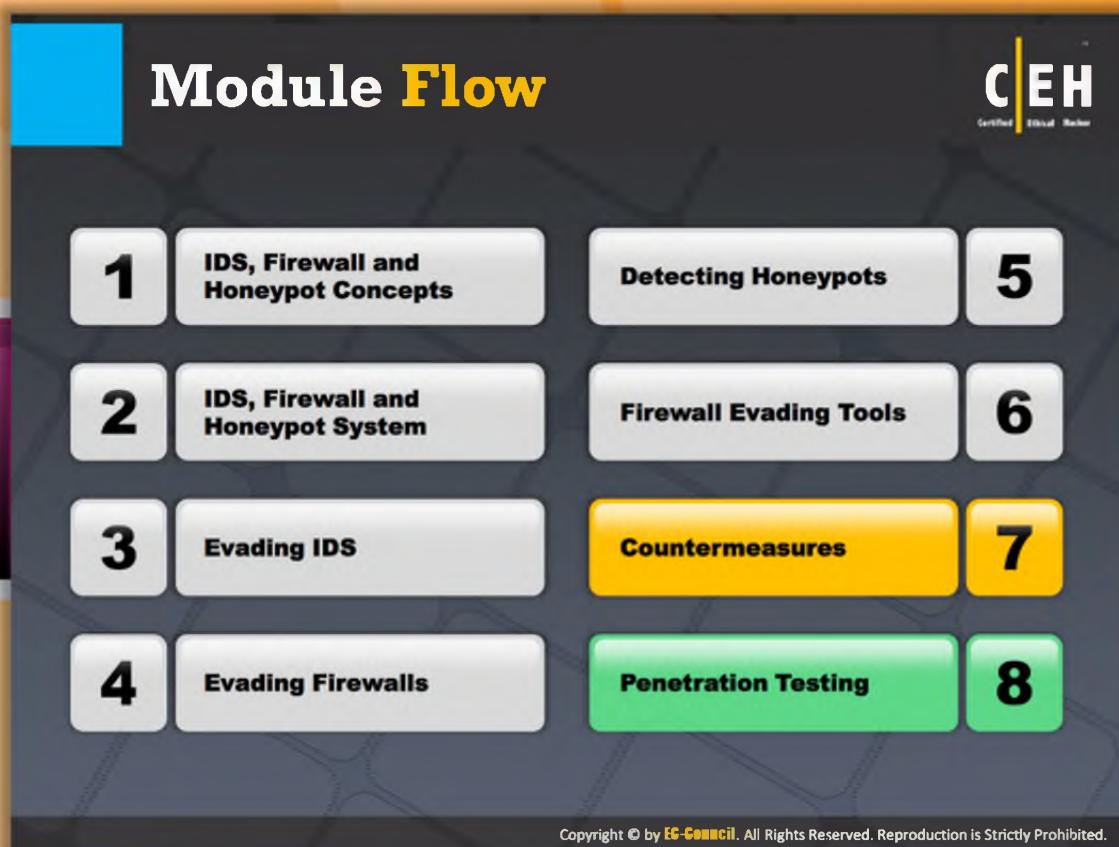
 Colasoft Packet Builder http://www.colasoft.com	 NConvert http://www.xnview.com
 CommView http://www.tamos.com	 fping 3 http://fping.org
 hping3 http://www.hping.org	 NetScanTools Pro http://www.netscantools.com
 Multi-Generator (MGEN) http://cs.itd.nrl.navy.mil	 pktgen http://www.linuxfoundation.org
 Net-Inspect http://search.cpan.org	 PacketMaker http://www.jdsu.com



Packet Fragment Generators

Packet fragment generators allow you to **edit** and **send packets** via your wireless network adapter. They allow you to hide your network file transfers across the Internet. By utilizing packet forgery, these tools hide your **file transfer** by cloaking it in seemingly harmless data. A few packet fragment generators are listed as follows:

- ④ Colasoft Packet Builder available at <http://www.colasoft.com>
- ④ CommView available at <http://www.tamos.com>
- ④ hping3 available at <http://www.hping.org>
- ④ Multi-Generator (MGEN) available at <http://cs.itd.nrl.navy.mil>
- ④ Net-Inspect available at <http://search.cpan.org>
- ④ NConvert available at <http://www.xnview.com>
- ④ fping 3 available at <http://fping.org>
- ④ NetScanTools Pro available at <http://www.netscantools.com>
- ④ Pktgen available at <http://www.linuxfoundation.org>
- ④ PacketMaker available at <http://www.jdsu.com>

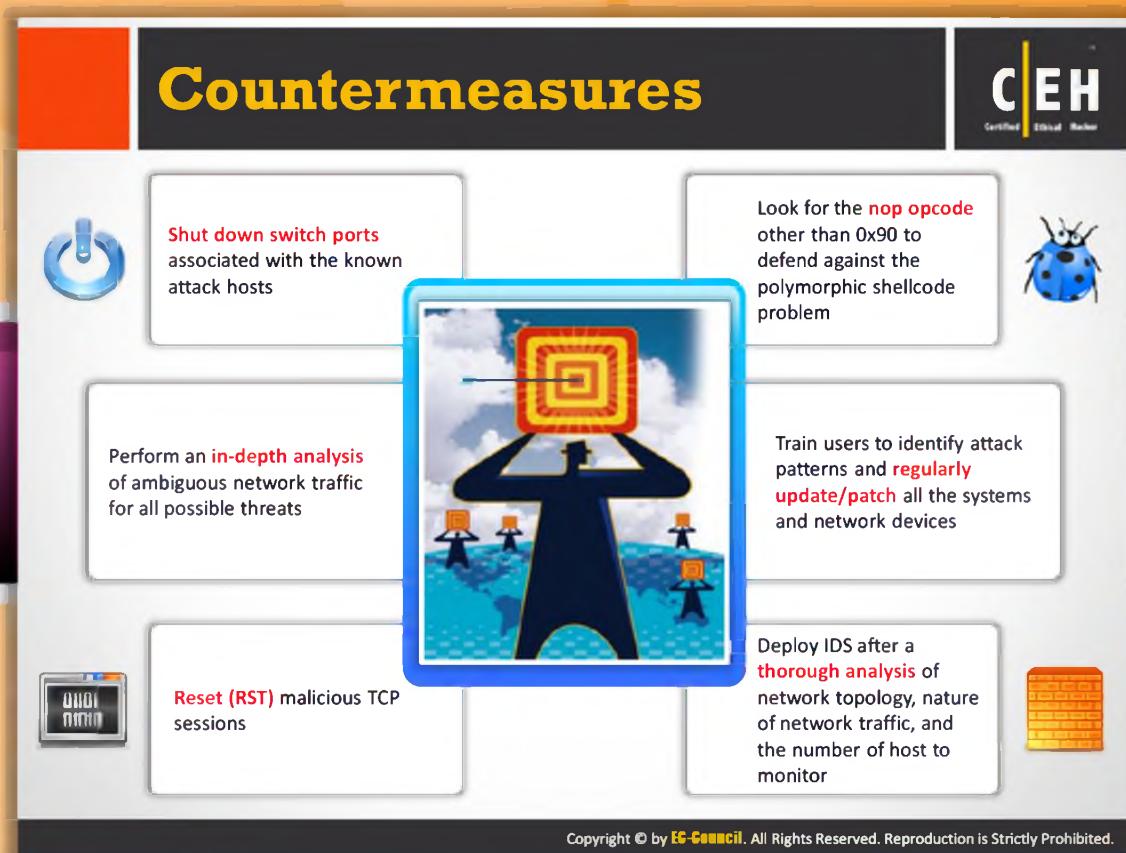


Module Flow

So far, we have discussed various concepts and topics related to intruding into or bypassing security mechanisms such as IDSes, firewalls, and honeypots. Now we will discuss the ways to protect them, i.e., countermeasures. Countermeasures help in enhancing security.

IDS, Firewall and Honeypot Concepts	Detecting Honeypots
IDS, Firewall and Honeypot System	Firewall Evading Tools
Evading IDS	Countermeasure
Evading Firewall	Penetration Testing

This section highlights various countermeasures against IDSes, firewalls, and honeypot attacks.



Countermeasures

The following are few countermeasures that provide **protection against evading IDSes, firewalls, and honeypots**:

- Administratively shut down a switch port interface associated with a system from which attacks are being launched.
- Look for the nop opcode other than **0x90** to defend against the polymorphic **shellcode problem**.
- Perform "bifurcating analysis," in which the monitor deals with ambiguous traffic streams by instantiating **separate analysis** threads for each possible interpretation of the ambiguous traffic.
- Maintain security vulnerability awareness, patch vulnerabilities as soon as possible, and wisely choose the IDS based on the network topology and network traffic received.
- Generate TCP RST packets to tear down malicious TCP sessions, any issues of several available ICMP error code packets in response to malicious UDP traffic.
- Interact with the external firewall or router to add a general rule to block all communication from individual IP addresses or entire networks.

Countermeasures (Cont'd)

C|EH
Certified Ethical Hacker

- Use a traffic normalizer to remove potential ambiguity from the packet stream before it reaches to the IDS
- Ensure that IDSs **normalize fragmented packets** and allow those packets to be reassembled in the proper order
- Define DNS server for client resolver in routers or similar network devices
- Harden the security of all communication devices such as modems, routers, switches, etc.
- If possible, block **ICMP TTL expired** packets at the external interface level and change the **TTL field to a large value**, ensuring that the end host always receives the packets

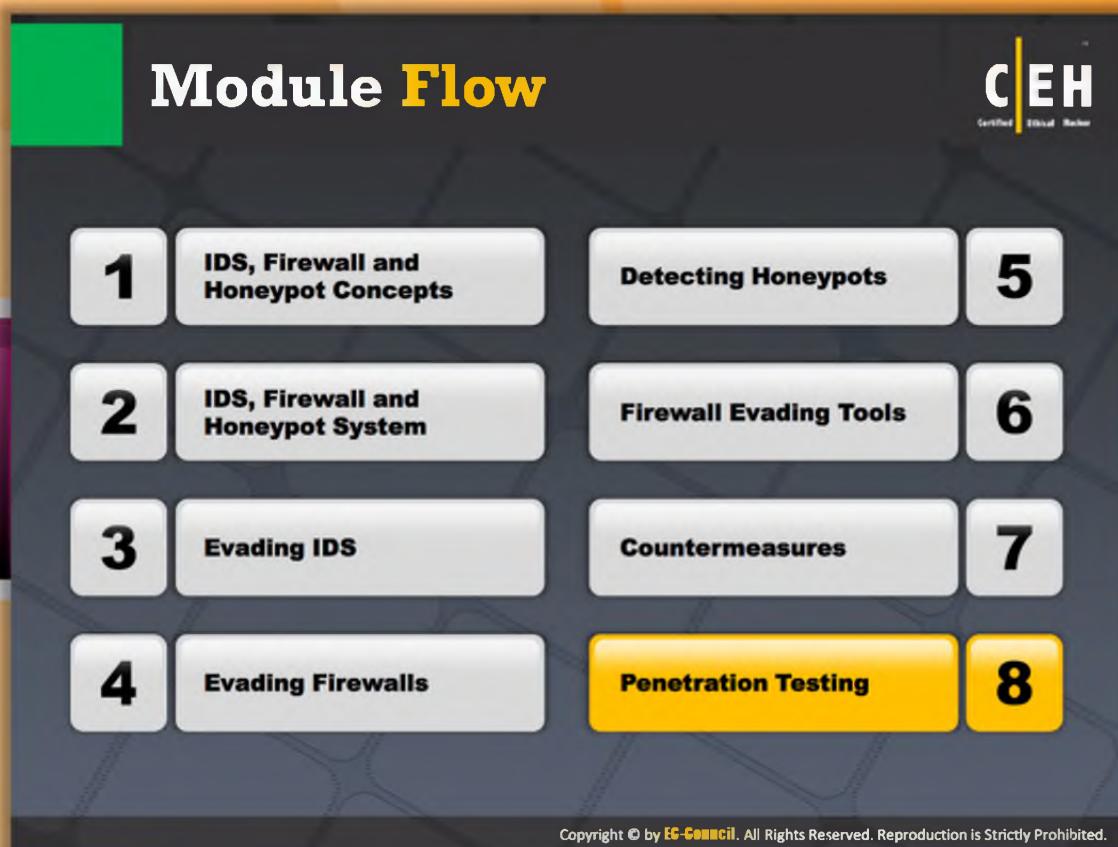
Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Countermeasures (Cont'd)

The following are additional countermeasures against evading IDSSes, firewalls, and honeypots:

- Implement a "**traffic normalizer**": a network forwarding element that attempts to eliminate ambiguous network traffic and reduce the amount of connection state that the monitor must maintain.
- Ensure that **IDSs normalize fragmented** packets and allow those packets to be reassembled in the proper order, which enables the IDS to look at the information just as the end host can see it.
- Keep updating the **IDS system** and **firewall software regularly**.
- Maintain security vulnerability awareness, patch vulnerabilities as soon as possible, and wisely choose the IDS based on the network topology and network traffic received.
- Change the TTL field to a large value, ensuring that the end host always receives the packets. In such case, attackers cannot slip information to the IDS. As a result, that data never reaches the end host, leaving the end host with the **malicious payload**.



Module Flow

You need to conduct penetration test on firewalls, IDSe, and honeypots in order to ensure that they can withstand against different types attacks carried out by attackers. As a pen tester, you should conduct penetration testing on firewalls, IDSe, and honeypots to determine the vulnerabilities present in them before the attacker determines and exploits them.

	IDS, Firewall and Honeypot Concepts		IDS, Firewall and Honeypot System
	Evading IDS		Evading Firewalls
	Detecting Honeypots		Firewall Evading Tools
	Countermeasures		Penetration Testing

This section shows the importance of firewall/IDS pen testing and also describes the steps involved in it.

Firewall/IDS Penetration Testing

C|EH Certified Ethical Hacker

Firewall/IDS penetration testing is to evaluate the Firewall and IDS for **ingress** and **egress** traffic filtering capabilities

Why Firewall/IDS pen testing?

To check if firewall/IDS properly enforces an organization's firewall/IDS policy	To check the amount of network information accessible to an intruder
To check if the IDS and firewalls enforces organization's network security policies	To check the firewall/IDS for potential breaches of security that can be exploited
To check if the firewall/IDS is good enough to prevent the external attacks	To evaluate the correspondence of firewall/IDS rules with respect to the actions performed by them
To check the effectiveness of the network's security perimeter	To verify whether the security policy is correctly enforced by a sequence of firewall/IDS rules or not

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



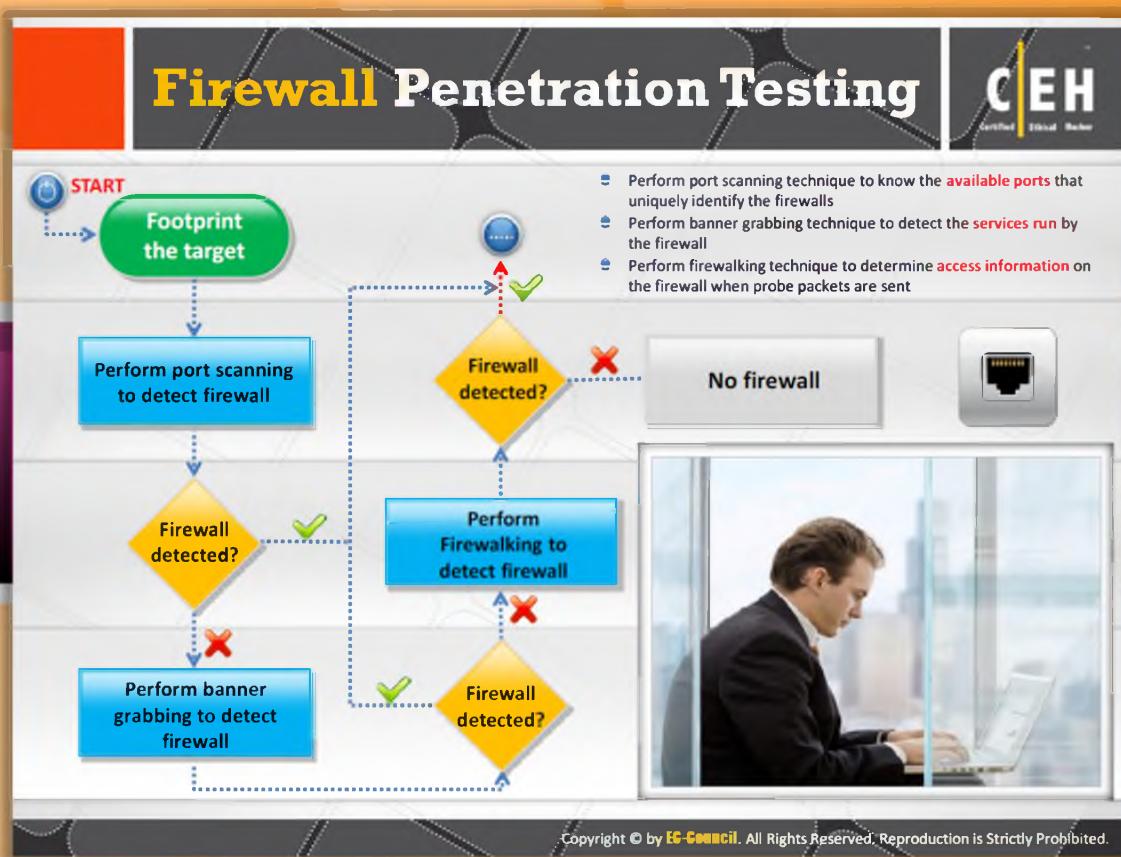
Firewall/IDS Penetration Testing

Firewall/IDS penetration testing is conducted to identify if there is any **security vulnerability** related to hardware, software and its configuration, and how to protect the network from outside attackers. It helps in **evaluating security** by testing for ingress and egress **vulnerabilities** and proper rule sets of the entire network with respect to the possibility of entry from an external location

Why firewall/IDS pen testing?

Firewall/IDS pen testing is required to:

- ☛ Check if firewall/IDS properly enforces an organization's firewall/IDS policy
- ☛ Check if firewall/IDS and components within network properly enforce an organization's network security policy
- ☛ Check the strength of **firewall/IDS protection** against externally initiated attacks
- ☛ Check the effectiveness of the network's security perimeter
- ☛ Check how much information about a network is available from outside a network
- ☛ Check the firewall/IDS for potential breaches of security that can be exploited
- ☛ Evaluate the correspondence of firewall/IDS rules with respect to the actions performed by them
- ☛ Verify whether the security policy is correctly enforced by a sequence of firewall/IDS rules or not



Firewall Penetration Testing

As a pen tester, you should implement the following steps to conduct penetration testing on a firewall.

Step1: Footprint the target

You should footprint the target by **using various tools** such as Sam Spade, nslookup, traceroute, Nmap, and nettrace to learn about a system, its **remote access capabilities**, its ports and services, and the other aspects of its security.

Step2: Perform port scanning

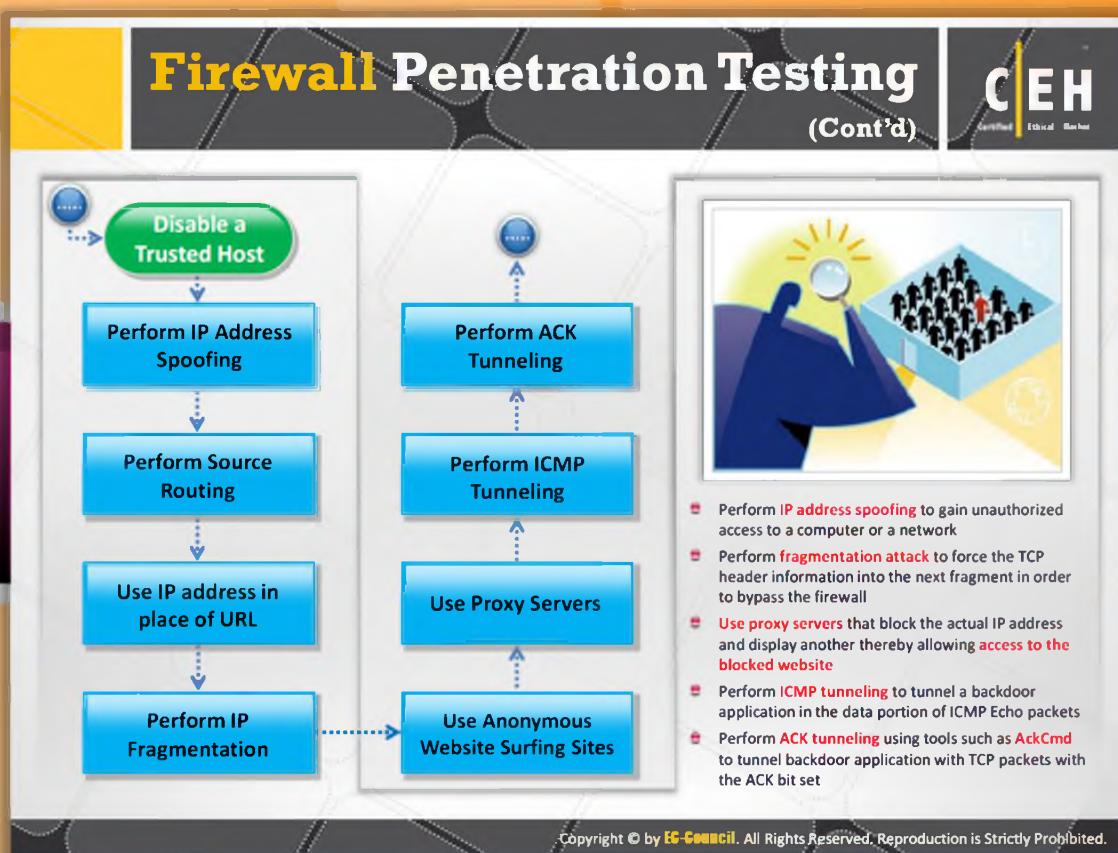
You should perform **port scanning** to detect the firewall to determine the available ports that uniquely identify the firewalls. If the firewall is detected, then disable a trusted host or perform banner grabbing to **detect the firewall**.

Step3: Perform banner grabbing

You should perform the **banner grabbing technique** to detect the services run by the firewall. If the firewall is detected, then disable a trusted host or perform **firewalking** to detect the firewall.

Step4: Perform firewalking

You should use the firewalking technique to determine access information on the firewall when probe packets are sent. If a firewall is detected, then disable a trusted host.



Firewall Penetration Testing (Cont'd)

Step 5: Disable the trusted host

Step 6: Perform IP address spoofing

You should perform IP address spoofing to **gain unauthorized access** to a computer or a network.

Step 7: Perform source routing

Step 8: Use an IP address in place of URL

Step 9: Perform a fragmentation attack

You should perform an IP fragmentation attack to force the **TCP header information** into the next fragment in order to bypass the firewall.

Step 10: Use anonymous website surfing sites

You should use anonymous website surfing sites to **hide your identity** from the Internet.

Step 11: Use proxy servers

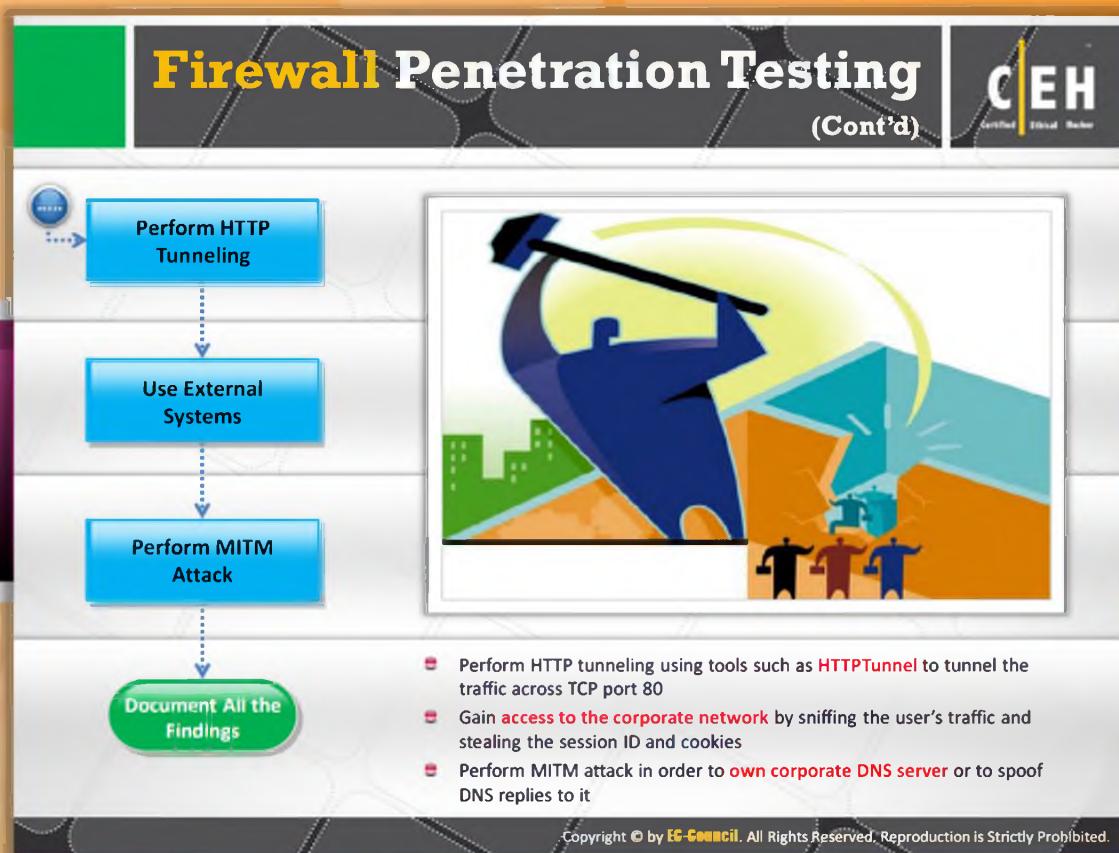
You should use proxy servers that block the actual IP address and display another, thereby allowing access to the blocked website.

Step12: Perform ICMP tunneling

You should perform ICMP tunneling to tunnel a backdoor application in the data portion of ICMP Echo packets.

Step13: Perform ACK tunneling

You should perform ACK tunneling using tools such as AckCmd to tunnel backdoor application with TCP packets with the ACK bit set.



Firewall Penetration Testing (Cont'd)

Step14: Perform HTTP tunneling

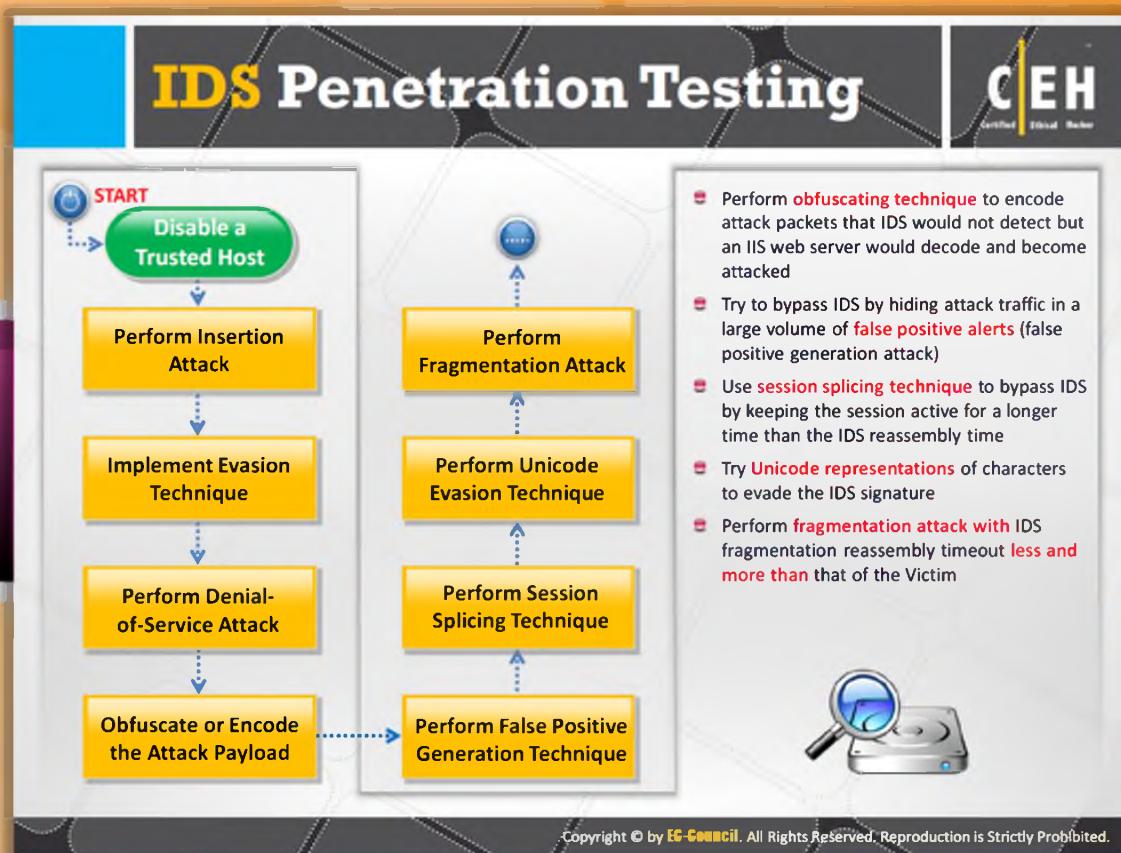
You should perform HTTP tunneling using tools such as **HTTPTunnel** to tunnel the traffic across TCP port 80.

Step15: Use external systems

Step16: Perform MITM Attack

You should perform an MITM attack in order to own corporate the **DNS server** or to **spoof DNS** replies to it.

Step 17: Document all the findings



IDS Penetration Testing

You should carry out following steps to conduct IDS penetration testing.

Step1: Disable a trusted host

You should try to find and **disable the trusted host** so that the targeted host thinks that the traffic that the attacker will generate emanates from there.

Step2: Perform an insertion attack

Step3: Implement the evasion technique

Step4: Perform a denial-of-service attack

Step5: Obfuscate or encode the attack payload

You should implement the **obfuscating technique** to encode attack packets that the IDS would not detect but an IIS web server would decode and be attacked.

Step6: Perform the false positive generation technique

You should use the **false positive generation technique** to create a great deal of log "noise" in an attempt to blend real attacks with the false.

Step7: Perform the Session Splicing Technique

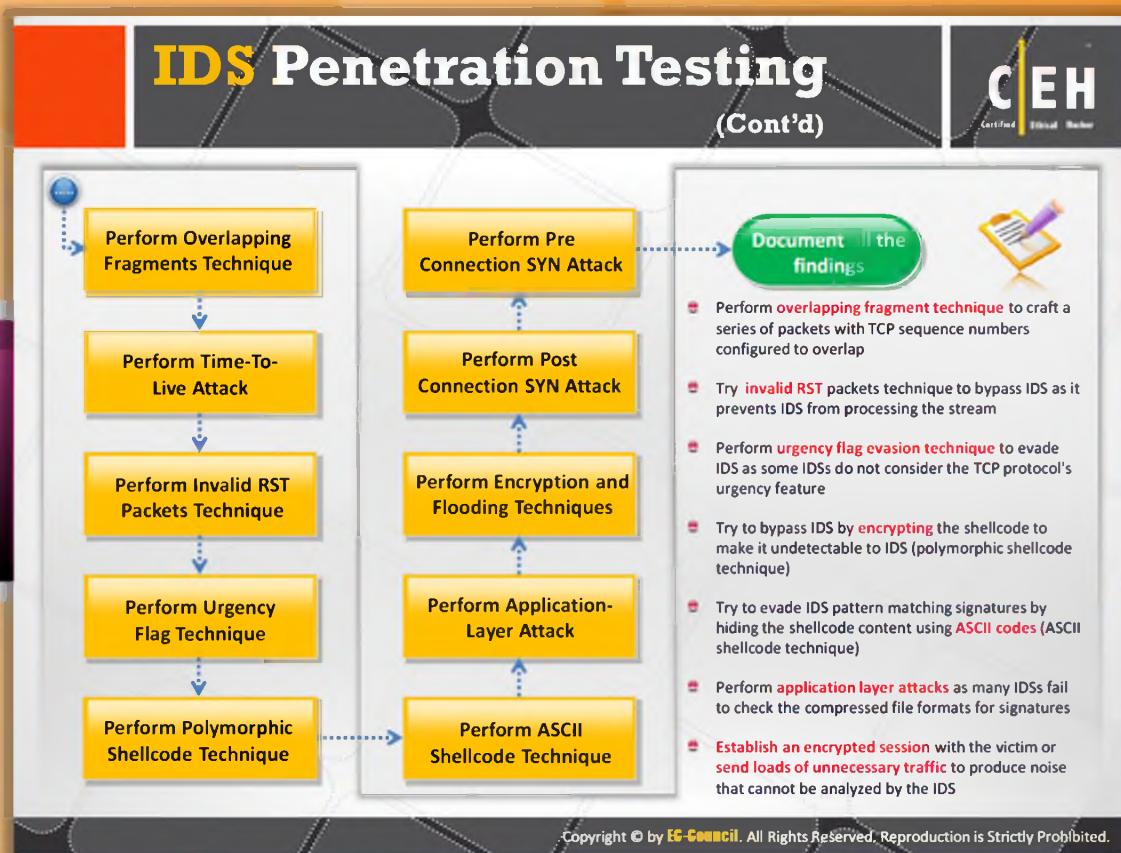
You should implement the **session splicing technique** to stop the IDS by keeping the session active longer than IDS will spend on reassembling it.

Step8: Perform the Unicode evasion technique

You should implement the **Unicode evasion technique** to evade IDSSes as it is possible to have multiple representations of a single character.

Step 9: Perform a fragmentation attack

You should perform a fragmentation attack with **IDS fragmentation** reassembly timeout less and more than that of the victim.



IDS Penetration Testing (Cont'd)

Step10: Perform the overlapping fragments technique

You should use **the overlapping** fragments technique to craft a series of packets with TCP sequence numbers configured to overlap.

Step 11: Perform a Time-To-Live attack

Step 12: Perform the invalid RST packets technique

You should use the invalid **RST packets** technique to evade detection by sending RST packets with an invalid checksum that causes the IDS to stop processing the stream.

Step13: Perform the urgency flag technique

You should use the urgency flag technique to evade **IDSrd** as some **IDSrds** do not consider the **TCP protocol's urgency** feature.

Step14: Perform the polymorphic shellcode technique

You should use the polymorphic shellcode technique to hide the shellcode by encrypting it in a simplistic form that is difficult for IDS to identify that data as a shellcode.

Step15: Perform the ASCII shellcode technique

You should perform the **ASCII shellcode** technique to bypass IDS pattern matching signatures because strings are hidden within the shellcode as in a polymorphic shellcode.

Step16: Perform an Application-layer attacks

You should try to perform Application-level attacks as many IDSes will have no way to check the compressed file format for signatures.

Step17: Perform encryption and flooding techniques

You should try **encryption** and **flooding attacks** with the victim or send loads of unnecessary traffic to produce noise that can't be analyzed by the IDS.

Step18: Perform a post-connection SYN attack

Step19: Perform a pre-connection SYN attack

Step 20: Document all the results obtained from this test

Module Summary



- Intrusion Detection Systems (IDS) monitor packets on the network wire and attempt to discover if an attacker is trying to break into a system
- System Integrity Verifiers (SIV) monitor the system files to find when an intruder changes. Tripwire is one of the popular SIVs
- Intrusion detection happens either by anomaly detection or signature recognition or Protocol Anomaly Detection
- Firewall is a hardware, software or a combination of both that is designed to prevent unauthorized access to or from a private network
- Firewall is identified by three techniques namely port scanning, banner grabbing, and firewalking
- Honeypots are programs that simulate one or more network services that are designated on a computer's ports
- In order to effectively detect intrusions that use invalid protocol behavior, IDS must re-implement a wide variety of application-layer protocols to detect suspicious or invalid behavior
- One of the easiest and most common ways for an attacker to slip by a firewall is by installing network software on an internal system that uses a port address permitted by the firewall's configuration

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Module Summary

- Intrusion detection systems (IDSe) monitor packets on the network wire and attempt to discover if an attacker is trying to break into a system.
- System integrity verifiers (SIVs) monitor the system files to find when an intruder changes. Tripwire is one of the popular SIVs.
- Intrusion detection happens either by anomaly detection or signature recognition or protocol anomaly detection.
- A firewall is hardware, software, or a combination of both that is designed to prevent unauthorized access to or from a private network.
- A firewall is identified by three techniques: port scanning, banner grabbing, and firewalking.
- Honeypots are programs that simulate one or more network services that are designated on a computer's ports.
- In order to effectively detect intrusions that use invalid protocol behavior, an IDS must re-implement a wide variety of Application-layer protocols to detect suspicious or invalid behavior.

- ➊ One of the easiest and most common ways for an attacker to slip by a firewall is by installing network software on an internal system that uses a port address permitted by the firewall's configuration.

Buffer Overflow

Module 18





Ethical Hacking and Countermeasures v8

Module 18: Buffer Overflow

Exam 312-50

The screenshot shows a news website with a dark header containing the text "Security News" and the "CEH Certified Ethical Hacker" logo. Below the header is a navigation bar with four tabs: "Home", "News" (which is highlighted in orange), "Services", and "About Us". The main content area features a story titled "Steam Gaming Platform Vulnerable to Remote Exploits; 50 Million at Risk". To the left of the article are three small icons: a green one showing a file, a yellow one with a skull and bomb, and a blue one showing a globe. The story discusses a vulnerability in the Steam platform's URL protocol handler, which allows attackers to abuse browser requests. It quotes researchers Luigi Auriemma and Donato Ferrante. The source of the news is cited as <http://threatpost.com>. The copyright notice at the bottom states "Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited."



Security News

Steam Gaming Platform Vulnerable to Remote Exploits; 50 Million at Risk

Source: <http://threatpost.com>

More than 50 million users of the Steam gaming and media distribution platform are at **risk for remote compromise** because of weaknesses in the platform's URL protocol handler, a pair of researchers at ReVuln wrote in a paper released this week.

Luigi Auriemma and Donato Ferrante discovered a number of **memory corruption** issues, including **buffer and heap overflows** that would allow an attacker to abuse the way the Steam client handles browser requests. Steam runs on Windows, Linux and Mac OSX.

The steam:// **URL protocol** is used to connect to game servers, load and uninstall games, backup files, run games and interact with news, profiles and download pages offered by Valve, the company that operates the platform. Attackers, Auriemma and Ferrante said, can abuse specific Steam commands via steam:// URLs to inject attacks and run other **malicious code** on victim machines.

"We proved that the current implementation of the **Steam Browser Protocol handling mechanism** is an excellent attack vector, which enables **attackers to exploit** local issues in a

remote fashion,” Auriemma and Ferrante wrote. “Because of the big audience, the support for several different platforms and the amount of effort required to exploit bug via the Steam Browser Protocol commands, Steam can be considered a high-impact attack vector.”

A large part of the problem rests in the fact that most browsers don’t ask for **user permission** before interacting with the **Steam client**, and those that do, don’t explain there could be a security issue. As a result, users could be **tricked into clicking** on a malicious steam:// URL or redirect browsers via JavaScript to a **malicious site**, the paper said.

The paper details five new remotely **exploitable vulnerabilities** in not only Steam, but also in the Source and Unreal game engines. Some of the games running on the affected platforms include Half-Life 2 Counter-Strike, Team Fortress 2, Left 4 Dead, Nuclear Dawn, Smashball and many others.

One of the more **dangerous** vulnerabilities discovered is involves the retailinstall command that allows Steam to install or restore backups from a local directory. An attacker can abuse the directory path to point to a remote network folder and then attack the function that processes a .tga splash image which is vulnerable to an integer overflow attack. A **heap-based overflow** results and an attacker could **remotely execute code**.

To exploit the Source game engine, Auriemma and Ferrante used a **malicious** .bat file placed in the startup folder of the user’s account that executes upon the gamer’s next login.

The pair also found several integer **overflow flaws** in the Unreal gaming engine by taking advantage of a condition where **Unreal** supports the loading of content from remote machines via Windows WebDAV or a SMB share. Malicious content could be **remotely injected** in this way.

Auto-update function vulnerabilities in a pair of games, All Points Bulletin and MicroVolts, were also discovered and exploited. The researchers were able to exploit a directory traversal to overwrite or create any malicious file.

Users reduce the impact of these issues by disabling the steam:// URL handler or using a browser that doesn’t allow direct execution of the Steam Browser Protocol. Steam could also deny the passing of **command-line arguments** to remote software.



Copyright © 2012 threatpost.com

By Michael Mimoso

http://threatpost.com/en_us/blogs/steam-gaming-platform-vulnerable-remote-exploits-50-million-risk-101912

Module Objectives

CEH
Certified Ethical Hacker

- Heap-Based Buffer Overflow
- Why Are Programs and Applications Vulnerable to Buffer Overflows?
- Knowledge Required to Program Buffer Overflow Exploits
- Buffer Overflow Steps
- Overflow Using Format String
- Buffer Overflow Examples

- How to Mutate a Buffer Overflow Exploit
- Identifying Buffer Overflows
- How to Detect Buffer Overflows in a Program
- BoF Detection Tools
- Defense Against Buffer Overflows
- Buffer Overflow Security Tools
- Buffer Overflow Penetration Testing



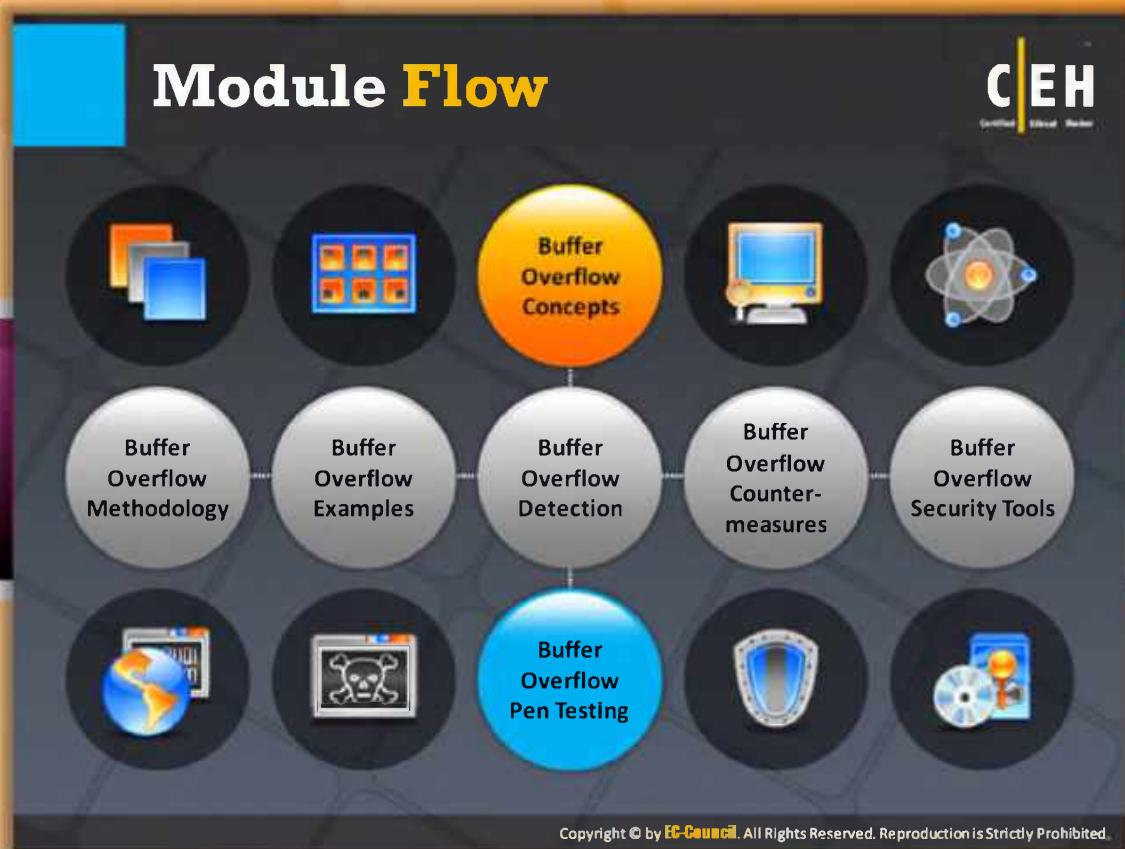
Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Module Objectives

Various security concerns, attack methods, and countermeasures have been discussed in the previous modules. **Buffer overflow attacks** have been a source of worry from time to time. This module looks at different aspects of buffer overflow **exploits** that include:

- Heap-Based Buffer Overflow
- Why Are Programs and Applications Vulnerable to Buffer Overflows?
- Knowledge Required to Program Buffer Overflow Exploits
- Buffer Overflow Steps
- Overflow Using Format String
- Buffer Overflow Examples
- How to Mutate a Buffer Overflow Exploit
- Identifying Buffer Overflows
- How to Detect Buffer Overflows in a Program
- BoF Detection Tools
- Defense Against Buffer Overflows
- Buffer Overflow Security Tools
- Buffer Overflow Penetration Testing



Module Flow

Many applications and programs are vulnerable to buffer overflow attacks. This is often overlooked by application developers or programmers. Though it seems to be simple, it may lead to severe consequences. To avoid the complexity of the buffer overflow vulnerability subject, we have divided it into various sections. Before going technically deep into the subject, first we will discuss buffer overflow concepts.

	Buffer Overflow Concepts		Buffer Overflow Countermeasures
	Buffer Overflow Methodology		Buffer Overflow Security Tools
	Buffer Overflow Examples		Buffer Overflow Pen Testing
	Buffer Overflow Detection		

This section describes buffer overflows, various kinds of buffer overflows (stack-based and heap-based), stack operations, shellcode, and NOPs.

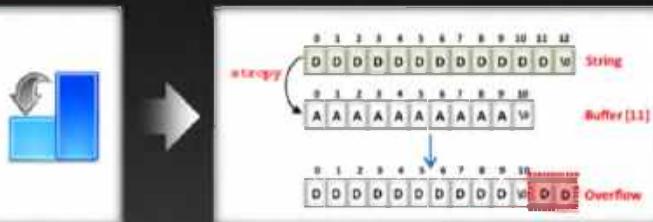
Buffer Overflows

C|EH
Certified Ethical Hacker

-  A generic buffer overflow occurs when a program tries to **store more data** in a buffer than it was intended to hold
-  When the **Buffer Overflow example code** shown below is compiled and run, an array "Buffer" of size 11 bytes is allocated to hold the "AAAAAAA" string
-  **strcpy()** will copy the string "DDDDDDDDDDDD" into the array "Buffer", which will exceed the buffer size of 11 bytes, resulting in buffer overflow

Buffer Overflow Example Code:

```
1: #include<stdio.h>
2: int main (int argc , char **argv)
3: {
4:     char Buffer[11]={"AAAAAAA"};
5:     strcpy(Buffer,"DDDDDDDDDDDD");
6:     printf("%\n",Buffer);
7:     return 0;
8: }
```



This type of vulnerability is prevalent in UNIX- and NT-based systems

Copyright © by EC Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Buffer Overflows

Buffers have **data storage capacity**. If the data count exceeds the original, a buffer overflow occurs. Buffers are developed to maintain finite data; additional information can be directed wherever it is needed. The extra information may **overflow** into neighboring buffers, **destroying** or overwriting the **legal data**. For example, the following C program illustrates how a buffer overflow attack works, where an attacker easily **manipulates** the code:

```
#include<stdio.h>
int main (int argc , char **argv)
{
    char target[5]="TTTT";
    char attacker[11]="AAAAAAAAA";
    strcpy( attacker," DDDDDDDDDDDDD");
    printf("%\n",target);
    return 0;
}
```

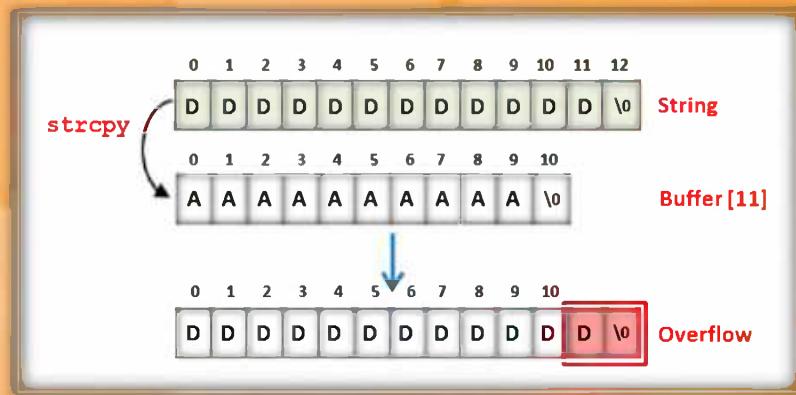


FIGURE 18.1: Buffer Overflows

The program seems to be just another normal program written by a programmer. However, the crux of this code lies in a small **manipulation** by the attacker, if examined closely. The actual problem is explained step-by-step as follows:

- During compilation of the program, the following lines of code are executed:

```
char target[5] = "TTTT";
char attacker[11] = "AAAAAAAAAA";
```

- At this point, a buffer called “target,” that can hold up to 5 characters, is created
- Then, the program places 4 Ts into the “**target**” buffer
- The program then creates a buffer called “attacker” that can hold up to 11 characters
- Then, the program places 10 As into the “attacker” buffer
- The program compiles these two lines of code

The following is a snapshot of the memory in the system. The contents of the target and **attacker buffer** are placed in the memory along with null characters, \0.

```
\0 T T T T
\0 A A A A
A A A A A A
```



Stack Memory initially

- After compiling the previously mentioned two lines of code, the compiler compiles the following lines of code:

```
strcpy( attacker, "DDDDDDDDDDDD");
printf("% \n", target);
```

- Here, in this line of code, the **string copy function** is used, which copies the 13 characters of the letter D into the attacker buffer

- ④ After this, the program prints the content of the **target buffer**
2. The `strncpy` function of the C program copies the 13 D characters into the attacker buffer, whose memory space is only 11 characters. Because there is no space for the remaining “D” characters, it eats up the memory of the “target” buffer, **destroying** the contents of the “target” buffer. Here is the snapshot of the system memory after the `strncpy` function is executed:

```
\0\0 D D D  
D D D D D  
D D D D D
```

This is how buffer overflow occurs:

A program, which seemed to be less problematic, created a **buffer overflow attack** just by manipulating one command. In the current scenario, the focus is primarily on the **Application Programming Interface** (API), which is a set of programming **conventions** facilitating direct communication with another piece of code; and the protocol, which is a set of data and commands to be passed between programs. It is a fact that many programs use a standard code set provided by the operating system when they want to use a protocol. The **APIs** associated with a program and the concerned **protocol** determines the nature of information that can be **exchanged** by the program. For instance, consider a simple login form. The login program can define the length of the input that can be accepted as the user name. However, if the program does not check for length, it is possible that the **storage space** allotted for the data may be used up, causing other areas in the memory to be used. If an attacker is able to detect this vulnerability, he or she can execute arbitrary code by causing the web application to act **erroneously**.

Why Are Programs and Applications Vulnerable to Buffer Overflows?

The diagram consists of three numbered hexagonal boxes connected by dashed lines to a central vertical dashed line. Box 1 (top) contains a blue 3D-style block icon with numbers 1, 2, and 3. Box 2 (middle) contains a person icon with a spider. Box 3 (bottom) contains a spider icon. To the right of each box is a text description:

- 1**: Boundary checks are not done fully or, in most cases, they are skipped entirely.
- 2**: Programming languages, such as C, have vulnerabilities in them. `strcat()`, `strcpy()`, `sprintf()`, `vsprintf()`, `bcopy()`, `gets()`, and `scanf()` functions do not validate target buffer size.
- 3**: Programs and applications do not adhere to good programming practices.

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Why Are Programs and Applications Vulnerable to Buffer Overflows?

In a completely **networked world**, no organization can afford to have its server go down, even for a minute. In spite of organizations taking **precautionary measures**, **exploits** are finding their way in to **disrupt** the networks due to the following reasons:

- Pressure on the deliverables—programmers are bound to make mistakes, which are overlooked most of the time
- Boundary checking** is not done or it is skipped in many of the cases
- Programming languages (such as C) that programmers still use to develop **packages** or applications contain errors
- The `strcat()`, `strcpy()`, `sprintf()`, `vsprintf()`, `bcopy()`, `gets()`, and `scanf()` calls in the C language can be abused because the functions quit **testing** if any buffer in the stack is not as large as **data copied** into that buffer
- Good programming practices** are not followed

Understanding Stacks

C|EH Certified Ethical Hacker

- Stack uses the **Last-In-First-Out (LIFO)** mechanism to pass arguments to functions and refer the local variables
- It acts like a **buffer**, holding all of the information that the function needs
- The stack is created at the beginning of the execution of a function and released at the **end of it**

Bottom of the memory

BP anywhere within the stack frame

Top of the memory

SP points here

Fill direction

Stack growth direction

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Understanding Stacks

A stack is a **contiguous block of memory** containing data. A close look at how memory is structured is shown as follows:



Code Segment

When a program runs, both **code** and **data** are **loaded into memory**. The code refers to the area where the instructions for the program are located. This segment contains all the compiled executable code for the program. Write **permission** to this segment is disabled here, as the code by itself does not contain any variables, and therefore has no need to write over itself. By having the **read-only** and **execute attributes**, the code can be shared between different copies of the program that are executing simultaneously.



Data Segment

The next section refers to the **data**, **initialized** and/or **un-initialized**, required by the running of the code instructions. This segment contains all the global data for the program. A read-write attribute is given, as programs would change the **global variables**. There is no 'execute' attribute set, as global variables are not usually meant for execution.



Stack Segment

Consider the stack as a **single-ended data structure** with first in, last out data ordering. This means that when two or more objects/elements are “**pushed**” onto the stack, to retrieve the first element, the subsequent ones have to be “**popped**” off of the stack. In other words, the most recent element remains on top of the stack. As shown previously, there is a progression from a **lower memory address** to a **higher memory address** as one moves down the stack.

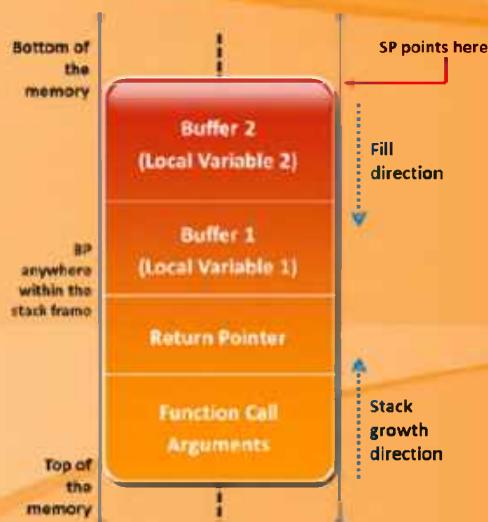
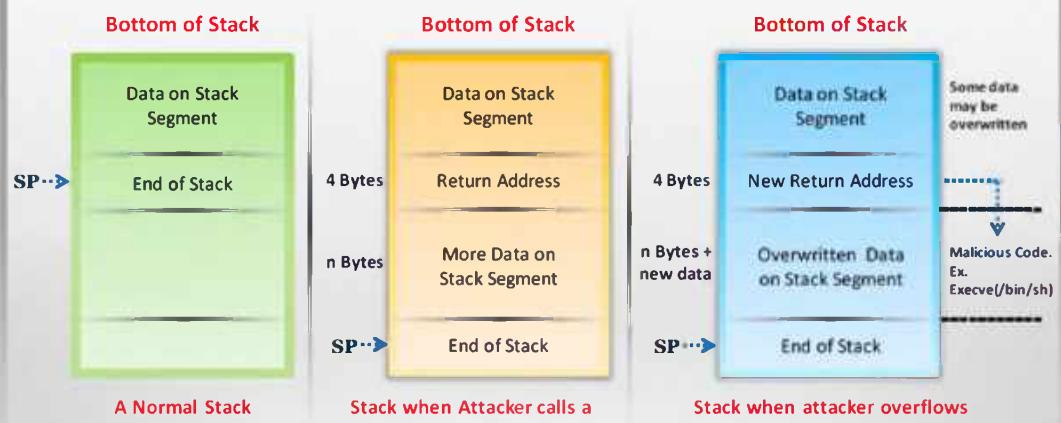


FIGURE 18.2: Stack Segment

Stack-Based Buffer Overflow

 A stack-based buffer overflow occurs when a buffer has been **overrun in the stack space**
Attacker **injects malicious code** on the stack and overflows the stack to overwrite the return pointer so that the flow of control switches to the malicious code



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Stack-based Buffer Overflow

Stack-based buffer overflows have been considered the common type of exploitable programming errors found in software applications. A stack overflow occurs when data is written past a buffer in the stack space, causing unpredictability that can often lead to compromise.

Since in the eyes of the non-security community, stack overflows have been the prime focus of security vulnerability education, these bugs are becoming less prevalent in mainstream software. Nevertheless, they are still important and warrant further examination and ongoing awareness.

Over 100 functions within LibC have security implications. These implications vary from as little as “pseudo randomness not sufficiently pseudorandom” (for example, `srand()`) to “may yield remote administrative privileges to a remote attacker if the function is implemented incorrectly” (for example, `printf()`).

The overflow can overwrite the return pointer so that the flow of control switches to the **malicious code**. C language and its **derivatives** offer many ways to put more data than **anticipated** into a buffer.

Consider an example given as follows for simple **uncontrolled** overflow:

- The program calls the `bof()` function

- Once in the bof () function, a string of 20 As is copied into a buffer that holds 8 bytes, resulting in a buffer overflow

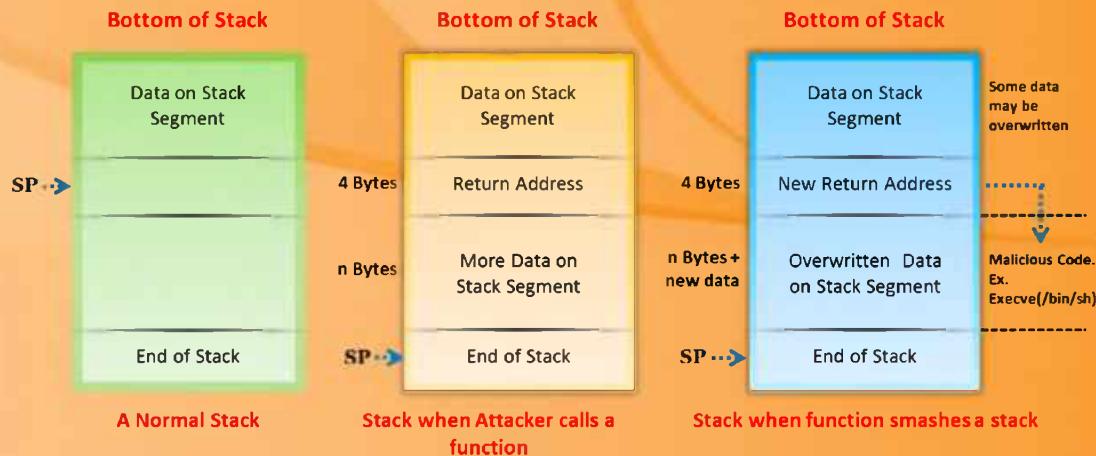


FIGURE 18.3: Stack-based Buffer Overflow

Understanding Heap

CEH
Certified Ethical Hacker

- Heap is a memory segment used by a program and is allocated **dynamically at run time** with functions such as `malloc()`, `calloc()`, `realloc()` in C and using `new` operator in C++
- Control data is stored on the heap along with the data allocated using the **malloc interface**
- Heap stores all instances or attributes, constructors, and methods of a class or object

A = `malloc(10);`
C = `malloc(4);`

Memory Contents
Control Data
Memory Contents
Control Data
Memory Contents
Control Data

Simple Heap Contents

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Understanding Heap

The heap is an area of memory utilized by an application and allocated dynamically at runtime. It is common for a buffer overflow to occur in the heap memory space, and exploitation of these bugs is different from **stack-based buffer overflows**. Heap overflows have been the prominent software security bugs. Unlike stack overflows, heap overflows can be inconsistent and can have varying exploitation techniques and consequences.

Heap memory is different from stack memory; in that heap, memory is persistent between functions, with memory allocated in one function remaining allocated until explicitly freed. This means that a heap overflow can occur, but it is not noticed until that section of memory is used later. There is no concept of saved EIP in relation to a heap, but other important things are stored in the heap and can be broken by overflowing dynamic buffers.

From a primitive point of view, the heap consists of many blocks of memory, some of which are allocated to the program and some are free, but allocated blocks are often placed in adjacent places of memory.

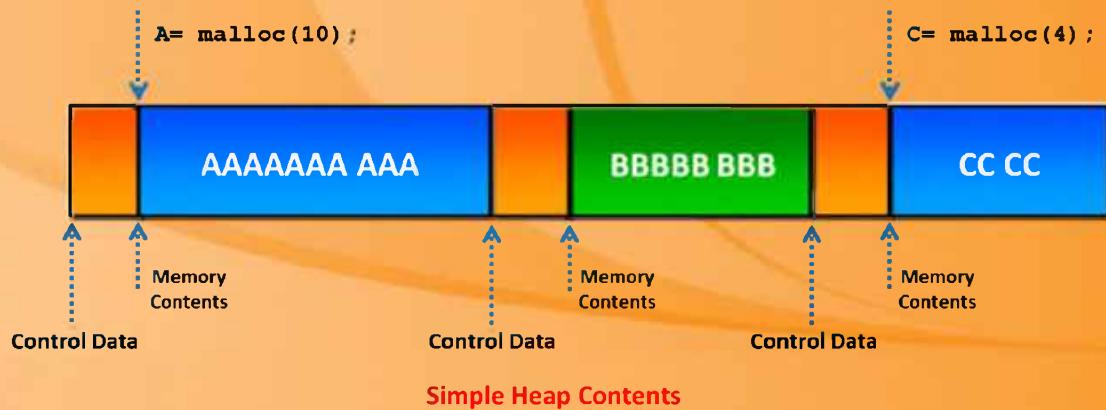


FIGURE 18.4: Understanding Heap

Heap-Based Buffer Overflow

CEH
Certified Ethical Hacker

- If an application copies the data without **checking** whether it fits into the target destination, attackers can supply the application with a large data, overwriting the heap management information
- Attackers overflow buffers on the **lower part of heap**, overwriting other dynamic variables, which can have unexpected and unwanted effects

Heap: Before Overflow

input=malloc(20); output=malloc(20);

XXXXXXXXXXXXXXXXXXXX "normal output\0"

Heap: After Overflow

input=malloc(20); output=malloc(20);

fnordfnordfnordfnordf fno rdfnordfnord\0

Note: In most environments, this may allow the attacker to control the program's execution

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Heap-based Buffer Overflow

The heap is an **area of memory utilized** by an application and allocated dynamically at runtime. It is common for buffer overflows to occur in the heap memory space, and exploitation of these bugs is different from that of stack-based buffer overflows. Heap overflows have been the prominent discovered software security bugs. Unlike stack overflows, heap overflows can be inconsistent and have varying exploitation techniques.

An application dynamically allocates heap memory as needed. This allocation occurs through the function call `malloc ()`. The `malloc ()` function is called with an argument specifying the number of bytes to be allocated and returns a pointer to the allocated memory.

- Variables that are **dynamically allocated** with functions, such as `malloc ()`, are created on the heap.
- An attacker overflows a buffer that is placed on the lower part of heap, overwriting other dynamic variables, which can have **unexpected** and unwanted effects.
- If an application copies data without first **checking** whether it fits into the target destination, the attacker could supply the application with a piece of data that is large, overwriting **heap management** information.
- In most environments, this may allow the attacker to **control** over the program's execution.

Check what happens to the program when input grows past the allocated space. This happens because there is no control over its size. Run the program several times with different input strings.

```
[root@localhost]# ./heap1 hackshacksuselessdata
input at 0x8049728: hackshacksuselessdata
output at 0x8049740: normal output
normal output

[root@localhost]# ./heap1
hacks1hacks2hacks3hacks4hacks5hacks6hacks7hackshackshackshackshack
shacks

input at 0x8049728:
hacks1hacks2hacks3hacks4hacks5hacks6hacks7hackshackshackshackshack
shacks

output at 0x8049740: hackshackshackshacks5hacks6hacks7
hackshacks5hackshacks6hackshacks7

[root@localhost]# ./heap1"hackshacks1hackshacks2hackshacks3hackshacks4wh
at have I done?"

Input at 0x8049728: hackshacks1hackshacks2hackshacks3hackshacks4 what
have I done?

output at 0x8049740: what have I done?
what have I done?

[root@localhost]#
```

Thus, overwriting variables on the heap is easy and does not always cause crashes.

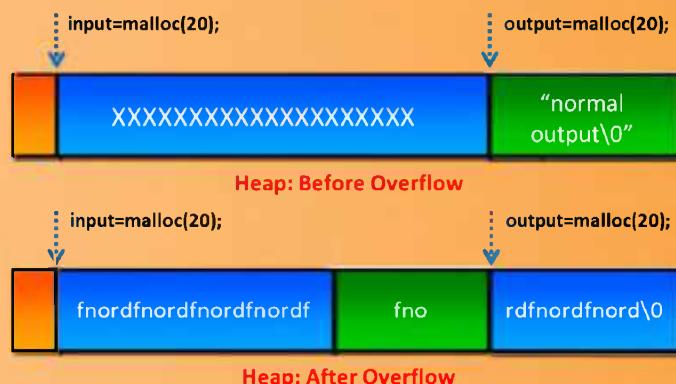
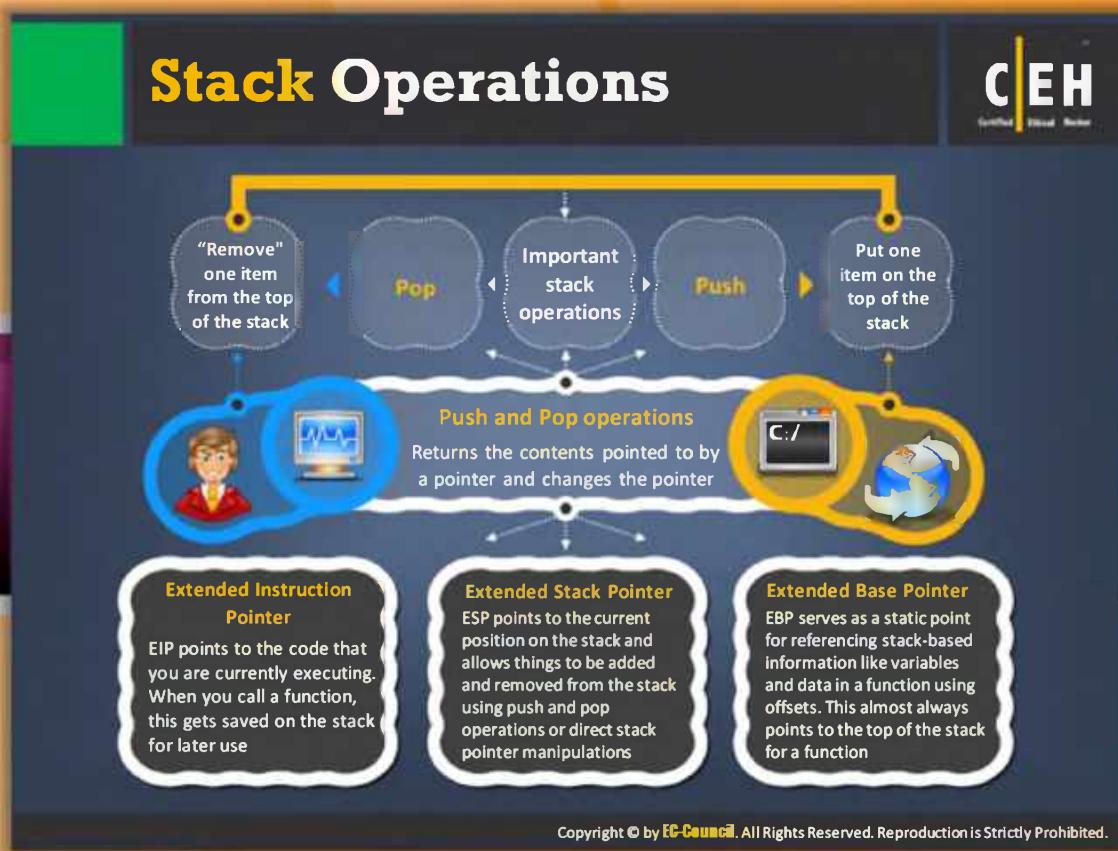


FIGURE 18.5: Heap-based Buffer Overflow



Stack Operations

A stack is implemented by the system for programs that run on the system. A variable can be deployed within the processor itself and memory can also be allocated. The variable is called the “**register**” and the region of memory is the “**stack**.” The register used to refer to the stack as the “**Stack Pointer**” or SP. The SP points to the top of the stack, while the bottom of the stack is a fixed address. The kernel adjusts the stack size **dynamically** at run time.

A stack frame, or record, is an activation record that is stored on the stack. The stack frame has the following: the parameters to a function, its local variables, and the data required to restore the previous stack frame, along with the value of the instruction pointer (pointer that points the next instruction to be **fetched** at the function call) at the time of the function call.

The majority functionality of the stack involves adding and removing items from the stack. This is **accomplished** with the help of two major operations. They are **push** and **pop**.

When the program is loaded, the stack pointer is set to the highest address. This will be the topmost item in the stack. When an item is pushed onto the stack, two events take place. **Subtracting** the size of the item in bytes from the initial value of the **pointer** reduces the stack pointer. Next, all the **bytes** of the items in consideration are copied into the region of the stack segment to which the stack pointer now points.

Similarly, when an item is popped from the stack, the size of the item in **bytes** is added to the stack pointer. However, the copy of the item continues to reside on the stack. This will eventually be overwritten when the next push operation takes place. Based on the **stack design** implementation, the **stack** can come down (toward lower memory addresses) or go up (toward higher memory addresses).

When a procedure is called, it is not the only item that pushes onto the stack. Among others is the address of the calling procedure's instruction immediately following the procedure call. This is followed by the **parameters** to the called function. As the called function completes, it would have **popped** its own local variables off the stack. The last **instruction** the called function runs is a special instruction called a return. The top values of the stack are popped up and loaded into the IP by the assembly language, a **special processor** instruction. At this point, the stack will have the address of the next instruction of the calling procedure in it. The other concept that the reader needs to **appreciate** in order to understand the complete essence of **stack overflows** is the **frame pointer**.

Apart from the stack pointer, which points to the top of the stack, there is a **frame pointer (FP)** that points to a **fixed location** within a frame. Local variables are usually referenced by their offsets from the stack pointer. However, as the stack operations take place, the value of these offsets vary. Moreover, on processors such as **Intel-based processors**, accessing a variable at a known distance from the stack pointer requires multiple instructions. Therefore, a second register may be used for **referencing** those variables and parameters whose relative distance from the frame pointer does not change with stack operations. On Intel processors, the **base pointer (BP)**, also known as the **Extended Base Pointer (EBP)**, is used.

Shellcode

Shellcode refers to code that can be used as payloads in the exploitation of a software vulnerability

Buffers are soft targets for attackers as they overflow easily due to poor coding techniques

Buffer overflow shellcodes, written in machine language, exploit vulnerabilities in stack and heap memory management

Example

```
\x2d\x0b\xd8\x9a\xac\x15\x1a\x6e\x2f\x0b\xdc\xda\x90\x0b\x80\x0e"
\x92\x03\x0a\x08\x94\x1a\x80\x0a\x9c\x03\x0a\x10\xec\x3b\xbf\xf0"
\xdc\x23\xbf\xf8\xc0\x23\xbf\xfc\x82\x10\x20\x3b\xaa\x10\x3f\xff"
\x91\xd5\x60\x01\x90\x1b\xc0\x0f\x82\x10\x20\x01\x91\xd5\x60\x01"
```

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Shellcode

Shellcode is a small code used as **payload** in the exploitation of a software vulnerability. Shellcode is a technique used to **exploit stack-based overflows**. Shellcodes **exploit** programming bugs in **stack handling**. Buffers are soft targets for attackers as they overflow easily if the conditions match. Buffer overflow shellcodes, written in **assemble language**, exploit vulnerabilities in stack and **heap memory management**.

For example, the VRFY command helps the attacker to identify potential users on the target system by verifying their email addresses. In addition, sendmail uses a set user ID of root and runs with **root privileges**. If the attacker connects to the sendmail daemon and sends a block of data consisting of 1,000 a's to the **VRFY command**, the VRFY buffer is overrun as it was only designed to hold **128 bytes**.

However, instead of sending 1000 a's, the attacker can send a specific code that can overflow the buffer and execute the command /bin/sh. In this case, when the attack is carried out, a special assembly code "**egg**" is transferred to the VRFY command, which is a part of the actual string used to overflow the buffer. When the VRFY buffer is overrun, instead of the **offending** function returning to its original **memory address**, the attacker executes the **malevolent machine code** that was sent as a part of the buffer overflow data, which executes /bin/sh with root privileges.

The following illustrates what an egg, specific to Linux X86, looks like:

```
Char shellcode [ ] =  
"\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"  
"\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"  
"\x80\xe8\xdc\xff\xff\xff/bin/sh";
```

No Operations (NOPs)

CEH
Certified Ethical Hacker

Most CPUs have a **No Operation (NOP)** instruction -- it does nothing but advance the instruction pointer

Most intrusion detection systems (IDSs) look for signatures of NOP sleds

Attacker changes identified IP to start anywhere within NOP area to create multiple zero-day infections (polymorphism)

1 2 3 4 5

Attacker pads the beginning of the intended buffer overflow with a long run of NOP instructions (a NOP slide or sled) so the CPU will do nothing until it gets to the "main event" (which preceded the "return pointer")

ADMmutate (by <http://www.ktwo.ca>) accepts a buffer overflow exploit as input and randomly creates a functionally equivalent version
Note: It is the NOP sled that ADMutate mutates (not the shellcode)

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



No Operations (NOPs)

Even the best guess may not be good enough for an attacker to find the **right address** on the stack. If the attacker is off by one byte, more or less, there can be a segmentation violation or an **invalid instruction**. This can even cause the system to crash. The attacker can increase the odds of finding the right address by **padding** his or her code with **NOP instructions**.

A NOP is just a command telling the **processor** to do nothing other than take up time to process the **NOP instruction** itself. Almost all processors have a NOP instruction that performs a null operation. In the Intel architecture, the NOP instruction is **1 byte** long and translates to **0x90** in **machine code**. A long run of NOP instructions is called a **NOP slide** or sled, and the CPU does nothing until it gets back to the main event (which precedes the "return pointer").

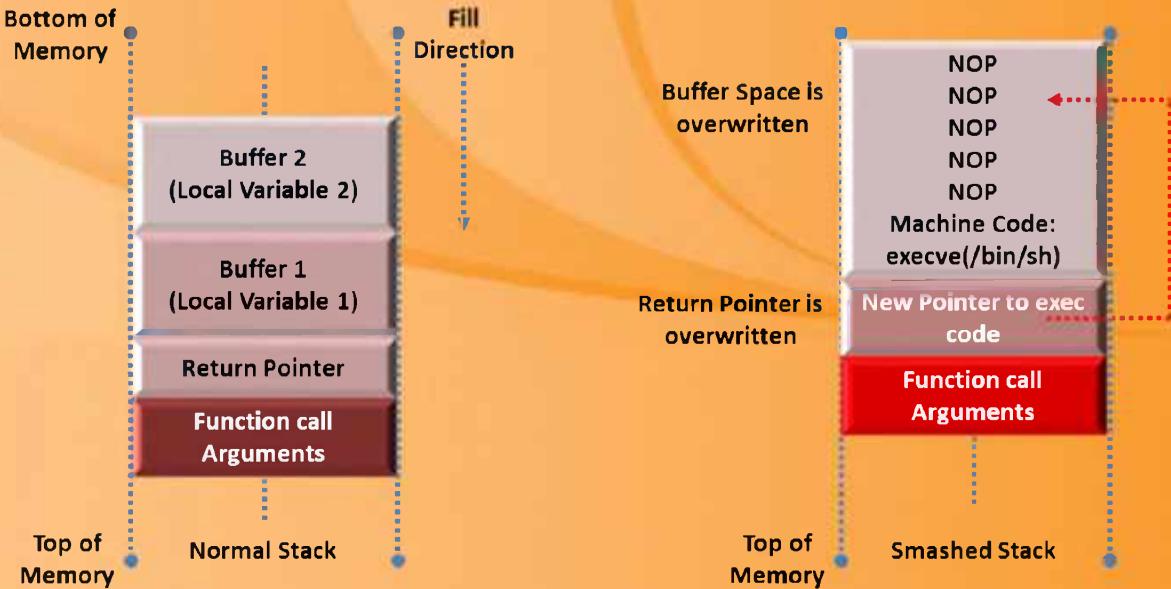
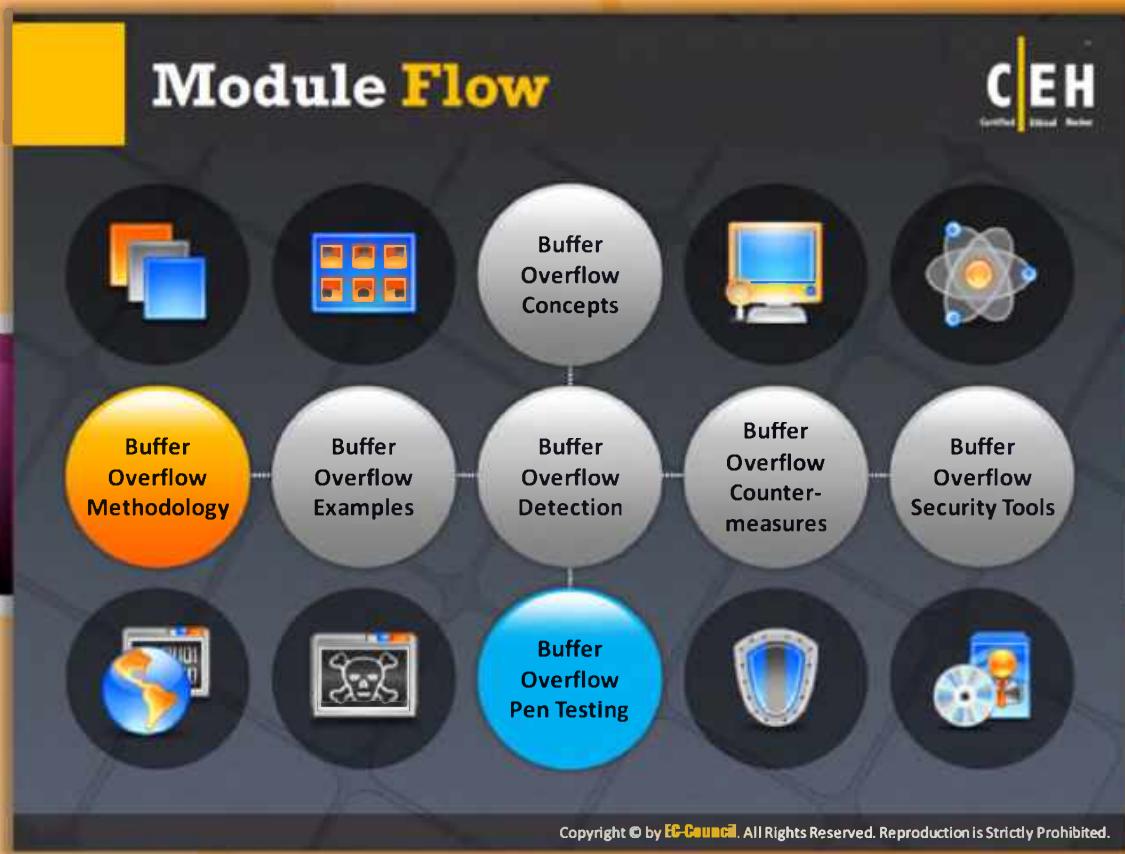


FIGURE 18.6: No Operations (NOPs)

By including NOPs in advance of the **executable code**, the attacker can avert a **segmentation violation** if an overwritten return pointer lands execution in the NOPs. The program can continue to execute down the stack until it gets to the attacker's exploit. In the **preceding illustration**, the attacker's data is written into the allocated buffer by the function. As the data size is not checked, the return pointer can be overwritten by the attacker's input. With this method, the attacker places the **exploited machine's code** in the buffer and overwrites the return pointer so that when the function returns, the attacker's code is executed.

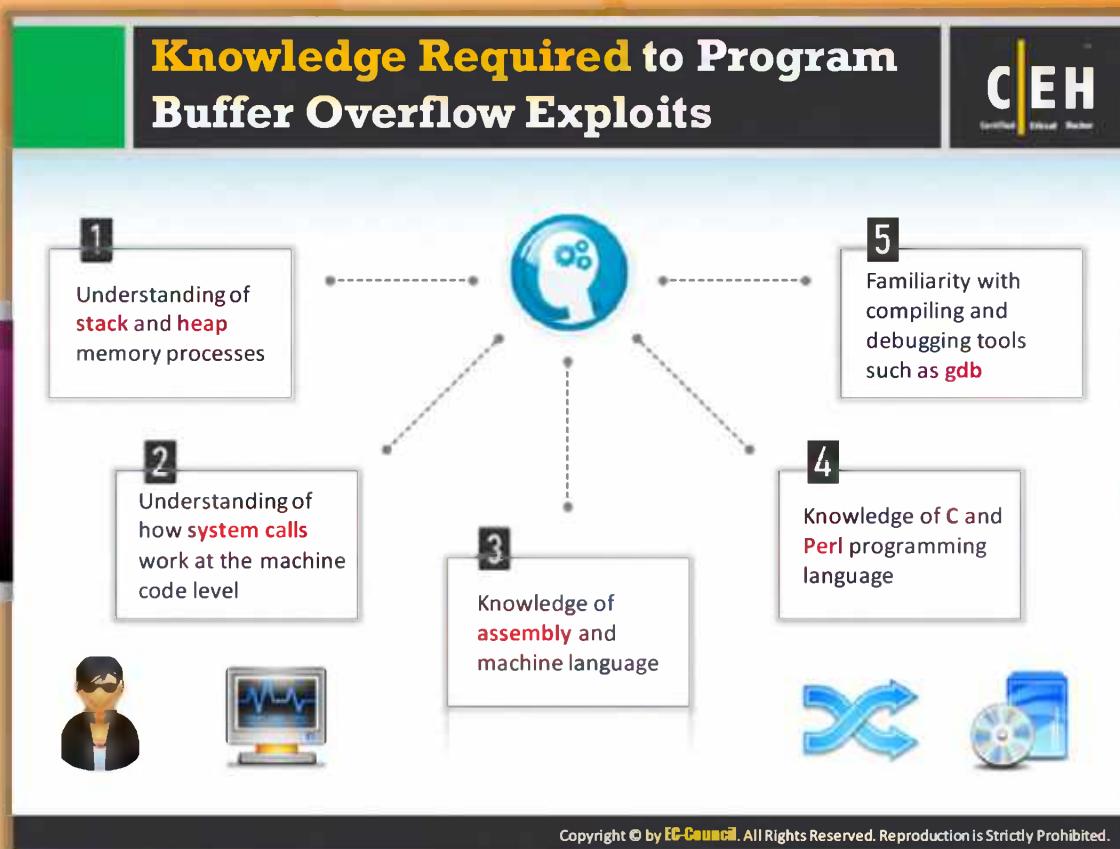


Module Flow

So far, we have discussed the basic buffer overflow concepts, Now we will discuss the buffer overflow methodology.

	Buffer Overflow Concepts		Buffer Overflow Countermeasures	
	Buffer Overflow Methodology		Buffer Overflow Security Tools	
	Buffer Overflow Examples		Buffer Overflow Pen Testing	
	Buffer Overflow Detection			

This section describes requirements to program buffer overflow exploits, buffer overflow steps, and buffer overflow vulnerabilities.



Knowledge Required to Program Buffer Overflow Exploits

Logically, the question that arises is why are stacks used when they pose such a **threat to security**? The answer lies in the **high-level, object-oriented programming languages**, where procedures or functions form the basis of every program.

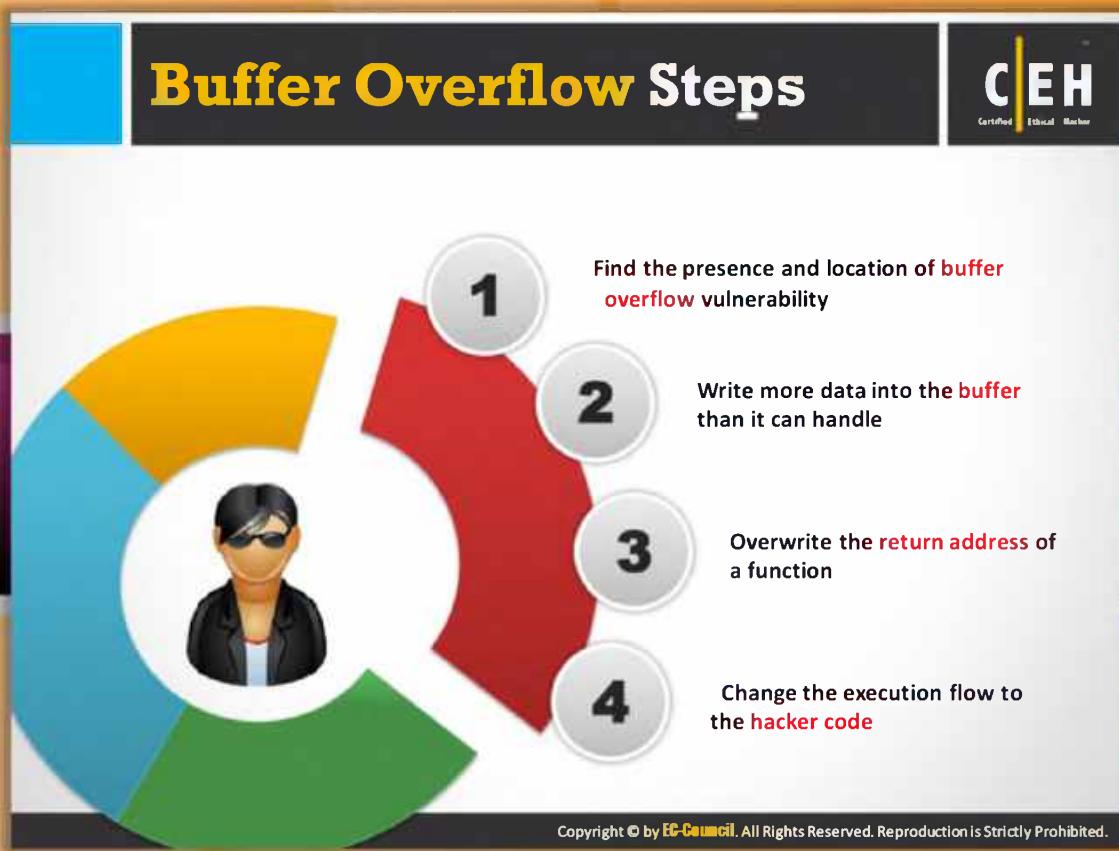
A stack is used for **storing context**. For instance, if a procedure simply pushes all its local variables onto the **stack** when it enters, and pops those off when it is over, its entire context is cleaned up so that when the procedure calls another procedure, the called procedure can do the same with its context, without the aid of the **calling procedure's** data. The flow of control is determined by the procedure or function, which is resumed after the current one is done. The stack implements the **high-level abstraction**. Apart from this, the stack also serves to **dynamically allocate** local variables used in functions, passing parameters to functions, and to return values from the function.

In fact, though several applications are written in C, programs written in C are particularly susceptible to **buffer overflow attacks**. This is due to the fact that direct pointer variations are permitted in C. Direct, **low-level memory access** and the **pointer arithmetic** is provided by C without checking the bounds. Moreover, the standard C library provides unsafe functions (such

as gets) that write an **unbounded** amount of user's input into a **fixed size buffer** without any **boundary checking**.

To program buffer overflow exploits, you should be **acquainted** with the following aspects:

- ⊕ Understanding of stack and heap memory processes
- ⊕ Understanding of how **system calls** work at the machine code level
- ⊕ Knowledge of assembly and machine language
- ⊕ Knowledge of C and Perl programming language
- ⊕ **Familiarity with compiling and debugging tools** such as gdb
- ⊕ exec() system calls How to guess some key parameters
- ⊕ How to guess some key parameters



Buffer Overflow Steps

Buffer overflow can be carried out in four steps:

Step 1: In order to perform a **buffer overflow attack**, first you should check whether the **target** application or program is **vulnerable** to buffer overflow or not. Typically buffer overflow occurs when the input entered **exceeds** the size of the buffer. If there is any potential buffer overflow vulnerability present in the program, then it displays an error when you enter a **lengthy string** (exceeding the size of buffer). Thus, you can **confirm** whether a program contains a buffer overflow vulnerability or not. If it is vulnerable, then find the location of the buffer overflow vulnerability.

Step 2: Once you find the location of the **vulnerability**, write more data into the buffer than it can handle. This causes the buffer overflow.

Step 3: When a buffer overflow occurs, it overwrites the memory. Using this advantage, you can overwrite the return address of a function with the address of the shellcode.

Step 4: When the **overwrite** occurs, the **execution flow** changes from normal to the **shell code**. Thus, you can execute anything you want.

Attacking a Real Program

Assuming that a **string function is exploited**, the attacker can send a long string as the input. This string overflows the buffer and causes a **segmentation error**

The return pointer of the function is **overwritten**, and the attacker succeeds in altering the flow of the execution

If the attacker inserts code as input, he or she has to know the **exact address** and **size** of the stack and make the return pointer point to the code for execution

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Attacking a Real Program

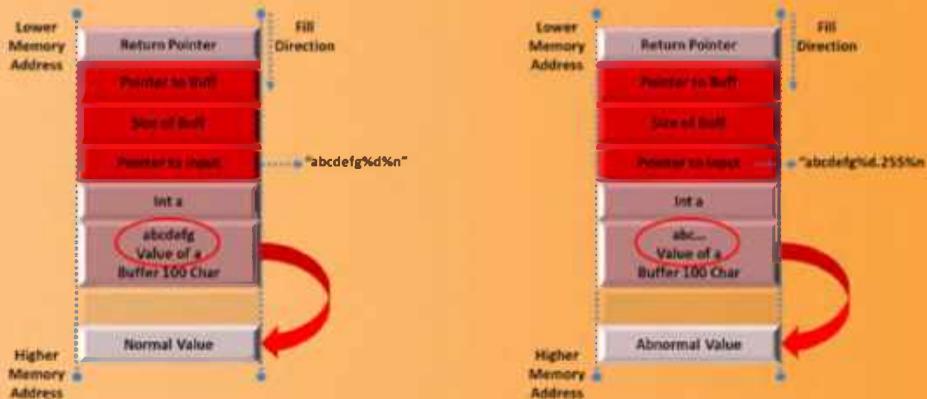


FIGURE 18.7: Attacking a Real Program

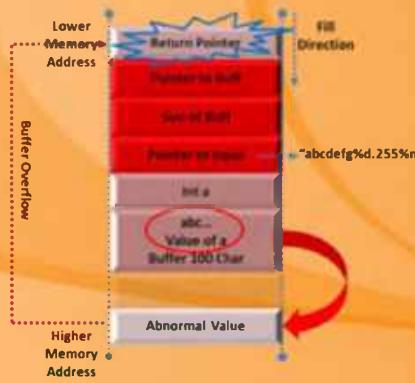


FIGURE 18.8: Attacking a Real Program

The previous illustration depicts the way an **abnormal input** causes the buffer to overflow and cause a **segmentation error**. Eventually, the return pointer is overwritten and the execution flow of the function is **interrupted**. Now, if the attacker wants to make the function **execute arbitrary code** of his or her choice, he or she can have to make the return pointer point towards this code.

When attacking a real program, an attacker has to assume that a string function is being exploited, and send a **long string** as the input. After passing the input string, the string overflows the buffer and causes a segmentation error. The return **pointer** of the function is overwritten, and the attacker succeeds in altering the **flow of execution**. If the user has to insert his or her code in the input, he or she has to:

- ⊕ Know the **exact address** on the stack.
- ⊕ Know the size of the stack.
- ⊕ Make the **return pointer** point to his/her code for execution.

The challenges that the attacker faces are:

- ⊕ Determining the **size** of the buffer.
- ⊕ The attacker must know the address of the stack so that he or she can get his or her input to rewrite the return pointer. For this, he or she must **ascertain** the exact address.
- ⊕ The attacker must write a program small enough that it can be passed through as input.

Usually, the goal of the attacker is to **spawn a shell** and use it to **direct further commands**.

The code to spawn a shell in C is as follows:

```
#include <stdio.h>
void main() {
    char *name[2];
    name[0] = "/bin/sh";
    name[1] = NULL;
    execve(name[0], name, NULL); }
```

The attacker can place **arbitrary code** to be executed in the buffer that is to be **overflown** and overwrite the return address so that it points back into the buffer. For this, he or she must know the exact location in the memory space of the program whose code has to be exploited. A workaround for this challenge is to use a **jump (JMP)** and a **CALL instruction**. These instructions allow **relative addressing** and permit the attacker to point to an offset relative to the instruction pointer. This **eliminates** the need to know the exact address in the memory to which the exploit code must point. As most **operating systems** mark the code pages with the **read-only** attribute, this makes the previously discussed workaround an **unfeasible** one. The alternative is to place the code to be executed into the stack or **data segment** and transfer control to it. One way of **achieving** this is to place the code in a **global array** in the data segment as shown in the previous code snippet.

Does the exploit work? Yes.

Nevertheless, in maximum **buffer overflow vulnerabilities**, it is the character buffer that is subjected to the attack. Therefore, any null code occurring in the shell code can be considered as the end of the string, and the **code transfer** can be terminated. The answer to this **hindrance** lies in NOP.

Format String Problem



In C, consider this example of Format string problem:

```
int func(char *user)
{
    fprintf( stdout, user);
}
```

Problem if user = "%s%s%s%s%s%s%"

Most likely program will crash causing a DoS
If not, program will print memory contents
Similar exploit occurs using user = "%n"

Correct form is:

```
int func(char *user)
{
    fprintf( stdout,
        "%s", user); }
```

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Format String Problem

Format string problems usually occur when the input is supplied from untrusted sources or when the data is passed as a **format string argument** to functions such as `syslog()`, `sprintf()`, etc. The format string vulnerabilities in **C/C++** can easily be exploited because of the **%n operator**. If any program contains this kind of vulnerability, then the **program's confidentiality** and the **access control** may be at risk because the format string vulnerability exploitation results in information disclosure and **execution of arbitrary code** without the knowledge. Thus, attackers can easily exploit the program or application containing format string **vulnerabilities**.

In C, consider this example of a format string problem:

```
int func(char *user)
{
    fprintf( stdout, user);
}
```

Problem if user = "%s%s%s%s%s%s%"

Most likely, the program will crash, causing a DoS. If not, the program will print memory contents. Full exploit occurs using user = "%n"

Correct form is:

```
int func(char *user)
{
    fprintf( stdout, "%s", user);
}
```

Overflow Using Format String

In C, consider this example of BoF using format string problem:

```
char errmsg[512],  
outbuf[512];  
sprintf (errmsg, "Illegal  
command: %400s", user);  
sprintf( outbuf, errmsg );
```

What if user = "%500d
<nops> <shellcode>"

- Bypasses "%400s" limitation
- Will overflow outbuf



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Overflow Using Format String

In C, the format string library functions take **variable numbers of arguments**. The format string variable is the one that is always required. The format string contains **format-directive characters** and printable characters. **Format string overflow attacks** are quite similar to that of buffer overflow attacks since in both attacks the attacker attempts to change the memory space and **consequently runs arbitrary code**. The only difference between these two attacks is that the attacker launches the format string overflow attack by **exploiting** the vulnerabilities in the **variadic functions**, such as format functions.

Format string overflow can be exploited in four ways:

- Memory viewing
- Updating a word present in the memory
- Making a buffer overflow by using minimum field size specifier
- Using %n **format directive** for overwriting the code

In C, consider this example of BoF using format string problem:

```
char errmsg[512], outbuf[512];  
sprintf (errmsg, "Illegal command: %400s", user);
```

```
sprintf( outbuf, errmsg );
```

If user = "%500d <nops> <shellcode>", this will bypass "%400s" limitation and overflow outbuf. Thus, the stack smashing buffer overflow attack is carried out.

Smashing the Stack

The general idea is to overflow a buffer so that it overwrites the return address

When the function is done, it will jump to whatever address is on the stack

Put some code in the buffer and set the return address to point to it

Buffer overflow allows us to change the return address of a function

Copyright © by EC-Council®. All Rights Reserved. Reproduction is Strictly Prohibited.



Smashing the Stack

Smashing the stack causes a **stack to overflow**. The stack is a **first-in last-out form** of buffer to hold the **intermediate results** of an operation. If you try to store more data than the stack's size, then it drops the excess data. The data that a stack holds may be critical for **system operation**.

The general idea behind **stack smashing** is to overflow a buffer which in turn overwrites the return address. If the attacker succeeds in smashing the stack, then he or she can overwrite the address on the stack with the address of **shellcode**. When the function is done, it jumps to the return address, i.e., the shellcode address. Thus an attacker can exploit the **buffer overflow vulnerability**.

The diagram is titled "Once the Stack is Smashed..." and features a central title bar with the CEH logo. Below it, a large arrow points from left to right, divided into three main sections: "Gain Access", "Create a backdoor", and "Use Netcat".

- Gain Access:** Describes how once the vulnerable process is commandeered, the attacker has the same privileges as the process and can gain normal access.
- Create a backdoor:** Lists methods for creating a backdoor:
 - Using (UNIX-specific) inetd
 - Using Trivial FTP (TFTP) included with Windows 2000 and some UNIX flavors
- Use Netcat:** Describes how Netcat can be used to make raw and interactive connections:
 - UNIX-specific GUI
 - Shoot back an Xterminal connection

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

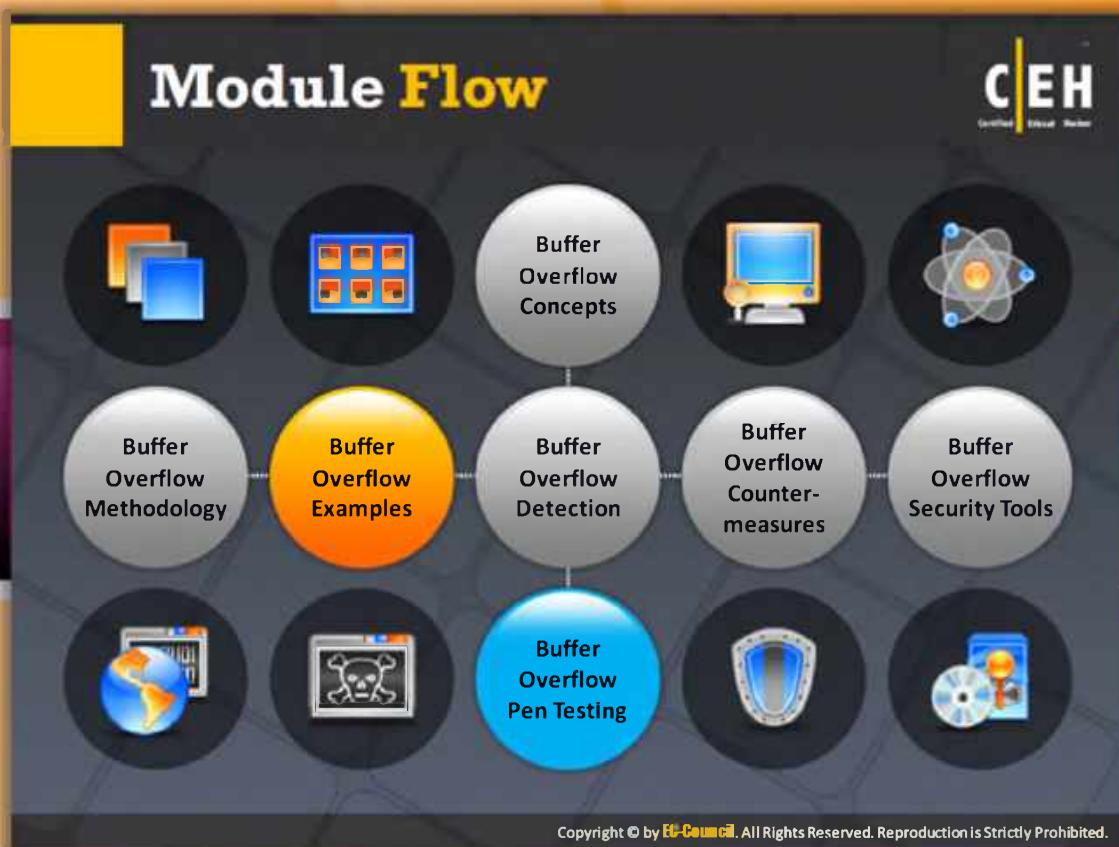


Once the Stack is Smashed

There are two parts of the **attacker's input**: an injection vector and a **payload**. They may be separate or put together. The **injection vector** is the correct entry-point that is tied unambiguously along with the bug itself. It is **OS/target/application/protocol/encoding-dependent**. On the other hand, the payload is usually not tied to bugs at all and is contained by the attacker's **ingenuity**. Even though it can be **independent** of the **injection vector**, it still depends on machine, processor, and so on.

Once the stack is smashed, the attacker can deploy his or her payload. This can be anything. For example, in **UNIX**, a command shell can be spawned. For example, with /bin/sh in Windows NT/2000 and a specific **Dynamic Link Library (DLL)**, external stacks may be preferable and may be used for further probing. For example, WININET.DLL can be used to send requests to and get information from the network and to download code or **retrieve commands** to execute.

The attacker may launch a **denial-of-service attack** or he or she may use the system as a **launching point** (ARP spoofing). The common attack is to spawn a remote shell. The exploited system can be converted into a covert channel or it can simulate Netcat to make a raw, **interactive connection**. The payload can be a worm that **replicates** itself and searches for fresh targets. The attacker can also eventually install a **rootkit** and remain in **stealth mode** after gaining super-user access.



Module Flow

So far, we have discussed buffer overflow concepts and the methodology. Now it's time to see buffer overflow examples.

Buffer Overflow Concepts	Buffer Overflow Countermeasures
Buffer Overflow Methodology	Buffer Overflow Security Tools
Buffer Overflow Examples	Buffer Overflow Pen Testing
Buffer Overflow Detection	

This section covers various buffer overflow examples.

Simple Uncontrolled Overflow



Example of Uncontrolled Stack Overflow

```
/* Program to show a simple uncontrolled overflow of the stack*/
1: #include <stdlib.h>
2: #include <stdio.h>
3: #include <string.h>
4: int buffer(){
5:     char buff[12];
6:     strcpy(buff,"DDDDDDDDDDDDDDDD"); /*copy 18 bytes of D into the buffer*/
7:     return 1; /*this causes an access violation due to stack corruption.*/
8: }
9: int main(int argc, char **argv){
10:    buffer(); /*function call*/
11:    /*print a short message, execution will never reach this point because of the overflow*/
12:    printf("Let's Go\n");
13:    return 1; /*leaves the main function*/
}
```

Example of Uncontrolled Heap Overflow

```
/* Program to show a simple heap overflow*/
1: #include <stdio.h>
2: #include <stdlib.h>
3: int main(int argc, char *argv[])
4: {
5:     char *in = malloc (18);
6:     char *out = malloc (18);
7:     strcpy (out, "Sample output");
8:     strcpy (in, argv[1]);
9:     printf ("input at %p: %s\n", in, in);
10:    printf ("output at %p: %s\n", out, out);
11:    printf ("\n\n%s\n", out);
12: }
```



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Simple Uncontrolled Overflow

Example of Uncontrolled Stack Overflow

```
/* stack3.c
This is a program to show a simple uncontrolled overflow of the stack. It will overflow EIP with 0x41414141, which is AAAA in ASCII.
*/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int bof()
{
    char buffer[8]; /* an 8 byte character buffer */
    /*copy 20 bytes of A into the buffer*/
    strcpy(buffer,"AAAAAAAAAAAAAAA");
    /*return, this will cause an access violation due to stack corruption.
    We also take EIP*/
    return 1;
}
```

```
}

int main(int argc, char **argv)
{
    bof(); /*call our function*/
    /*print a short message, execution will never reach this point because
    of the overflow*/
    printf("Not gonna do it!\n");
    return 1; /*leaves the main function*/
}
```

The main function in this program calls the bof() function. In the first line of bof() **function code** a buffer of char type with 8-bit size is initiated. Later a string of 20 As is copied into the buffer. This causes the **buffer overflow** because the size of buffer is just 8 bits,whereas the string copied into the buffer is 20 bits. This leads to an **uncontrolled overflow**.

Example of Uncontrolled Heap Overflow

The following code is an example of uncontrolled head overflow.

```
/*heap1.c - the simplest of heap overflows*/
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    char *input = malloc (20);
    char *output = malloc (20);
    strcpy (output, "normal output");
    strcpy (input, argv[1]);
    printf ("input at %p: %s\n", input, input);
    printf ("output at %p: %s\n", output, output);
    printf("\n\n%s\n", output);
}
```

Simple Buffer Overflow in C



Command Prompt

```
#include <stdio.h>
main() {
    char *name;
    char *dangerous_system_command;
    name = (char *) malloc(10);
    dangerous_system_command = (char *) malloc(128);
    printf("Address of name is %d\n", name);
    printf("Address of command is %d\n", dangerous_system_command);
    sprintf(dangerous_system_command, "echo %s", "Hello world!");
    printf("What's your name?");
    gets(name);
    system(dangerous_system_command);
}
```

- The first thing the program does is **declare** two string variables and assign memory to them
- The "name" variable is given **10 bytes** of memory (which will allow it to hold a 10-character string)
- The "dangerous_system_command" variable is given **128 bytes**
- You have to understand that, in C, the memory chunks given to these variables will be located directly next to each other in the virtual memory space given to the program



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Simple Buffer Overflow in C

To understand how buffer overruns works, you need to look at the small C program that follows:

```
#include <stdio.h>
main() {
    char *name;
    char *dangerous_system_command;
    name = (char *) malloc(10);
    dangerous_system_command = (char *) malloc(128);
    printf("Address of name is %d\n", name);
    printf("Address of command is %d\n", dangerous_system_command);
    sprintf(dangerous_system_command, "echo %s", "Hello world!");
    printf("What's your name?");
    gets(name);
    system(dangerous_system_command);
}
```

This program is designed to be run by a user on a console, but it illustrates the trouble that a poorly written **network daemon** can cause.

The first thing the program does is to declare two string variables, and assign memory to them. The "name" variable is given 10 bytes of memory (which will allow it to hold a 10-character string). The "**dangerous_system_command**" variable is given 128 bytes. The thing you have to understand is that in C, the memory chunks given to these variables will be located directly next to each other in the **virtual memory space** given to the program. If you run the program with a short name, you can see how things are supposed to work:

```
[jturner@secure jturner]$ ./overrun
Address of name is 134518696
Address of command is 134518712
What's your name?James
Hello world!
[jturner@secure jturner]$
```

As you can see, the address given to the "dangerous_system_command" variable is 16 bytes from the start of the "name" variable. The extra 6 bytes are overhead used by the "malloc" system call to allow the memory to be returned to **general usage** when it is freed.

Simple Buffer Overflow in C: Code Analysis

The "gets" command, which reads a string from the standard input to the specified memory location, does not have a "length" specification. This means it will read as many characters as it takes to get to the end of the line, even if it overruns the end of the memory allocated. Knowing this, an attacker can overrun the "name" memory into the "dangerous_system_command" memory, and run whatever command he or she wishes.

To compile the overrun.c program, run this command in Linux:

```
gcc overrun.c -o overrun
[xx] $ ./overrun
Address of name is 134518696
Address of command is 134518712
What's your name?xmen
Hello world!
[xx] $
```

The address given to the "dangerous_system_command" variable is 16 bytes from the start of the "name" variable. The extra 6 bytes are overhead used by the "malloc" system call to allow the memory to be returned to general usage when it is freed.

Buffer Overrun Output

```
[xx] $ ./overrun
Address of name is 134518696
Address of command is 134518712
What's your
name?0123456789123456cat/etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail
```

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Simple Buffer Overflow in C: Code Analysis

After allocating the memory and printing the **memory locations** of the two variables, the program generates a **command** that will later be sent to the "**system**" call, which causes it to be executed as if it had been typed at a keyboard. In this case, all it does is print "Hello world!" Then, it prompts the user for his or her name and reads it using the "gets" system call. In a real **network daemon**, this might be printing a prompt and **awaiting** a command from the client program such as a website address or email address.

The important thing to know is that "**gets**," which reads a string from standard input to the specified memory location, DOES NOT have a "length" specification. This means it will read as many characters as it takes to get to the end of the line, even if it **overruns** the end of the memory allocated. Knowing this, a hacker can overrun the "name" memory into the "**dangerous_system_command**" memory, and run whatever command they wish. For example:

```
[jturner@secure jturner]$ ./overrun
Address of name is 134518696
Address of command is 134518712
What's your name?0123456789123456cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

```
bin:x:1:1:bin:/bin:  
daemon:x:2:2:daemon:/sbin:  
adm:x:3:4:adm:/var/adm:  
lp:x:4:7:lp:/var/spool/lpd:  
sync:x:5:0:sync:/sbin:/bin/sync  
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown  
halt:x:7:0:halt:/sbin:/sbin/halt  
mail:x:8:12:mail:/var/spool/mail
```

By **padding** out the response to the name query to 16 character and then adding a system command, the system command overwrites "echo Hello World!" with "cat /etc/passwd". As you can see, this causes that command to be run instead of the appropriate one.

So what can be done to prevent this? First, use the fgets system call, which specifies a maximum length, will eliminate the possibility altogether. By changing the "gets" call to:

```
fgets(name, 10, stdin);
```

The problem is solved:

```
[jturner@secure jturner]$ ./overrun  
Address of name is 134518768  
Address of command is 134518784  
What's your name?01234567890123456cat /etc/passwd  
Hello world!  
[jturner@secure jturner]$
```

But, since many sites run software that they do not have source code to (commercial databases, for example), you cannot protect yourself from all **buffer overruns**. The other important step you need to take is to turn off any network services you do not use, and only run the ones you do use at a **permission** level that meets the needs of the program. For example, do not run a **database** as root; give it its own user and group. That way, if it is exploited, it cannot be used to take over the system.

Buffer overruns are one of those things that every first-year **programming** student should be taught to avoid. That attackers still use it with such **frequency** is an indication of how far we have to go in the quest for truly **reliable** and **secure software**.

Exploiting Semantic Comments in C (Annotations)



Adding "@ after the /*"

- Adding "@" after the "/*" which is considered a comment in C) is recognized as syntactic entities by the LCLint tool
- So, in a parameter declaration, it indicates that the value passed for this parameter may not be NULL
- Example: /*@ this value may not be null@*/

Annotations can be defined by LCLint using clauses

- Describe assumptions about buffers that are passed to functions
- Constrain the state of buffers when functions return; assumptions and constraints used in the example below: minSet, maxSet, minRead, and maxRead



```
char *strcpy (char *s1, const char *s2)
/*@requires maxSet(s1) >= maxRead(s2)@*/
/*@ensures maxRead(s1) == maxRead(s2)
\ result == s1@*/;rr
```



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Exploiting Semantic Comments in C (Annotations)

Though many run-time approaches have been proposed to mitigate the risk associated with buffer overflows, they are not widely used. Hence, static analysis of a program's source code came into practice to detect buffer overflows. This can be accomplished with the help of the LCLint tool. It performs static detection of buffer overflows by exploiting semantic comments added to the source code. Thus, it enables local checking of interprocedural properties.

Adding "@" after the "/*"

- Adding "@" after the "/*" which is considered a comment in C) is recognized as syntactic entities by the LCLint tool
- So, in a parameter declaration, it indicates that the value passed for this parameter may not be NULL
- Example: /*@ this value need not be null@*/

Annotations can be defined by LCLint using clauses

- Describe **assumptions** about buffers that are passed to functions
- Constrain the state of buffers when functions return **assumptions** and **constraints** used in the example below: minSet, maxSet, minRead, and maxRead

```
char , const char *s2)
/*@requires maxSet(s1) >= maxRead(s2)@*/
/*@ensures maxRead(s1) == maxRead(s2)
 \ result == s1@*/;rr
```

How to Mutate a Buffer Overflow Exploit

C|EH Certified Ethical Hacker

For the NOP Portion	For the “Main Event”	For the “Return Pointer”
<ul style="list-style-type: none">Randomly replace the NOPs with functionally equivalent segments of the code (e.g.: <code>x++;</code> <code>x-;</code> ? <code>NOP NOP</code>)  	<ul style="list-style-type: none">Apply XOR to combine code with a random key unintelligible to IDS. The CPU code must also decode the gibberish in time to execute payload. The XORing makes the payload polymorphic and therefore hard to spot 	<ul style="list-style-type: none">Randomly tweak LSB of the pointer to land in the NOP-zone 

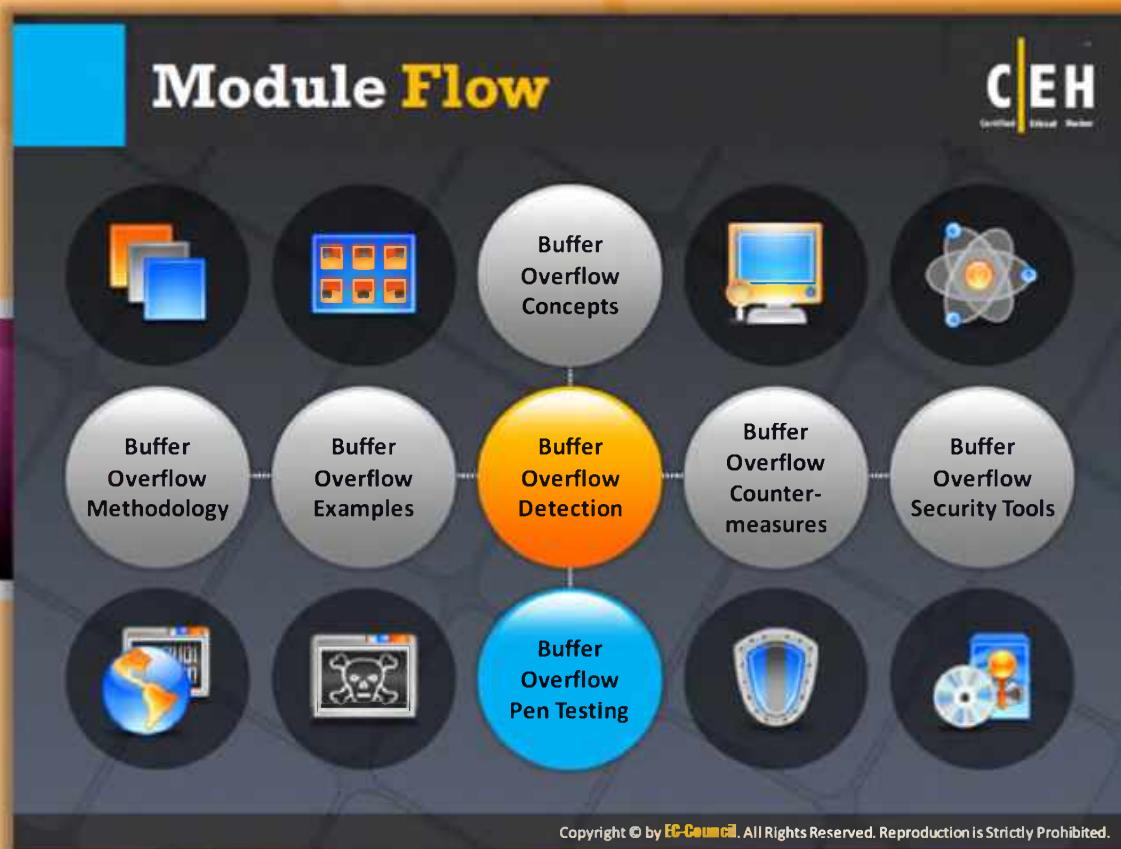
Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



How to Mutate a Buffer Overflow Exploit

Most IDSe look for signs of **NOP sleds**. Detecting an array of **NOPs** can be indicative of a buffer overflow exploit across the network. **ADMutate** takes the concept a bit further. It accepts a **buffer overflow exploit** as input and randomly creates a functionally equivalent version (polymorphism, part deux).

ADMutate substitutes the conventional NOPs with operationally inert commands. **ADMutate** encodes the shellcode with a **simple mechanism (xor)** so that the shellcode will be unique to any NIDS sensor. This allows it to bypass shellcode signature analysis. XORing encodes the shellcode with a randomly generated key. It **modulates** the return address and the least significant byte is altered to jump into different parts of the stack. It also allows the attacker to apply different weights to generate **ASCII equivalents** of machine language code and to tweak the **statistical distribution** of resulting characters. This formulates the traffic as the “standard” for a given protocol, from a statistical perspective, for example, more heavily weighted characters "<" and ">" in **HTTP protocol**. To further reduce the pattern of the decoder, out-of-order decoders are supported. This allows the user to specify where in the decoder certain operational instructions are placed. ADMutate was developed to offend **IDS signature checking** by manipulation of buffer overflow exploits. It uses **techniques borrowed** from **virus creators** and works on Intel, Sparc, and HPPA processors. The likely **targets** are Linux, Solaris, IRIX, HPUX, OpenBSD, UnixWare, OpenServer, TRU64, NetBSD, and FreeBSD.

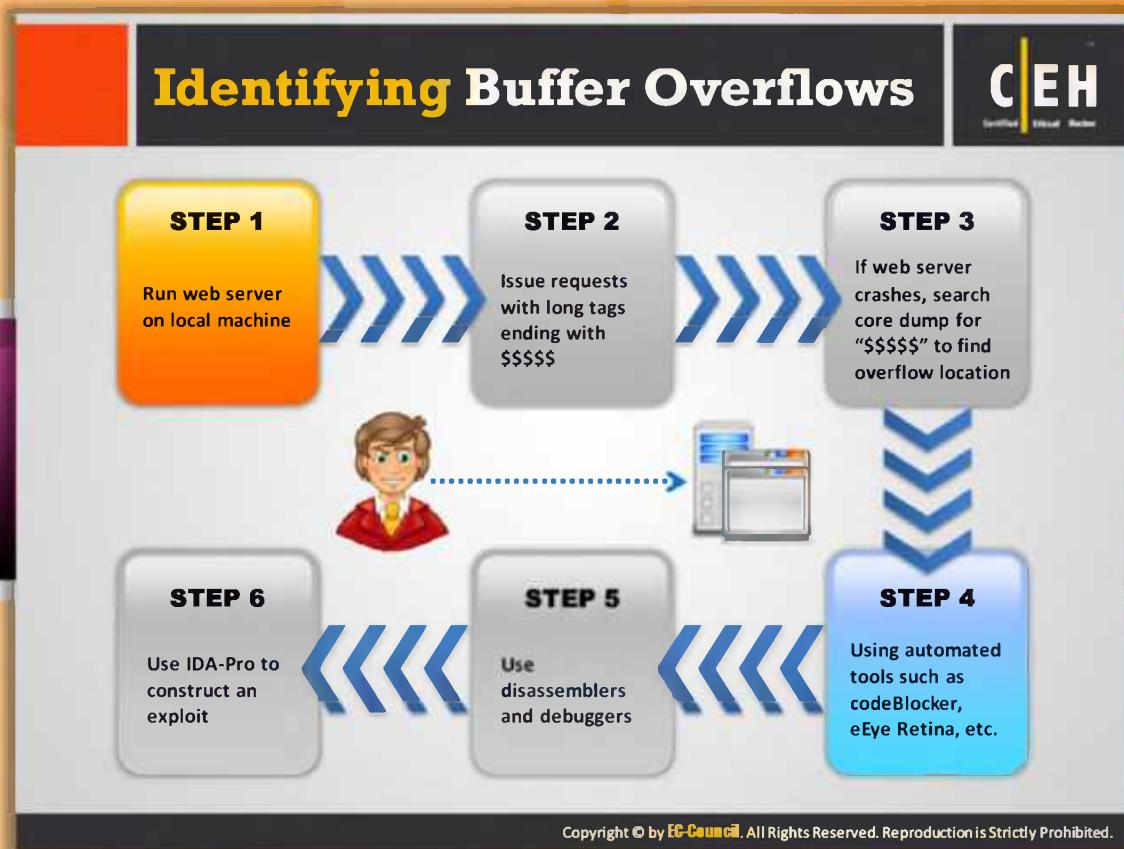


Module Flow

So far, we have discussed what buffer overflow is and how to exploit it. Now it's time to discuss how to detect buffer overflows.

Buffer Overflow Concepts	Buffer Overflow Countermeasures
Buffer Overflow Methodology	Buffer Overflow Security Tools
Buffer Overflow Examples	Buffer Overflow Pen Testing
Buffer Overflow Detection	

This section focuses on various buffer overflow detection methods such as testing for heap and stack overflows, formatting string conditions, and buffer overflow detection tools.



Identifying Buffer Overflows

In order to identify the buffer overflow **vulnerability**, follow the steps mentioned as follows:

- ④ Step 1: Run web server on local machine
- ④ Step 2: Issue requests with long tags-all long tags end with "\$\$\$\$\$"
- ④ Step 3: If the web server crashes, search core dump for "\$\$\$\$\$" to **find overflow** location
- ④ Step 4: Using automated tools such as codeBlocker, eEye Retina, etc.
- ④ Step 5: Use **disassemblers** and debuggers
- ④ Step 6: Use IDA-Pro to **construct an exploit**

How to Detect Buffer Overflows in a Program

Local Variables
In this case, the attacker can look for **strings declared as local variables in functions or methods**, and verify the presence of boundary checks

Standard Functions
It is also necessary to check for **improper use of standard functions**, especially those related to strings and input or output

Another way is to **feed the application** with huge amounts of data and check for abnormal behavior

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



How to Detect Buffer Overflows in a Program

To **identify buffer overflows**, you need to **examine programs systematically** in order to discover vulnerabilities. There are basically two main ways to detect buffer overflow **vulnerabilities**:

- The first method is to look at the source code:

The attacker looks for strings declared as local variables in functions or methods and verifies the presence of a **boundary check** or use of “SAFE” C functions. In addition, it is also necessary to check for the improper use of standard functions, especially those related to strings and input/output.

- The second method is to feed the huge volumes of data to the application and check for abnormal behavior.

To start with, you can attempt to reverse the code using a **disassembler** or **debugger** and examine the code for vulnerabilities.

Disassembly starts from the entry point of the program and then proceeds with all routes of execution to search for the functions that are **external** to the main flow of the program. The user may keep his or her focus on functions lying outside main () and check those **subroutines** that take strings as their input or generate them as output.

As already mentioned, programs written in C are particularly susceptible, because the language does not have any **built-in bounds checking**, and overflows are **undetected** as they write past the end of a character array. The standard C library offers many functions for the purpose of copying or appending strings that do not perform any **boundary check**. These include `strcat()`, `strcpy()`, `sprintf()`, and `vsprintf()`. These functions operate on **null-terminated** strings and do not check for an overflow resulting from a received string.

The `gets()` function reads a line from `stdin` into a buffer until a terminating new line or EOF occurs. It does not check for any buffer overflows. The `scanf()` function also gives rise to potential overflows, if the program attempts to match a sequence of non-white-space characters (`%s`) or a **non-empty sequence** of characters from a specified set (`%[]`).

The array pointed to by the `char` pointer is inadequate to accept the entire sequence of characters, and the optional maximum field width is not specified. If the target of any of these functions is a buffer of static size, and its arguments are derived from user input, there is a good chance of encountering a buffer overflow.

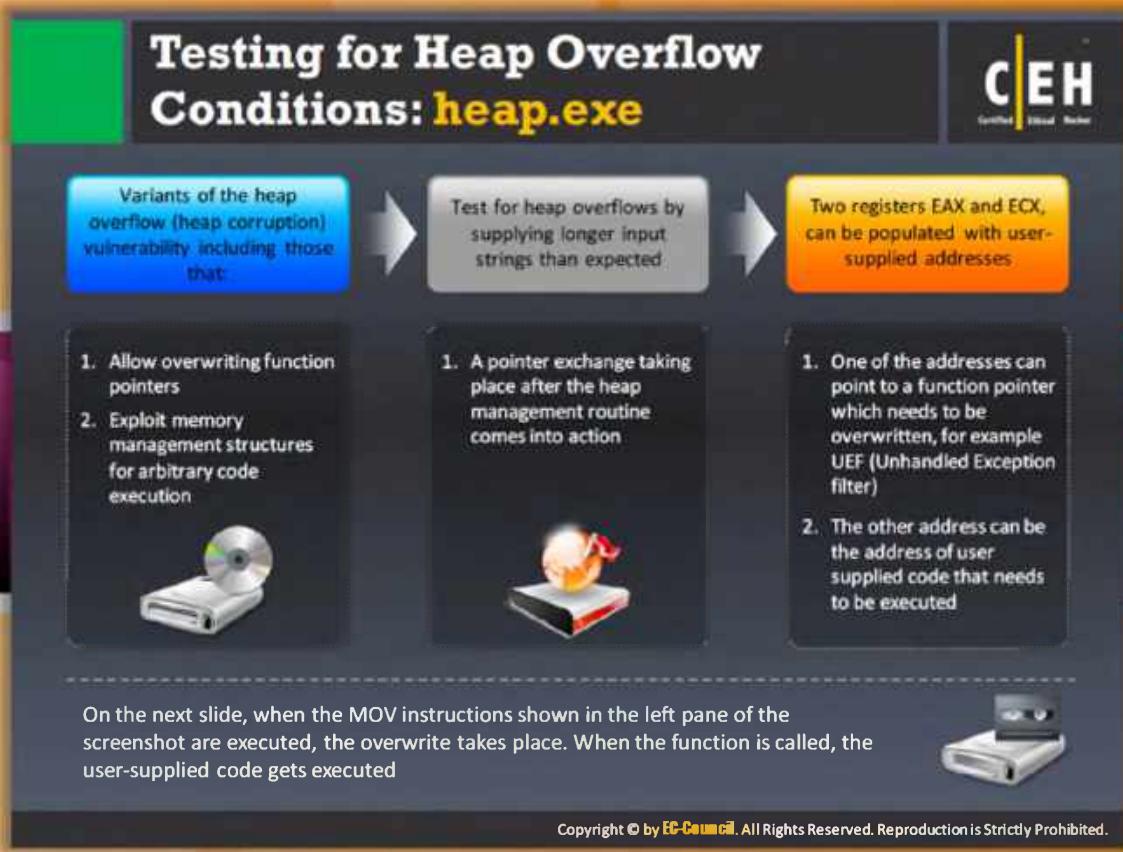
Most attackers point out that **ingenuity** is critical for **exploiting** buffer overflow vulnerabilities. This is true especially when one has to guess a few parameters. For instance, if the attacker is looking at software that assists in communication such as FTP, he or she may be looking at commands that are typically used and how they are implemented.

For example, the attacker can search for text and pick out a suspect variable from a table. He or she can then go on and check the code for any boundary checks and functions such as `strcpy()` that take input directly from the buffer. The **emphasis** can be on local variables and parameters. The attacker can then test the code by providing **malformed** input and observe the resulting behavior of the code.

Another method is to adopt a brute force approach by using an automated tool to **bombard** the program with excessive amounts of data and cause the program to crash in a meaningful way. The attacker can then examine the register dump to check whether the data bombarding the program made its way into the instruction pointer.

What happens after the buffer overflow vulnerability is discovered? After discovering a vulnerability, the attacker can observe carefully how the call obtains its user input and how it is routed through the function call. He or she can then write an exploit, which makes the software do things it would not do normally. This can range from simply crashing the machine to injecting code so that the attacker can gain remote access to the machine. He or she might then use the **compromised system** as a launch base for further attacks.

However, the greatest threat comes from a **malicious program** such as a worm that is written to take advantage of the **buffer overflow**.



Testing for Heap Overflow Conditions: heap.exe

A heap is memory that is **allocated dynamically**; these are dynamically removed (example delete, free) and created (example new, malloc). In some cases, heaps are reallocated by the programmer. Each memory chunk in a heap is associated with **boundary tags** containing information about **memory management**.

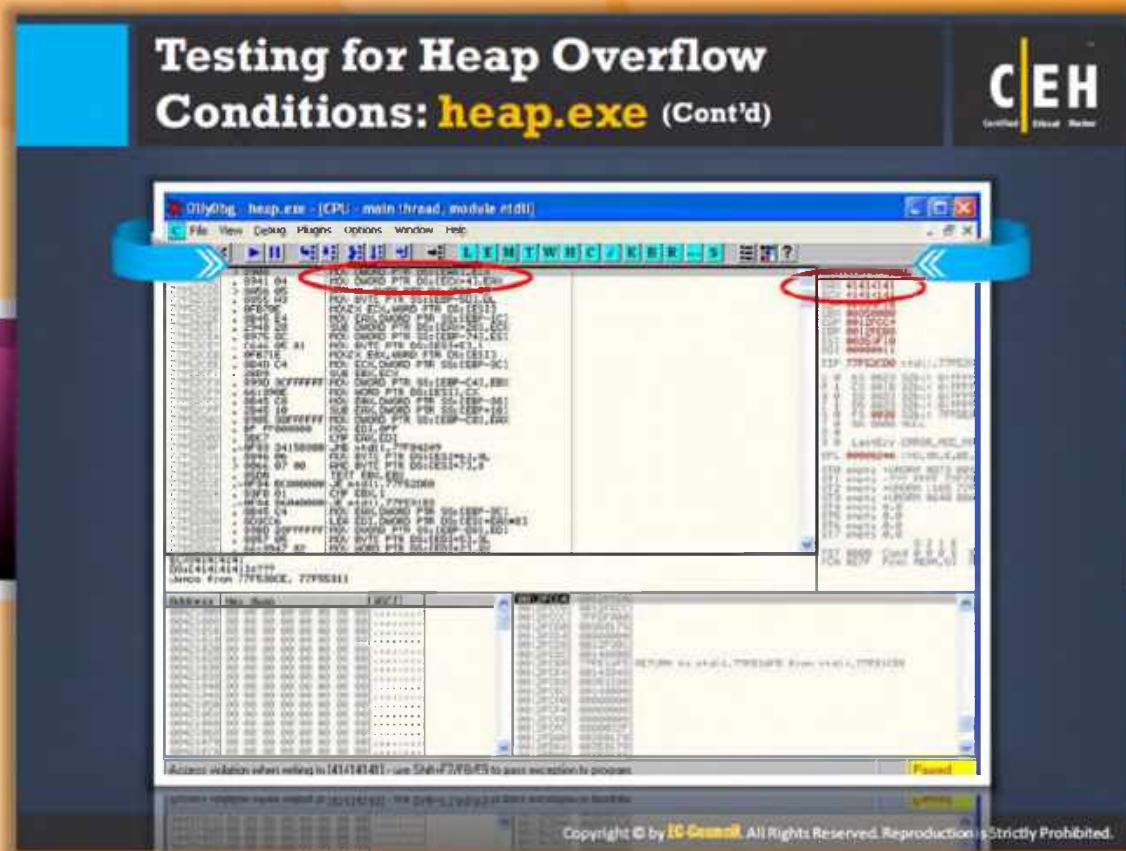
Heap-based buffer overflow causes overwriting the **control information**. This leads to an access violation when the heap management routine frees the buffer. This overflow vulnerability allows an attacker to overwrite a desired memory location with a user-controlled value, when executed in a controlled fashion. Thus, it allows an attacker to **overwrite function** pointers and other addresses stored in TEB, GOP, or .dtors structures with the shellcode's address.

There are many ways in which the heap overflow vulnerability can be **exploited** to **execute shellcode** by overwriting function pointers. In order to exploit these vulnerabilities, certain conditions need to exist in the code. Hence, identifying or locating these **vulnerabilities** requires closer examination when compared to **stack overflow vulnerabilities**.

You can test for **heap overflows** by supplying input strings longer than expected. **Heap overflow** in a Windows program may appear in various forms. The most common one is pointer exchange taking place after the **heap management routine** frees the buffer.

Two registers EAX and ECX, can be populated with user-supplied addresses:

- ➊ One of the addresses can point to a function pointer that needs to be overwritten, for example, UEF (Unhandled Exception filter)
- ➋ The other address can be the address of user-supplied code that needs to be executed



Testing for Heap Overflow Conditions: heap.exe (Cont'd)

Let us consider the following example of heap overflow vulnerability:

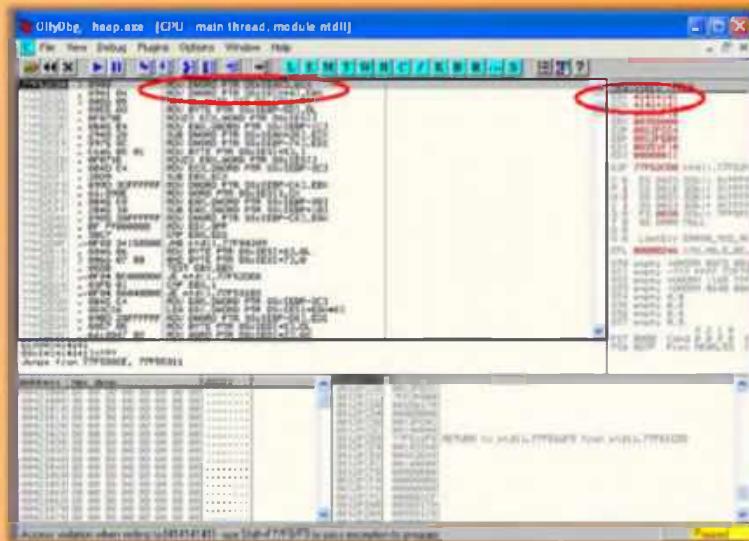


FIGURE 18.9: Testing for Heap Overflow Conditions in heap.exe

The two registers shown, **EAX** and **ECX**, can be populated with user-supplied addresses used to overflow the heap buffer. One address points to a function pointer such as **Unhandled Exception Filter (UEF)** that needs to be overwritten, and the other holds the address of the arbitrary code. The overwrite takes place when the **MOV instructions** are executed. When the function is called, the **arbitrary code** gets executed.

In addition to this method, the **heap-based buffer overflows** can be identified by **reverse engineering** the application binaries and using **fuzzing techniques**.

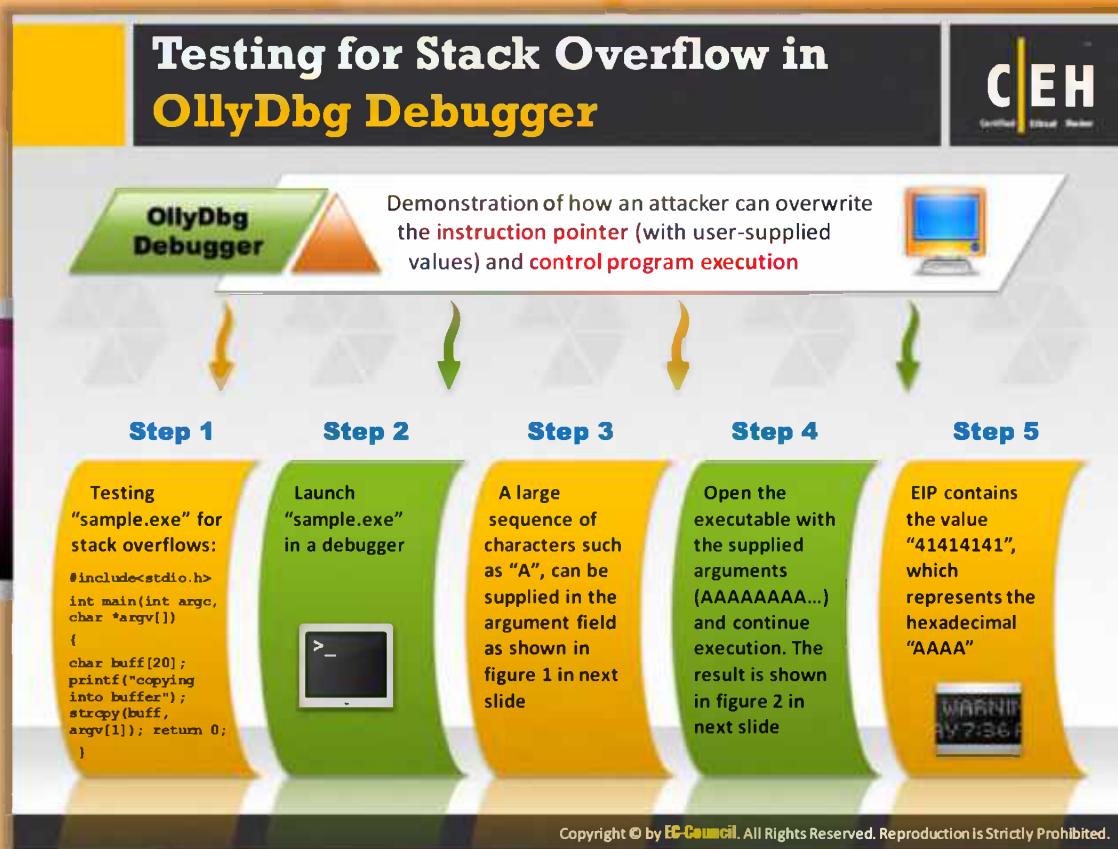


Steps for Testing for Stack Overflow in OllyDbg Debugger

Stack buffer overflow occurs when the **variable data** size larger than the buffer size is placed in the program stack without bound checking. This can be a serious vulnerability and may even cause denial-of-service attacks. A **stack overflow vulnerability** allows an attacker to take control of the **instruction pointer**. This may lead to severe consequences. Therefore, you need to test your application or processes against stack overflow vulnerabilities.

Similar to **heap-based buffer overflow** testing, the stack overflow vulnerability can also be tested by supplying a large amount of input data than the normal or expected. However, this alone is not enough. In addition to sending a large amount of input data, you need to inspect the execution flow of the application and the responses to check whether an overflow has occurred or not. You can do this in four steps with the help of a **debugger**, a computer program used to test other programs. Here we are testing for stack overflow with the help of the **OllyDbg debugger**. The first step in testing for stack overflow is to attach a debugger to the target application or process. Once you successfully attach the program, you need to generate the **malformed** large input data for the target application to be tested. Now, supply the malformed input to the application and **inspect** the responses in a **debugger**. The ollydbg debugger allows you to observe the execution flow and the state of registers when the stack

overflow gets **triggered**. On the next slide, we will discuss these steps by considering an example.



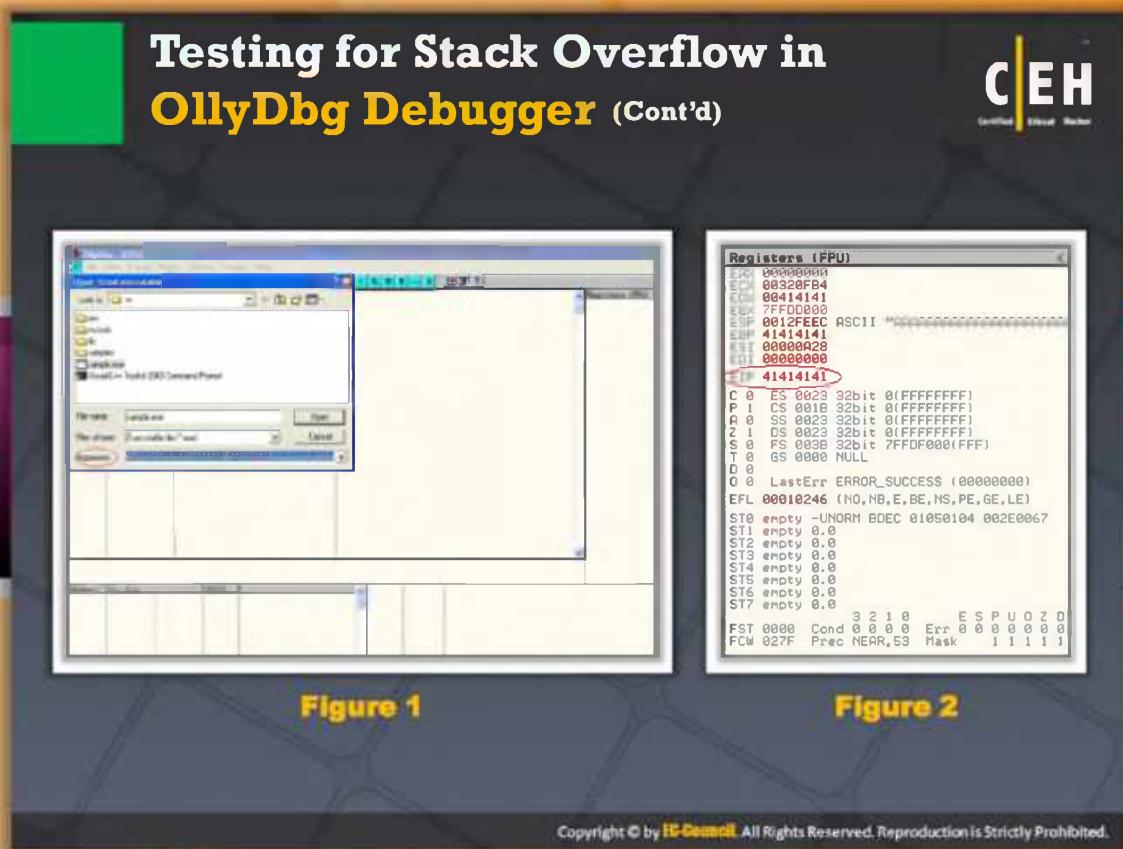
Testing for Stack Overflow in OllyDbg Debugger

Here we are going to demonstrate how an attacker can **overwrite** the instruction pointer (with user-supplied values) and **control** program execution. Consider the following example of "sample.exe" to test for stack overflows:

```
#include<stdio.h>
int main(int argc, char *argv[])
{
    char buff[20];
    printf("copying into buffer");
    strcpy(buff, argv[1]);
    return 0;
}
```

Launch the sample.exe in OllyDbg debugger. The sample.exe accepts **command line arguments**. So you can supply a large **sequence of characters** such as 'A' in the **argument field** as shown in figure 1 on the next slide. Now open sample.exe with the supplied arguments (AAAAAAA...) and continue execution. The result is shown in figure 2 on the next slide. From the figure 2 in

next slide, it is clear that the EIP (Extended Instruction Pointer) contains the value "41414141." The hexadecimal representation of character 'A' is 41. Hence "41414141" represents the string "AAAA."



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Testing for Stack Overflow in OllyDbg Debugger (Cont'd)

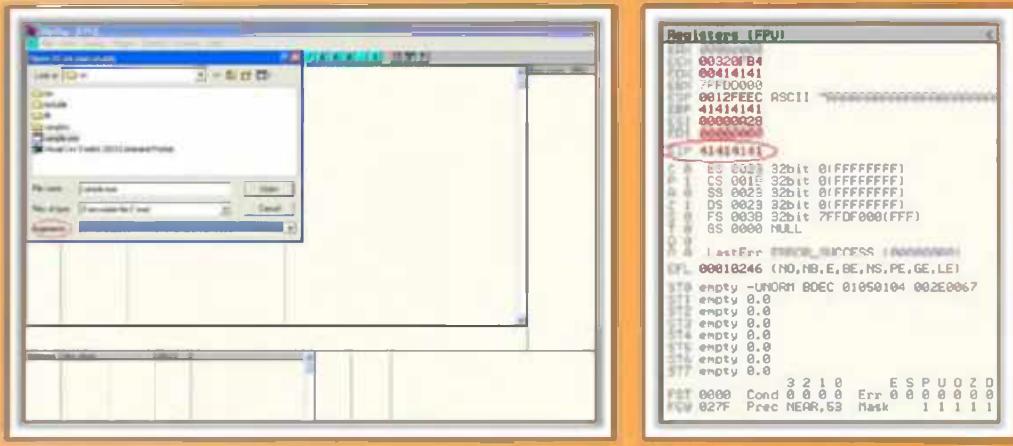


FIGURE 18.10: Testing for Stack Overflow in OllyDbg Debugger

Testing for Format String Conditions Using IDA Pro



Format String Vulnerabilities

Format string vulnerabilities are most often exploited within:

- Web servers
- Application servers
- Web applications utilizing C/C++ based code
- CGI scripts written in C



Manipulating Input Parameters

Attacker manipulates input parameters to include %x or %n type specifiers

For example, a legitimate request like:
`http://hostname/cgi-bin/
query.cgi?name=john&code=45765`

is changed into:
`http://hostname/cgi-bin/
query.cgi?name=john%x.%x.%x&code=45765%x.%x`



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Testing for Format String Conditions Using IDA Pro

Applications can be tested for format **string vulnerabilities** by supplying format type specifiers in application input. Format string vulnerabilities usually appear in **web servers**, **application servers**, or web applications utilizing **C/C++** based code or **CGI scripts** written in C. Most of these format string vulnerabilities are resulting because of the **insecure** call to error **reporting** or **logging** function like `syslog()`.

An attacker manipulates input parameters to include %x or %n type specifiers in a CGI script. Consider a legitimate example:

`http://hostname/cgi-bin/query.cgi?name= john&code=45765`

Attacker can manipulate this to

`http://hostname/cgi-bin/query.cgi?name= john%x.%x.%x&code=45765%x.%x`

If the routine processing the altered request contains any format string vulnerability, then it prints out the stack data to browser.

Testing for Format String Conditions Using IDA Pro (Cont'd)

The slide features a large central image showing a screenshot of the IDA Pro debugger. The assembly view shows several printf instructions. One instruction, at address 0x401010, has its assembly line highlighted: `printf("The string entered is\n");`. A tooltip above this line indicates the address of the format type specifier: `0x401010 offset of %s in main()`. Below the assembly view is a command prompt window containing the following C code:

```
int main(int argc, char **argv)
{
    printf("The string entered is\n");
    printf("%s", argv[1]);
    return 0;
}
```

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Testing for Format String Conditions Using IDA Pro (Cont'd)

An attacker identifies the presence of a format **string vulnerability** by checking instances of code (assembly fragments). Consider the following example code:

```
int main(int argc, char **argv)
{
    printf("The string entered is\n");
    printf("%s", argv[1]);
    return 0;
}
```

Examine the **disassembly** of the code using IDA Pro. You can clearly see the address of format type specifier being pushed on the stack before a call to printf is made.

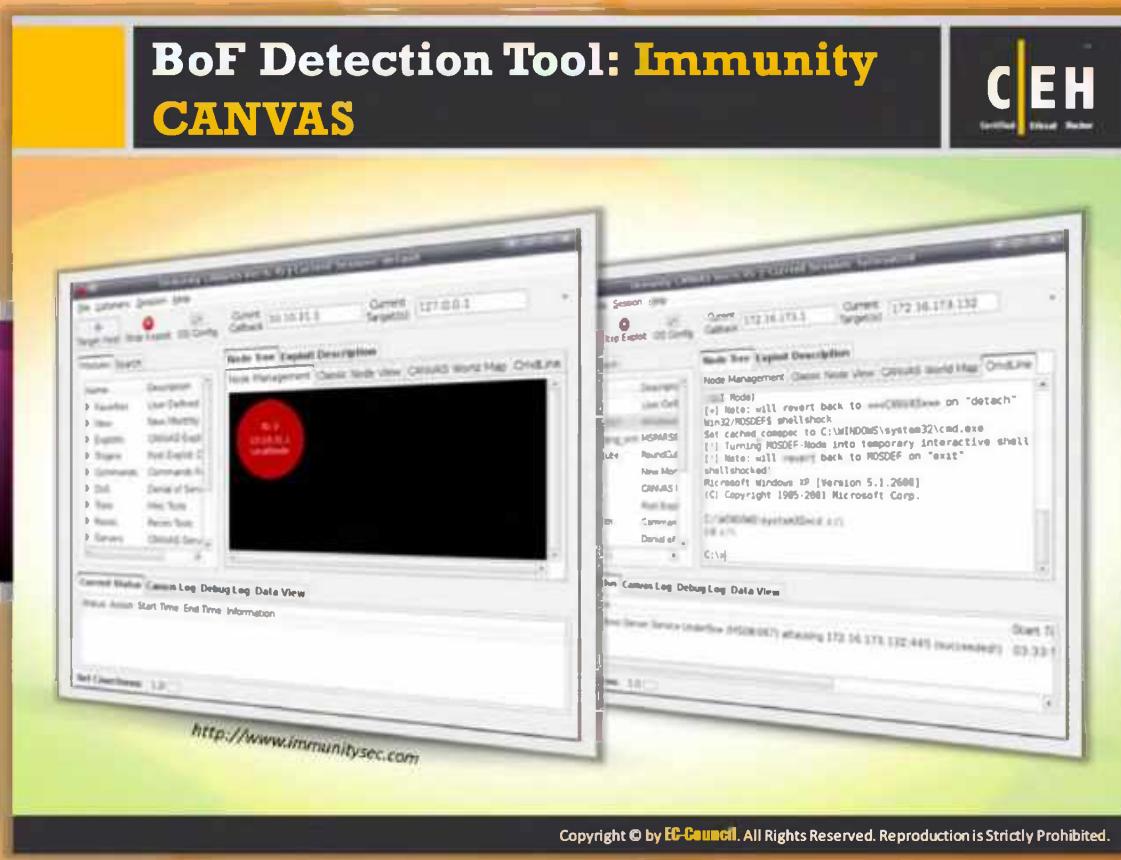
No offset will be pushed on the stack before calling printf, when the same code is compiled without "%s" as an argument.

The screenshot shows the assembly view of IDA Pro for a buffer overflow exploit. The assembly code is as follows:

```
text:00401010 arg_4 = dword ptr 0Ch
text:00401010 push    ebp
text:00401011 mov     esp, ebp
text:00401013 sub     esp, 40h
text:00401016 push    ebx
text:00401017 push    esi
text:00401018 push    edi
text:00401019 lea     edi, [ebp+var_40]
text:0040101C mov     ecx, 10h
text:00401021 mov     eax, 0CCCCCCCCh
rep stosd
text:00401028 push    offset ??_C@_0B@H@H@K@H@The?5String?5entered?5i
text:0040102D call    printf
text:00401032 add    esp, 4
text:00401035 mov     eax, [ebp+arg_4]
text:00401038 mov     ecx, [eax+4]
push    ecx
text:0040103C push    offset ??_C@_02D@L@?$CFS?$AA@ db 25h ; %
text:00401041 call    printf
??_C@_0B@H@H@K@H@The?5String?5entered?5i@?6?$AA@ db 'The string entered is',0Ah,0
; DATA XREF: main+2CTo
; DATA XREF: heap_alloc_dbg+8CTo ...
db    73h ; s
db    0
db    0
db    0
```

A red oval highlights the instruction `push offset ??_C@_0B@H@H@K@H@The?5String?5entered?5i@?6?$AA@`. Another red oval highlights the string `'The string entered is',0Ah,0`.

FIGURE 18.11: Testing for Format String Conditions Using IDA Pro



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



BoF Detection Tool: Immunity CANVAS

Source: <http://www.immunitysec.com>

Immunity's CANVAS is an **exploit development framework** for **penetration testers** and **security professionals**. It allows you to discover how vulnerable you really are. It comes with packaged vulnerability exploitation modules for scripting and framework for developing **original exploits**. Thus, it provides a way for any organization to have a picture of their security posture, without guesswork or estimation.

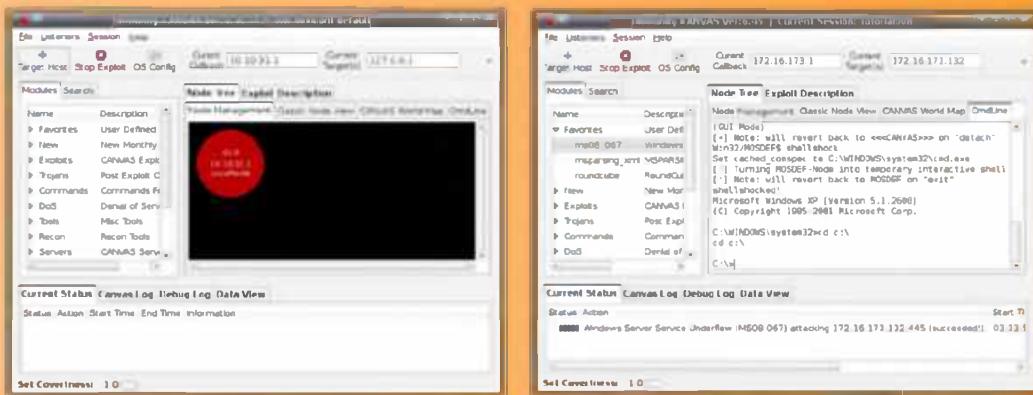
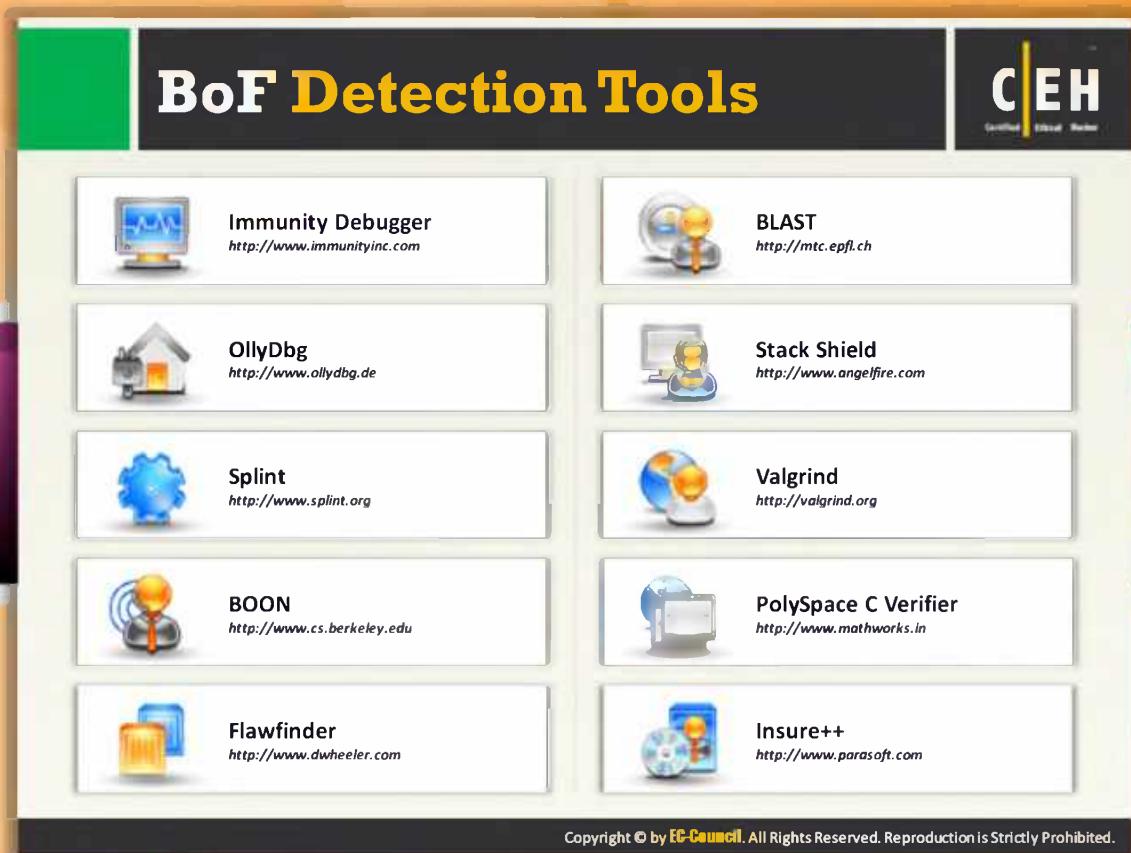


FIGURE 18.12: Immunity CANVAS Screenshot



The slide features a dark header with a green vertical bar on the left and the text "BoF Detection Tools" in yellow. On the right is the CEH logo. Below the header is a grid of eight tool entries, each with an icon, name, and URL. The tools listed are: Immunity Debugger, BLAST, OllyDbg, Stack Shield, Splint, Valgrind, BOON, PolySpace C Verifier, Flawfinder, and Insure++. A copyright notice at the bottom states "Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited."

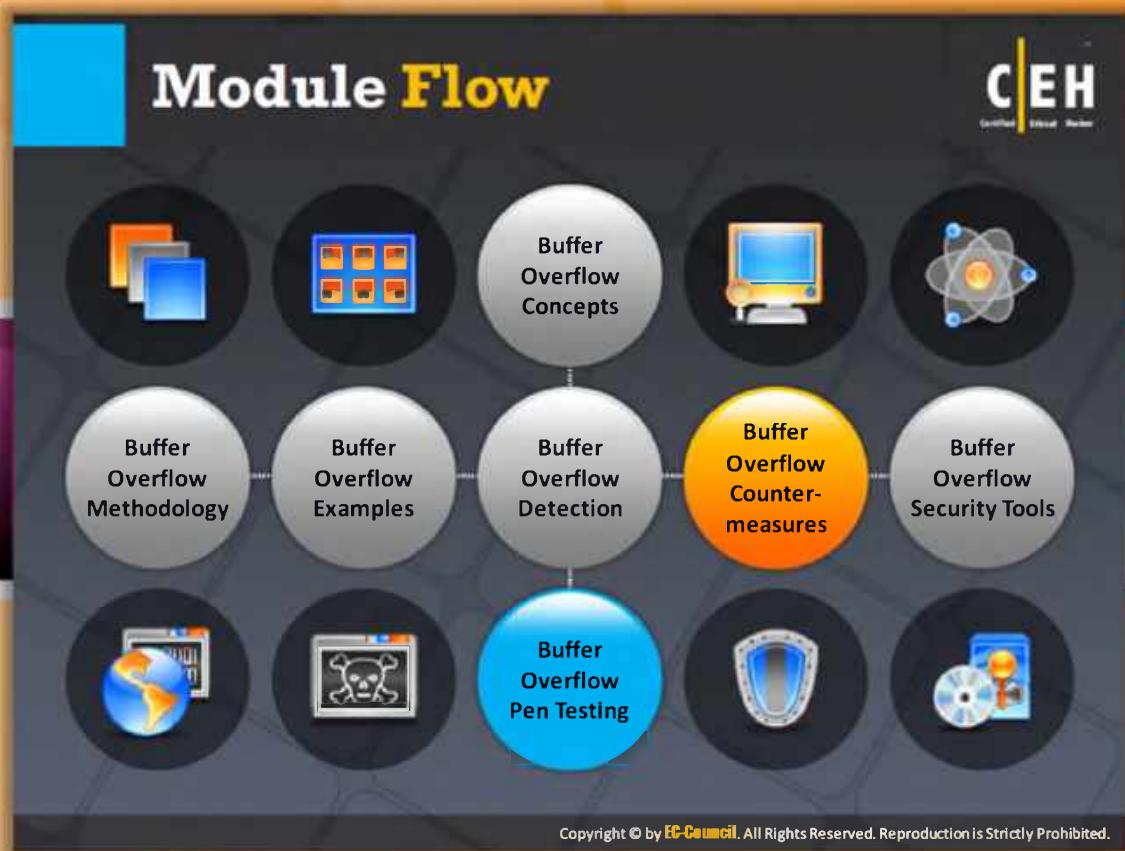
Tool	URL
Immunity Debugger	http://www.immunityinc.com
BLAST	http://mtc.epfl.ch
OllyDbg	http://www.ollydbg.de
Stack Shield	http://www.angelfire.com
Splint	http://www.splint.org
Valgrind	http://valgrind.org
BOON	http://www.cs.berkeley.edu
PolySpace C Verifier	http://www.mathworks.in
Flawfinder	http://www.dwheeler.com
Insure++	http://www.parasoft.com



BoF Detection Tools

In addition to OllyDbg Debugger, IDA Pro, and Immunity CANVAS, many other tools have the capability to detect buffer overflows. A few **buffer overflow detection** tools are listed as follows:

- ④ Immunity Debugger available at <http://www.immunityinc.com>
- ④ OllyDbg available at <http://www.ollydbg.de>
- ④ Splint available at <http://www.splint.org>
- ④ BOON available at <http://www.cs.berkeley.edu>
- ④ Flawfinder available at <http://www.dwheeler.com>
- ④ BLAST available at <http://mtc.epfl.ch>
- ④ Stack Shield available at <http://www.angelfire.com>
- ④ Valgrind available at <http://valgrind.org>
- ④ PolySpace C Verifier available at <http://www.mathworks.in>
- ④ Insure++ available at <http://www.parasoft.com>



Module Flow

So far, we have discussed the buffer overflow vulnerability, how to exploit it, and how to detect it. Once you detect buffer overflows, you should immediately apply or take countermeasures to protect your resources from being **compromised**. There are many reasons for buffer overflow exploits. The countermeasures to be applied may vary depending on the kind of buffer overflow vulnerability.

	Buffer Overflow Concepts		Buffer Overflow Countermeasures
	Buffer Overflow Methodology		Buffer Overflow Security Tools
	Buffer Overflow Examples		Buffer Overflow Pen Testing
	Buffer Overflow Detection		

This section suggests various **countermeasures** to different kinds of buffer overflow vulnerabilities. Thus, it can help you to **prevent buffer overflow attacks**.



Defense against Buffer Overflows

The errors in programs are the main cause of **buffer flow problems**. These problems are responsible for **security vulnerabilities** using which the attacker tries to gain unauthorized access to a remote host. Attackers easily insert and execute attack code. To avoid such problems, some protection measures have to be taken. **Protection measures** to defend against buffer overflows include:



Manual auditing of code

Search for the use of **unsafe functions** in the C library like **strcpy()** and replace them with safe functions like **strncpy()**, which takes the size of the buffer into account. Manual auditing of the source code must be undertaken for each program.



Compiler techniques

Range checking of indices is defined as a defense that guarantees **100% efficiency** from buffer overflow attacks. Java automatically checks if an array index is within the proper bounds. Use compilers like Java, instead of C, to avoid buffer overflow attacks.



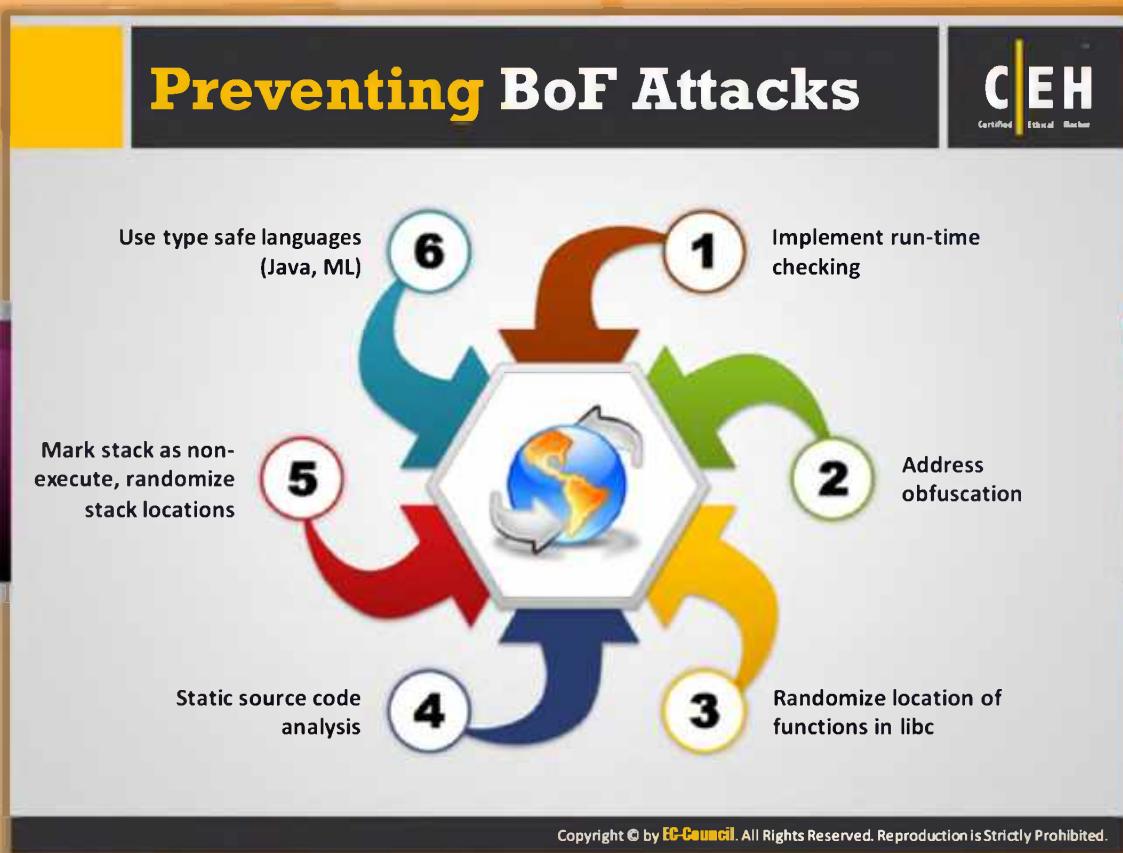
Safer C library support

A robust alternative is to provide safe versions of the **C library functions** where it attacks by overwriting the return address. It works with the binaries of the **target program's** source code and does not require access to the program's source code. It can be handled according to the occurrence of the threat without any **vendors operating** against it. It is available for Windows 2000 systems and is an **effective technique**.



Disabling stack execution

This is an easy solution that provides an option to install the **OS-disabling stack** execution. The idea is simple, inexpensive, and relatively effective against the current crop of attacks. A weakness in this method is that some programs depend on the execution of the stack.



Preventing BoF Attacks

A buffer overflow attack occurs when large amounts of data are sent to the system, more than it is intended to hold. This attack usually occurs due to **insecure programming**. Often this may lead to a **system crash**. To avoid such problems, some **preventive measures** are adopted. They are:

- ⊕ Implement run-time checking
- ⊕ **Address obfuscation**
- ⊕ **Randomize** location of functions in libc
- ⊕ Static source code analysis
- ⊕ Mark stack as non-execute, random stack location
- ⊕ Use type safe languages (Java, ML)

Programming Countermeasures



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

	Design programs with security in mind		Consider using "safer" compilers such as StackGuard
	Disable Stack Execution (it's possible with hardware segmentation, or software segmentation such as DEP)		Prevent return addresses from being overwritten
	Test and debug the code to find errors		Validate arguments and reduce the amount of code that runs with root privilege
	Prevent use of dangerous functions: gets, strcpy, etc.		Prevent all sensitive information from being overwritten

Programming Countermeasures

- ④ Design programs with security in mind.
- ④ Disable stack execution (possible on Solaris).
- ④ Test and debug the code to find errors.
- ④ Prevent use of dangerous functions: gets, strcpy, etc.
- ④ Consider using "safer" compilers such as StackGuard.
- ④ Prevent return addresses from being overwritten.
- ④ **Validate arguments** and reduce the amount of code that runs with **root privilege**.
- ④ Prevent all sensitive information from being overwritten.

Programming Countermeasures (Cont'd)

I Make changes to the **C language** itself at the language level to reduce the risk of buffer overflows.

II Use **static or dynamic source code analyzers** at the source code level to check the code for buffer overflow problems

III Change the **compiler** at the compiler level that does bounds checking or protects addresses from overwriting

IV Change the rules at the **operating system level** for which memory pages are allowed to hold executable data

V Make use of **safe libraries**

VI Make use of tools that can **detect buffer overflow vulnerabilities**



Programming Countermeasures (Cont'd)

- ➊ Make changes to the C language itself at the language level to **reduce the risk** of buffer overflows.
- ➋ Use static or **dynamic source code analyzers** at the source code level to check the code for buffer overflow problems.
- ➌ Change the compiler at the **compiler level** that does bounds checking or protects addresses from overwriting.
- ➍ Change the rules at the **operating system** level for which memory pages are allowed to hold executable data.
- ➎ Make use of safe libraries.
- ➏ Make use of tools that can **detect** buffer overflow vulnerabilities.

Data Execution Prevention (DEP)

The slide features a list of five points about DEP, followed by a screenshot of the Windows 'Performance Options' dialog box under the 'Data Execution Prevention' tab.

- DEP is a set of **hardware and software** technologies that **monitors** programs to verify whether they are using system memory **safely and securely**
- It prevents applications from accessing memory that wasn't assigned for the **process** and lies in another region
- When a violation is attempted, **hardware-enforced DEP** detects code that is running from these locations and raises an exception
- To prevent Malicious code from taking advantage of exception-handling mechanisms, Windows uses **Software-enforced DEP**
- DEP helps in preventing code execution from within data pages, such as the **default heap pages**, **memory pool pages**, and various **stack pages**, where code is not executed from the default heap and the stack

Performance Options

Data Execution Prevention (DEP) helps protect against damage from viruses and other security threats. How does it work?

Turn on DEP for essential Windows programs and services only

Turn on DEP for all programs and services except those I select:

Add... Remove

Your computer's processor supports hardware-based DEP.

OK Cancel Apply

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Data Execution Prevention (DEP)

Data execution prevention (DEP) is a set of **hardware** and **software technologies** that monitors programs to verify whether they are using system memory safely and securely. It prevents the applications that may access memory that wasn't assigned for the process and lies in another region. When an execution occurs, **hardware-enforced DEP** detects code that is running from these locations and raises an exception. To prevent malicious code from taking advantage of **exception-handling** mechanisms in Windows, use software-enforced DEP.

DEP helps in **preventing code execution** from data pages, such as the **default heap pages**, **memory pool pages**, and various **stack pages**, where code is not executed from the default heap and the stack.

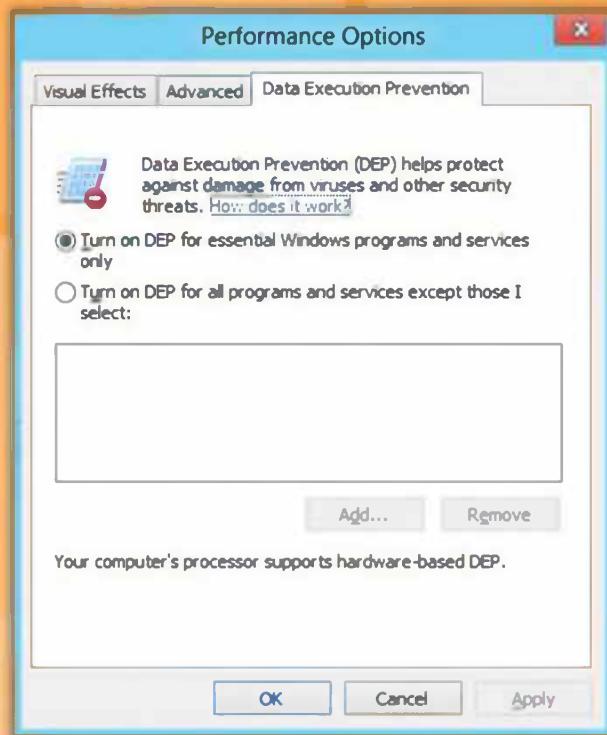


FIGURE 18.13: Data Execution Prevention (DEP)

Enhanced Mitigation Experience Toolkit (EMET)

C|EH
Certified Ethical Hacker

- Enhanced Mitigation Experience Toolkit (EMET) is designed to make it **more difficult** for an attacker to **exploit the vulnerabilities** of software and gain access to the system
- It supports **mitigation techniques** that prevent common attack techniques, primarily related to stack overflows and the techniques used by malware to interact with the operating system as it attempts a compromise
- It improves the **resiliency** of Windows to the exploitation of buffer overflows

Structure Exception Handler Overwrite Protection (SEHOP)
It prevents common techniques used for exploiting stack overflows in Windows by performing **SEH chain validation**

Dynamic Data Execution Prevention (DDEP)
It marks portions of process memory **non-executable**, making it difficult to exploit **memory corruption vulnerabilities**

Address Space Layout Randomization (ASLR)
New in EMET 3.0 is mandatory **address space layout randomization (ASLR)**, as well as non-ASLR-aware modules on all new Windows Versions

<http://support.microsoft.com>
Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Enhanced Mitigation Experience Toolkit (EMET)

Enhanced Mitigation Experience Toolkit (EMET) is designed to make it more difficult for an attacker to **exploit the vulnerabilities** of software and gain access to the system. It supports mitigation techniques that prevent **common attack techniques**, primarily related to **stack overflows** and the techniques used by malware to interact with the operating system as it attempts the compromise. It improves the **resiliency of Windows** to the exploitation of buffer overflows.

Structure Exception HandlerOverwrite Protection (SEHOP):

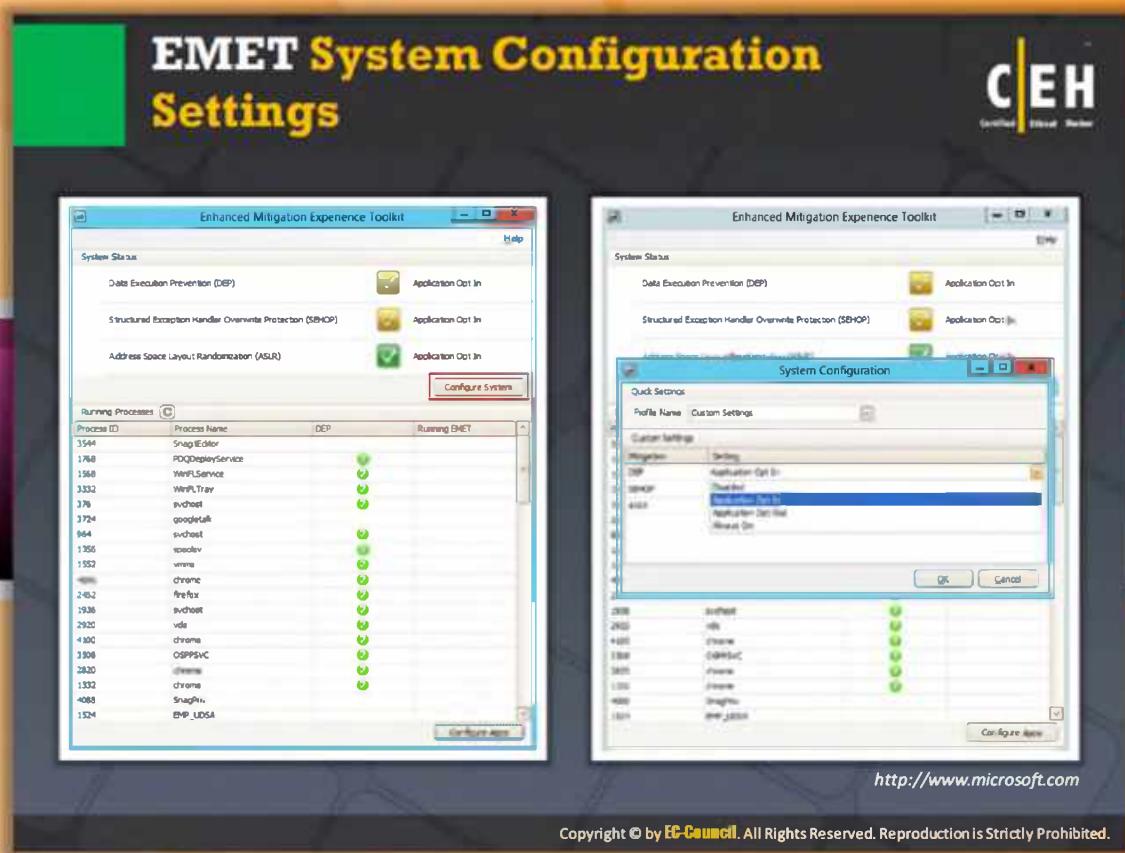
It prevents common techniques used for exploiting stack overflows in Windows by performing **SEH chain validation**.

Dynamic Data Execution Prevention (DDEP):

It marks portions of a process's memory non-executable, making it difficult to exploit **memory corruption vulnerabilities**.

Address Space Layout Randomization (ASLR):

New in EMET 2.0 is mandatory address space layout randomization (ASLR), as well as non-ASLR-aware modules on all new Windows Versions.



EMET System Configuration Settings

Source: <http://www.microsoft.com>

The Enhanced Mitigation Experience Toolkit (EMET) was designed specifically for **preventing vulnerabilities** in software from being **exploited**. After installation, you need to configure EMET to provide protection for software. System and application are the two main **categories** you need to configure on EMET. To configure both these categories, you need to click the respective button present on the right side of the EMET main window. The system status display may vary from one operating system to the other.

The **System Configuration** section is used to configure system-wide (i.e., no need to explicitly define the process to be protected) **specific mitigations** such as DEP, SEHOP, and ASLR. In operating systems such as Windows 7, when the system configuration is set to maximum security, the DEP option will be set to Always On, SEHOP to Application Opt Out, and ASLR to Application Opt In modes. But, setting DEP to Always On may cause all applications that are not compatible with DEP to crash. This in turn may cause system instability. Hence, if you wish to accomplish stability, then it is recommended to set all these settings to Application Opt In.

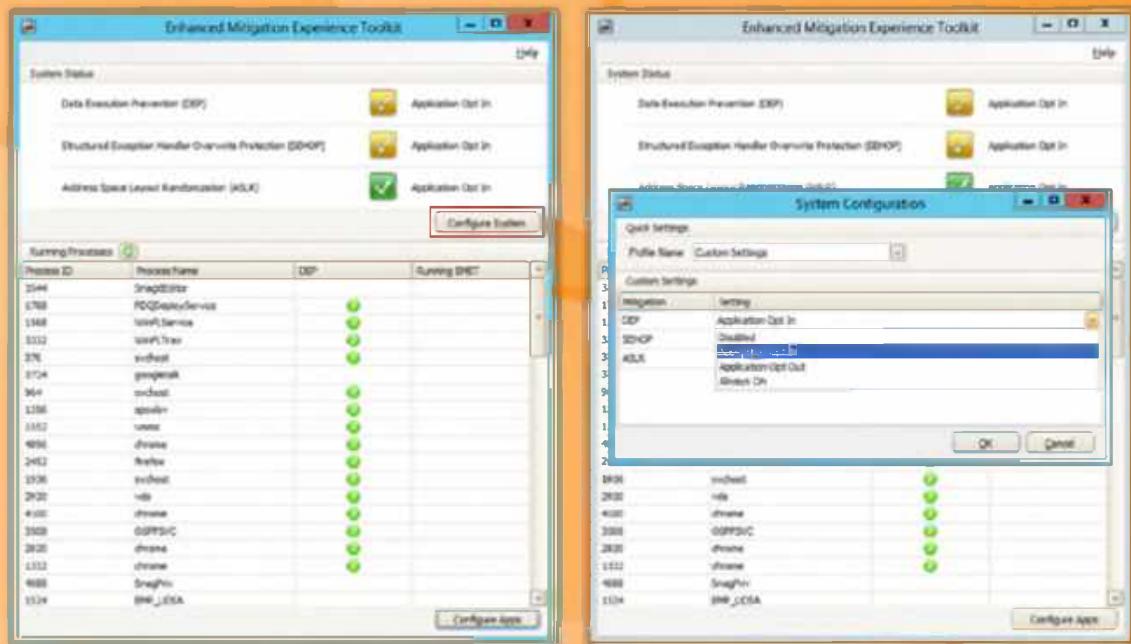
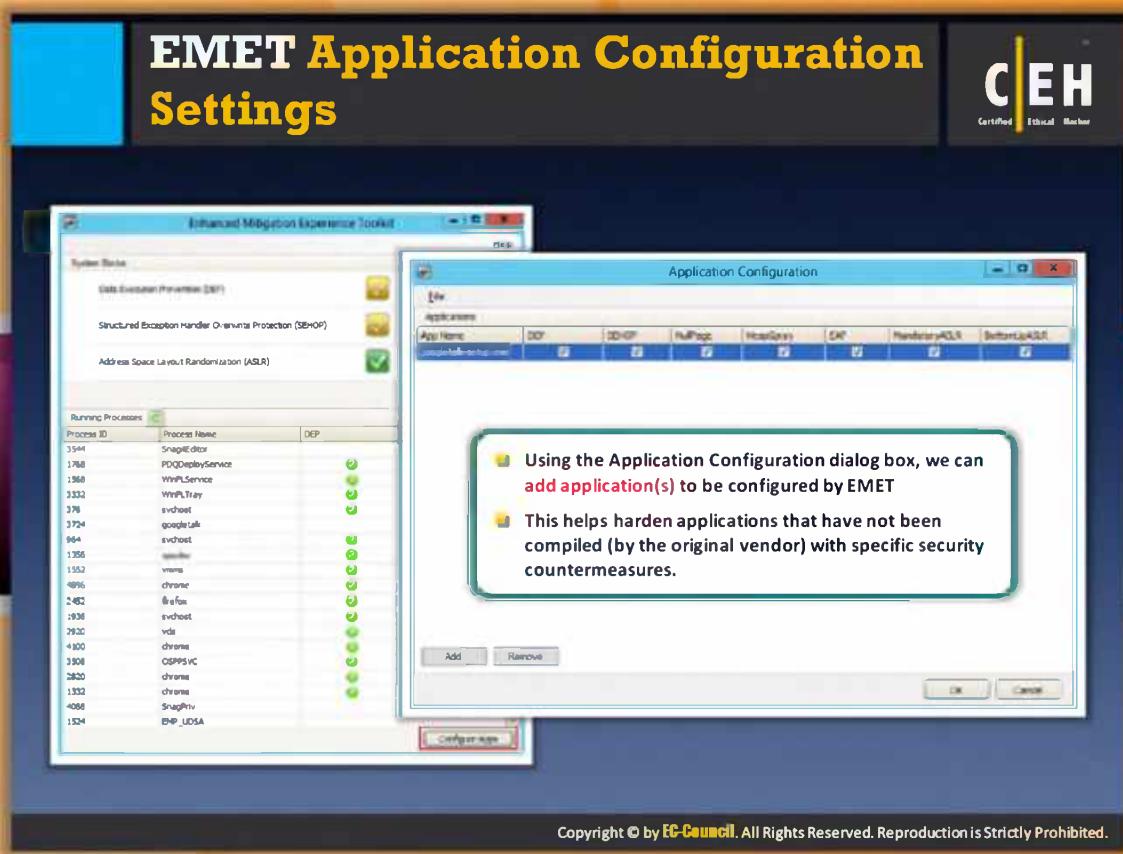


FIGURE 18.14: EMET System Configuration Settings



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



EMET Application Configuration Settings

Contrary to system configuration, application configuration enables **mitigations** such as DEP per application rather than **system-wide**. In order to configure applications, you need to click the **Configure Apps** button in EMET's main window. This will prompt you with the Application Configuration window of EMET. By default, this window will be blank. If you want to protect any particular program, then click the Add button and specify the path where the executables of the programs are installed.

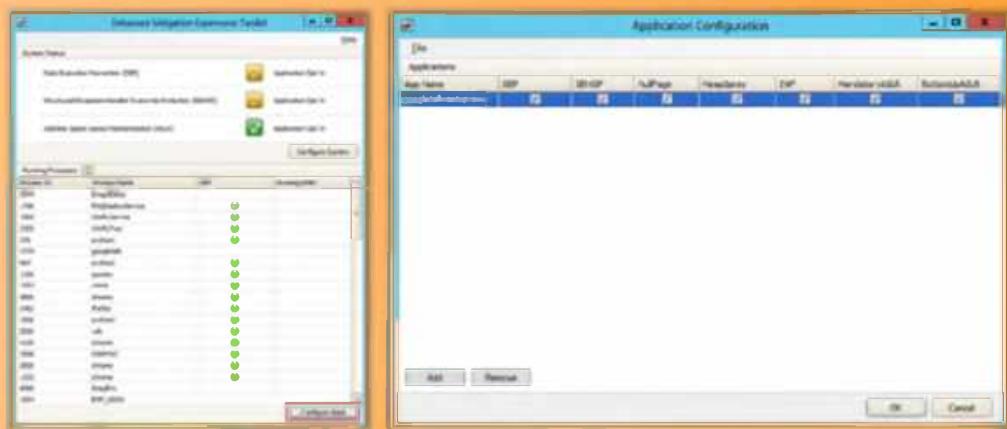
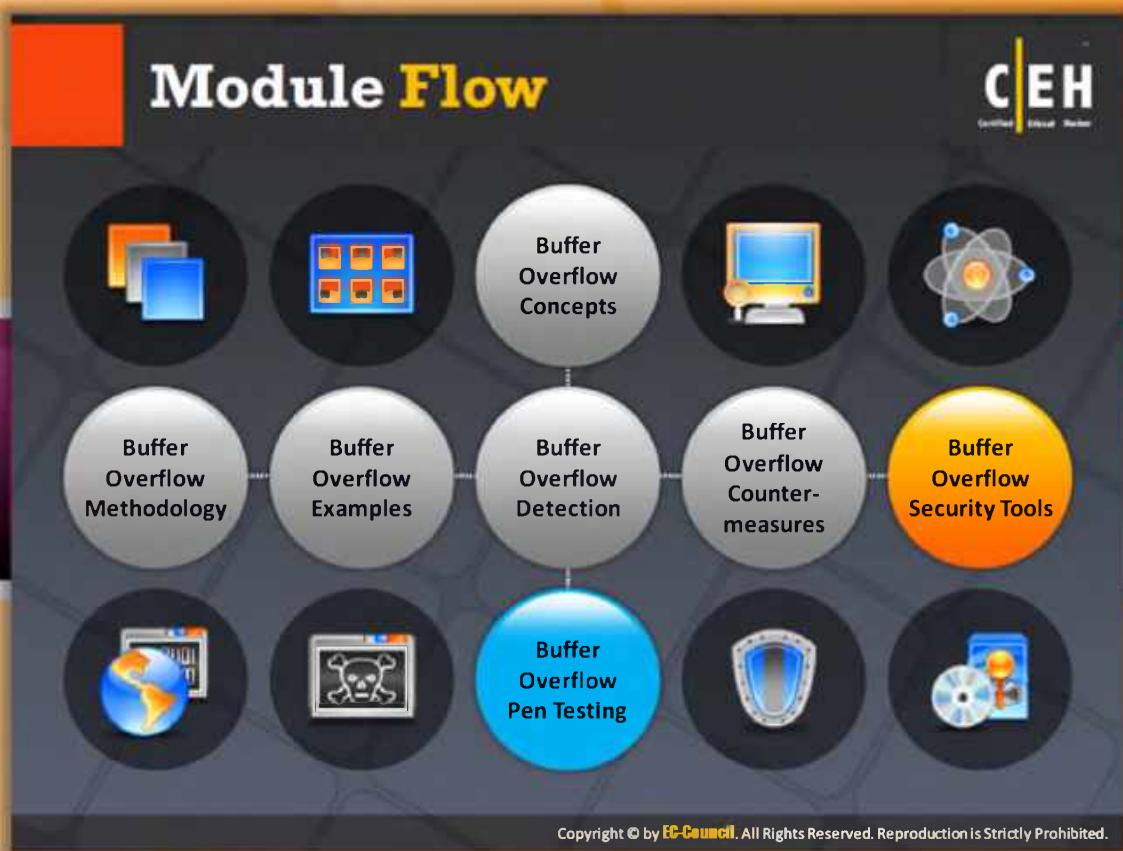


FIGURE 18.15: EMET Application Configuration Settings



Module Flow

So far, we have discussed what buffer overflow is, how to exploit it, buffer overflow examples, detection methods, and countermeasures. In addition to countermeasures, there are some automated buffer overflow security tools that detect and prevent the exploitation of buffer overflows.

	Buffer Overflow Concepts		Buffer Overflow Countermeasures
	Buffer Overflow Methodology		Buffer Overflow Security Tools
	Buffer Overflow Examples		Buffer Overflow Pen Testing
	Buffer Overflow Detection		

This section lists and describes buffer overflow security tools.

The diagram shows a circular stack layout centered around a Windows logo. The segments are labeled clockwise from top: Function Return Address, Cookie, Locally Declared Variables and Buffers, Exception Handler Frame, Callee Save Registers, and Function Parameters. A small orange download icon is positioned above the Function Parameters segment. The CEH logo is in the top right corner.

/GS
<http://www.microsoft.com>

- The Buffer overrun attack exploits **poor coding practices** that programmers adopt when writing and handling the C and C++ string functions
- /GS compiler switch can be **activated** from the Code Generation option page on the C/C++ tab
- The /GS switch provides a "**speed bump**," or **cookie**, between the buffer and the return address that helps in preventing buffer overrun
- If an overflow writes over the return address, it will have to overwrite the cookie put in between it and the buffer, resulting in a new stack layout:

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Source: <http://www.microsoft.com>

The buffer overrun attack utilizes poor coding practices that programmers adopt when writing and handling the **C** and **C++ string functions**. The /GS compiler switch can be activated from the Code Generation option page on the C/C++ tab. The /GS switch provides a "**speed bump**," or **cookie**, between the buffer and the return address that helps in **preventing buffer overrun**.

If an overflow writes over the return address, it will have to overwrite the cookie put in between it and the buffer, resulting in a new **stack layout**:

- ➊ Function parameters
- ➋ Function return address
- ➌ Frame pointer
- ➍ Cookie
- ➎ Exception Handler frame
- ➏ Locally declared variables and buffers
- ➐ Callee save registers

The screenshot displays the BufferShield security tool. On the left, a sidebar lists features: "BufferShield allows you to detect and prevent the exploitation of buffer overflows, responsible for the majority of code injection attacks" and "Features: Detects code execution on the stack, default heap, dynamic heap, virtual memory, and data segments; Terminates applications in question if a buffer overflow was detected". Below this are two icons: a blue cube with orange arrows and a grey atom-like symbol. On the right, a window titled "Sys-Manage BufferShield Configuration" shows "Global Setting" options like "Terminate processes if detecting unwanted code execution" and "Check application stack". It also includes a note that "By default all options are enabled" and a status bar indicating "Up and running". The URL <http://www.sys-manage.com> is visible at the bottom.

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



BoF Security Tool: BufferShield

Source: <http://www.sys-manage.com>

BufferShield a security tool that prevents the exploitation of buffer overflows. It allows you to detect and prevent the exploitation of **buffer overflows**, responsible for the majority of security related problems.

Key features of BufferShield:

- It detects code execution on the stack, default heap, dynamic heap, virtual memory, and data segments
- It can **terminate** applications in question if a buffer overflow was detected
- It reports to the Windows event log in case of any **detected overflows**
- It allows the definition of a **protection scope** to either protect only defined applications or to exclude certain applications or memory ranges from being protected
- It utilizes Intel XD / AMD NX **hardware based technology** if available
- It has SMP support

- It uses Address Space Layout Randomization (ASLR)

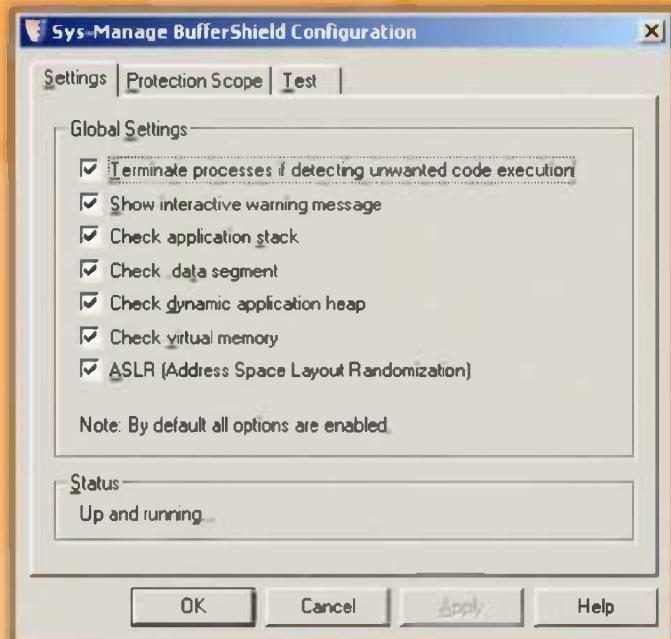


FIGURE 18.16: BufferShield Screenshot

The slide features a title 'BoF Security Tools' in large yellow font, with a 'CEH' logo in the top right corner. Below the title are eight tool entries, each with an icon, name, and URL:

- DefenseWall (<http://www.softsphere.com>)
- FireFuzzer (<http://code.google.com>)
- TIED (<http://www.security.iitk.ac.in>)
- BOON (<http://www.cs.berkeley.edu>)
- LibsafePlus (<http://www.security.iitk.ac.in>)
- The Enhanced Mitigation Experience Toolkit (<http://support.microsoft.com>)
- Comodo Memory Firewall (<http://www.comodo.com>)
- CodeSonar® Static Analysis Tool (<http://www.grammatech.com>)
- Clang Static Analyzer (<http://clang-analyzer.llvm.org>)
- CORE IMPACT Pro (<http://www.coresecurity.com>)

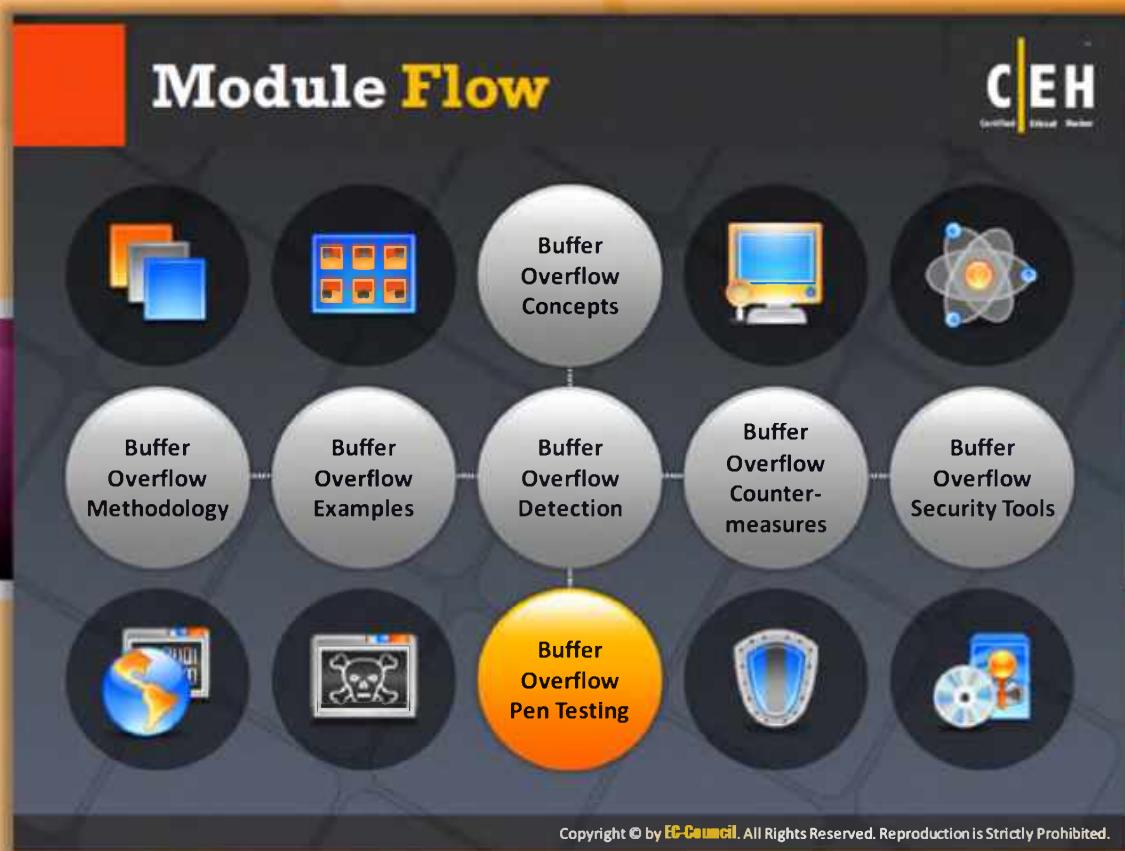
Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



BoF Security Tools

In addition to **/GS** and **BufferShield**, many other buffer overflow security tools are readily available in the market. A more buffer overflow security tools that can detect and prevent buffer overflows are listed as follows:

- DefenseWall available at <http://www.softsphere.com>
- TIED available at <http://www.security.iitk.ac.in>
- LibsafePlus available at <http://www.security.iitk.ac.in>
- Comodo Memory Firewall available at <http://www.comodo.com>
- Clang Static Analyzer available at <http://clang-analyzer.llvm.org>
- FireFuzzer available at <http://code.google.com>
- BOON available at <http://www.cs.berkeley.edu>
- The Enhanced Mitigation Experience Toolkit available at <http://support.microsoft.com>
- CodeSonar® Static Analysis Tool available at <http://www.grammatech.com>
- CORE IMPACT Pro available at <http://www.coresecurity.com>



Module Flow

So far, we have discussed all the necessary elements required to test the security of an application or program against buffer overflow vulnerabilities. Now it's time to test the security of an application, service, or program. The test conducted to check the security of your own application by simulating the actions of an attacker or external user is called penetration testing.

Buffer Overflow Concepts	Buffer Overflow Countermeasures
Buffer Overflow Methodology	Buffer Overflow Security Tools
Buffer Overflow Examples	Buffer Overflow Pen Testing
Buffer Overflow Detection	

This section provides a detailed step-by-step process of testing the security of application, service, or program against buffer overflow flaws.

Buffer Overflow Penetration Testing

CEH Certified Ethical Hacker

Buffer overflow penetration testing is based on the assumption that the application will result in a **system crash** or in **extraordinary behavior** when supplied with **format type specifiers** and input strings that are longer than expected

Skills of a Penetration Tester

- Understanding of how buffer overflow attack works
- Understanding of memory management in various operating environments
- Proficiency in running debuggers, disassemblers, and fuzzers
- Understanding of programming languages such as C/C++, assembly, and machine language

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

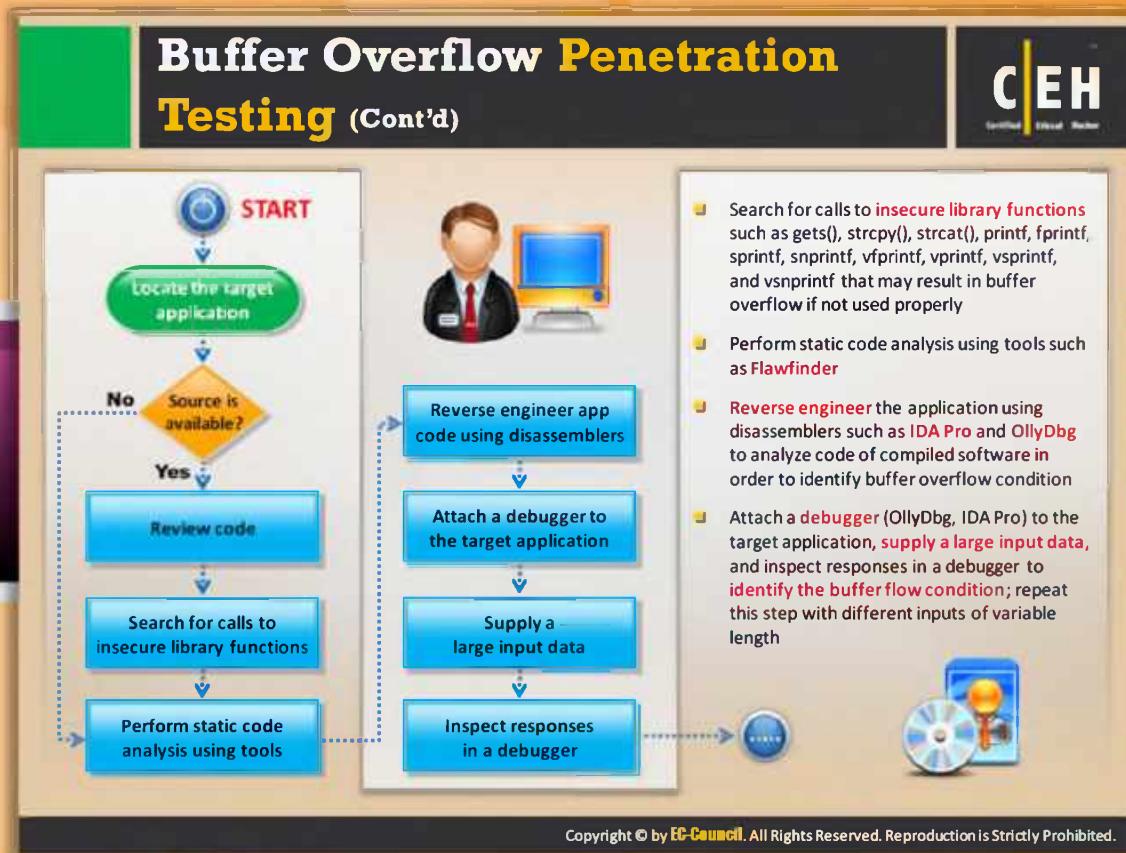


Buffer Overflow Penetration Testing

Buffer overflow penetration testing is based on the assumption that the application will result in a **system crash** or an **extraordinary** behavior when supplied with format type specifies and input strings that are longer than expected. A **penetration tester's** job is to not only scan for the **vulnerabilities** in the applications or server, but also he need to exploit them to gain access to the remote server.

A good pen tester should possess the following skills:

- Understanding of how buffer overflow attack works
- Understating of **memory management** in various operating environments
- Understanding of programming languages such as C/C++, assembly, and machine language
- Proficiency in running debuggers, disassemblers, and fuzzers



Buffer Overflow Penetration Testing (Cont'd)

An application can be tested against buffer overflows by supplying a **larger amount of input data** than the usual. Then you should observe the **application's responses** and **execution flow** to check whether an overflow occurred or not. To test the application against buffer overflow with all possible cases, follow these steps:

Step 1: Locate the target application

In order to perform **penetration testing**, first you should locate the target application on which you want to conduct the test. Then check whether the source code of the **target application** is available or not.

If the source is not available, then go to step 4 to perform **static code analysis** using tools and if the source is available, then review the code.

Step 2: Review code

Review the source code of the application to find the vulnerabilities in the application development and try to **exploit** those vulnerabilities. Test for common vulnerabilities such as buffer overflows, format string exploits, etc.

Step 3: Search for calls to insecure library functions

Library functions make the application development easy but all the library function calls are not secure. Though they seem to be normal, they can be exploited. Hence, you should search for insecure library function calls and **secure** them from **exploitation**.

Step 4: Perform static code analysis using tools

Static code analysis allows you to test the application without actually executing the application. This is usually done with the help of **automated tools** such as **RATS** and **Flawfinder**.

Step 5: Reverse engineer app code using disassemblers

Reverse engineer app code involves testing the **assembly code** with help of disassemblers passively. In this method of testing, various sections of the code are scanned for vulnerable **assembly fragment signatures**.

Step 6: Attach a debugger to the target application

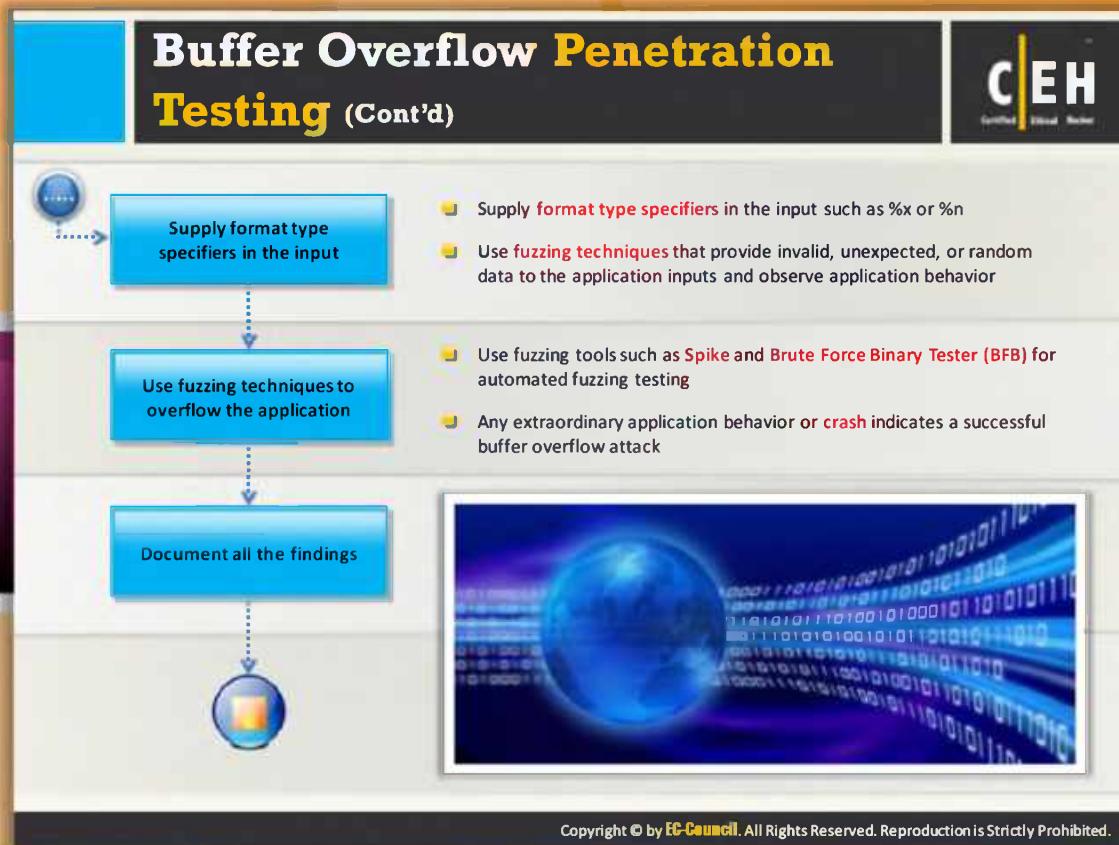
In order to locate and validate a **buffer overflow vulnerability**, you need to attach a debugger to the target application.

Step 7: Supply a large amount of input data

Create a larger string than the actual size and supply it as input data to the application and observe the responses of the application to the given input.

Step 8: Inspect responses in a debugger

The debugger attached to the application allows you to see the execution flow and state of registers when the buffer overflow gets exploited.



Buffer Overflow Penetration Testing (Cont'd)

Step 9: Supply format type specifiers in the input

Supply format type specifiers such as **%x** or **%n** in the application input to test for **format string vulnerabilities** that in turn may lead to buffer overflows.

Step 10: Use fuzzing techniques to overflow the application

Provide invalid, unexpected, or random data as input to the target application using fuzzing techniques and then observe the **application behavior**. This way you can find whether the application is vulnerable to buffer overflows or not. You can also conduct **automated fuzzing** test with the help of fuzzing tools such as **Spike** and **Brute Force Binary Tester (BFB)**.

Step 11: Document all the findings

Documenting all the findings is the last and the most important step that should be carefully carried out. It is the most important step because in this step you need to document all the critical information that can lead to **exploitation**. Sometimes even a small piece of information left out may lead to great losses for a company. Therefore, this step should be done carefully.

Module Summary



A buffer overflow occurs when a program or process tries to store more data in a buffer (temporary data storage area) than it was intended to hold.

Buffer overflow attacks depend on lack of boundary testing, and a machine that can execute a code that resides in the data or stack segment.

A stack-based buffer overflow occurs when a buffer has been overrun in the stack space.

Buffer overflow vulnerability can be detected by skilled auditing of the code as well as boundary testing.

Shellcode is machine level code used as payload in the exploitation of a software vulnerability.

Countermeasures include checking the code, disabling stack execution, supporting a safer C library, and using safer compiler techniques.

Tools like stackguard, Immunix, and vulnerability scanners help in securing systems.

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Module Summary

- ➊ A buffer overflow occurs when a program or process tries to store more data in a buffer (temporary data storage area) than it was intended to hold.
- ➋ Buffer overflow attacks depend on the lack of **boundary testing** and a machine that can execute a code that resides in the data or **stack segment**.
- ➌ A stack-based buffer overflow occurs when a buffer has been overrun in the stack space.
- ➍ A buffer overflow **vulnerability** can be detected by **skilled auditing** of the code as well as boundary testing.
- ➎ Shellcode is small code used as payload in the exploitation of a software vulnerability.
- ➏ Countermeasures include checking the code, disabling stack execution, supporting a safer C library, and using **safer compiler techniques**.
- ➐ Tools such as stackguard, Immunix, and vulnerability scanners help in **securing systems**.

Cryptography

Module 19





The slide features a dark background with the title 'Cryptography' in large yellow font at the top center. Below it, 'Module 19' is written in a smaller white font. A horizontal bar near the bottom contains the text 'Engineered by Hackers. Presented by Professionals.' in white. At the bottom, there is a row of five colored icons: a black square with 'CEH Certified Ethical Hacker' text and logo, a green square with a person icon, a blue square with a key icon, a yellow square with a laptop icon, and an orange square with a globe icon.

Ethical Hacking and Countermeasures v8

Module 19: Cryptography

Exam 312-50

The screenshot shows a news article from Techworld.com. The header reads "Security News" and features the "CEH Certified Ethical Hacker" logo. The main headline is "Ransom Malware Hits Australia as 30 Businesses Attacked". The date is 01 October 2012. The article text discusses the 2012 epidemic of ransom malware, mentioning 30 Australian businesses affected, including medical, entertainment, retail, and insurance sectors. It details a recent incident where a business paid AUD \$3,000 via Western Union for financial records. The text also notes the use of 256-bit encryption and the difficulty of cracking it if the key is not exposed. A quote from Detective Superintendent Brian Hay is included. The source URL is <http://news.techworld.com>. The copyright notice at the bottom states "Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited."



Security News

Ransom Malware Hits Australia as 30 Businesses Attacked

Source: <http://news.techworld.com>

The 2012 epidemic of ransom malware appears to have turned even nastier with reports that as many as 30 Australian businesses have now asked police for help coping with attacks in a matter of days.

According to local news, police in the state of **Queensland** have received reports from a dozen businesses while many other are believed to have chosen to keep incidents to themselves.

Businesses affected included those in the medical, entertainment, retail and insurance sectors, the news source said, with several dozen affected in total.

In one recent incident, a business in the Northern Territories reportedly paid an AUD \$3,000 (about £2,000) ransom via Western Union to get back access to important financial records, including credit card data and debtor invoices. The attackers demanded the money within seven days or the sum would increase by **AUD \$1,000 per week**.

Worryingly, this attack used, to all intents and purposes impossible to crack if the key has not been exposed during the attack.

"A lot of businesses can't afford the interruptions to their trade and will pay straight away," detective superintendent Brian Hay of **Queensland's fraud** and corporate crime group told press.

Ransom malware has become a serious issue during 2012, although its effect on businesses is rarely recorded. Most of the data that has become public has been in the form of police warnings based on attacks against consumers.

Most attacks simply attempt to engineer users into believing their files are encrypted when they are not or make more general threats, often to report victims to national police for non-existent crimes.

The use of industrial-strength encryption is rare although this sort of technique is actually where the form started as long ago in 2006 with a piece of malware called '**Cryzip**'.

In August, the FBI said it had been "**inundated**" with ransom malware reports from consumers, not long after the UK's Police Central e-Crime Unit (PCeU) publicised an identical spate of attacks that had affected over a thousand PCs in the UK.

In the past the few security companies that have investigated the issue have pinned the blame on a single cabal of Russian criminals that seem able to operate with impunity. Now the same tactics appear to have spread to gangs in nearby countries such as the **Ukraine** and **Romania**.

The suspicion is that some security vendors say little about the problem because not only is their software unable to stop infections but they can't always unlock the files after the fact either.



All contents © IDG 2012

By: John E Dunn

<http://news.techworld.com/security/3401328/ransom-malware-hits-australia-as-30-businesses-attacked/>

Module Objectives



The slide features two columns of objectives. An orange arrow points from the left column to the right column. Below the columns are three icons: a whiteboard, a user profile, and a stack of books.

Cryptography	Digital Signature
Encryption Algorithms	Disk Encryption
Ciphers	Disk Encryption Tool
What Is SSH (Secure Shell)?	Cryptography Attacks
Cryptography Tools	Code Breaking Methodologies
Public Key Infrastructure (PKI)	Cryptanalysis Tools
Certification Authorities	Online MD5 Decryption Tools

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Module Objectives

Having dealt with various security concerns and countermeasures in the preceding modules, it is obvious that cryptography, as a security measure, is here to stay. This module will familiarize you with:

- Cryptography
- Encryption Algorithms
- Ciphers
- What Is SSH (Secure Shell)?
- Cryptography Tools
- Public Key Infrastructure (PKI)
- Certification Authorities

- Digital Signature
- Disk Encryption
- Disk Encryption Tool
- Cryptography Attacks
- Code Breaking Methodologies
- Cryptanalysis Tools
- Online MD5 Decryption Tools



Module Flow

To understand cryptography security measures, let's begin with **cryptography** and its **associated concepts**.

Cryptography Concepts	Encryption Algorithms
Cryptography Tools	Public Key Infrastructure (PKI)
Email Encryption	Disk Encryption
Cryptography Attacks	Cryptanalysis Tools

This section describes cryptography and the types of cryptography.

Cryptography is the **conversion of data** into a scrambled code that is decrypted and sent across a private or public network.

Cryptography is used to protect confidential data such as **email messages**, chat sessions, web transactions, personal data, **corporate data**, e-commerce applications, etc.

Protection

Objectives

- Confidentiality
- Integrity
- Authentication
- Non-Repudiation

Process

```
graph LR; A[Plaintext] -- Encryption --> B[Ciphertext]; C[Ciphertext] -- Decryption --> D[Plaintext]
```

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Cryptography

Everyone has secrets, and when it is necessary to transfer that secret information from one person to another, it's very important to protect that information or data during the transfer. Cryptography takes plaintext and transforms it into an unreadable form (ciphertext) for the purpose of maintaining security of the data being transferred. It uses a key to transform it back into readable data when the information reaches its destination. The word crypto is derived from the Greek word kryptos. **Kryptos** was used to depict anything that was **concealed**, hidden, veiled, secret, or mysterious. Graph is derived from graphia, which means writing; hence, cryptography means the art of "the secret writing."

Cryptography is the study of mathematical techniques involved in information security such as confidentiality, data integrity, entity authentication, and data origin authentication. Cryptography transforms plaintext messages to ciphertext (encrypted messages) by means of encryption. Modern cryptography techniques are virtually unbreakable, though it is possible to break encrypted messages by means of **cryptanalysis**, also called code breaking. There are four main objectives of cryptography:



Confidentiality

According to the **International Standards Organization** (ISO), confidentiality is "ensuring that the information/data can be accessed only by those authorized." Confidentiality is the

term used to describe the prevention of revealing information to unauthorized computers or users.

Any breach in confidentiality may lead to both financial and emotional distress. There have been instances of organizations going bankrupt due to a system breach by **rival organizations**. Moreover, personal information in the wrong hands can ruin the lives of system users. Therefore, only authorized users should possess access to information.



Integrity

Integrity is “ensuring that the information is accurate, complete, reliable, and is in its original form.” Valuable information is stored on the computer. Any data corruption/modification can reduce the value of the information. The damage that data corruption/modification can do to an organization is **unfathomable**.

Integrity of the data is affected when an insider (employee) of an organization or an attacker deletes/alters important files or when malware infects the computer.

Although it may be possible to restore the modified data to an extent, it is impossible to restore the value and reliability of the information.

Examples of violating the data integrity include:

- ⌚ A frustrated employee deleting important files and modifying the payroll system
- ⌚ **Vandalizing** a website and so on



Authentication

Authenticity is “the identification and assurance of the origin of information.” It is important to ensure that the information on the system is authentic and has not been tampered with. It is also important to ensure that the computer users or those who access information are who they claim to be.



Nonrepudiation

In digital security, **nonrepudiation** is the means to ensure that a message transferred has been sent and received by the persons or parties who actually intended to. Let us assume that party A is sending a message M with the signature S to the party B. Then party A cannot deny the authenticity of its signature S. It can be obtained through the use of:

- ⌚ **Digital signatures:** A digital signature functions as unique identifier for an individual, like a written signature. It is used to ensure that a message or document is electronically signed by the person.
- ⌚ **Confirmation services:** It is possible to indicate that messages are received and/or sent by creating digital receipts. These digital receipts are generated by the message transfer agent.



FIGURE 19.1: Illustrating cryptography process

Types of Cryptography

C|EH
Certified Ethical Hacker

Symmetric Encryption

Symmetric encryption (secret-key, shared-key, and private-key) **uses the same key** for encryption as it does for decryption

The diagram illustrates symmetric encryption with a central key icon. An arrow labeled "Encryption" points from a document labeled "Plain text" to a document labeled "Cipher text". Another arrow labeled "Decryption" points from the "Cipher text" document back to the "Plain text" document.

Asymmetric Encryption

The diagram illustrates asymmetric encryption with two separate key icons. One key is labeled "Encryption" and the other "Decryption". An arrow labeled "Encryption" points from a document labeled "Plain text" to a document labeled "Cipher text". Another arrow labeled "Decryption" points from the "Cipher text" document back to the "Plain text" document.

Symmetric Encryption

The diagram illustrates symmetric encryption with a central key icon. An arrow labeled "Encryption" points from a document labeled "Plain text" to a document labeled "Cipher text". Another arrow labeled "Decryption" points from the "Cipher text" document back to the "Plain text" document.

Asymmetric Encryption

Asymmetric encryption (public-key) **uses different encryption keys** for encryption and decryption. These keys are known as public and private keys

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Types of Cryptography

The following are the two types of cryptography:

- ⦿ Symmetric encryption (secret key cryptography)
- ⦿ Asymmetric encryption (public key cryptography)



Symmetric Encryption

The symmetric encryption method uses the same key for encryption and decryption. As shown in the following figure, the sender uses a key to encrypt the plaintext and sends the ciphertext to the receiver. The receiver **decrypts** the **ciphertext** with the same key that is used for encryption and reads the message in plaintext. As a single secret key is used in this process symmetric encryption is also known as secret key cryptography. This kind of **cryptography** works well when you are communicating with only a few people.

Symmetric Encryption



FIGURE 19.2: Symmetric Encryption method

The problem with the secret key is transferring it over the large network or Internet while preventing it from falling into the wrong hands. In this process, anyone who knows the secret key can decrypt the message. This problem can be fixed by **asymmetric encryption**.



Asymmetric Encryption

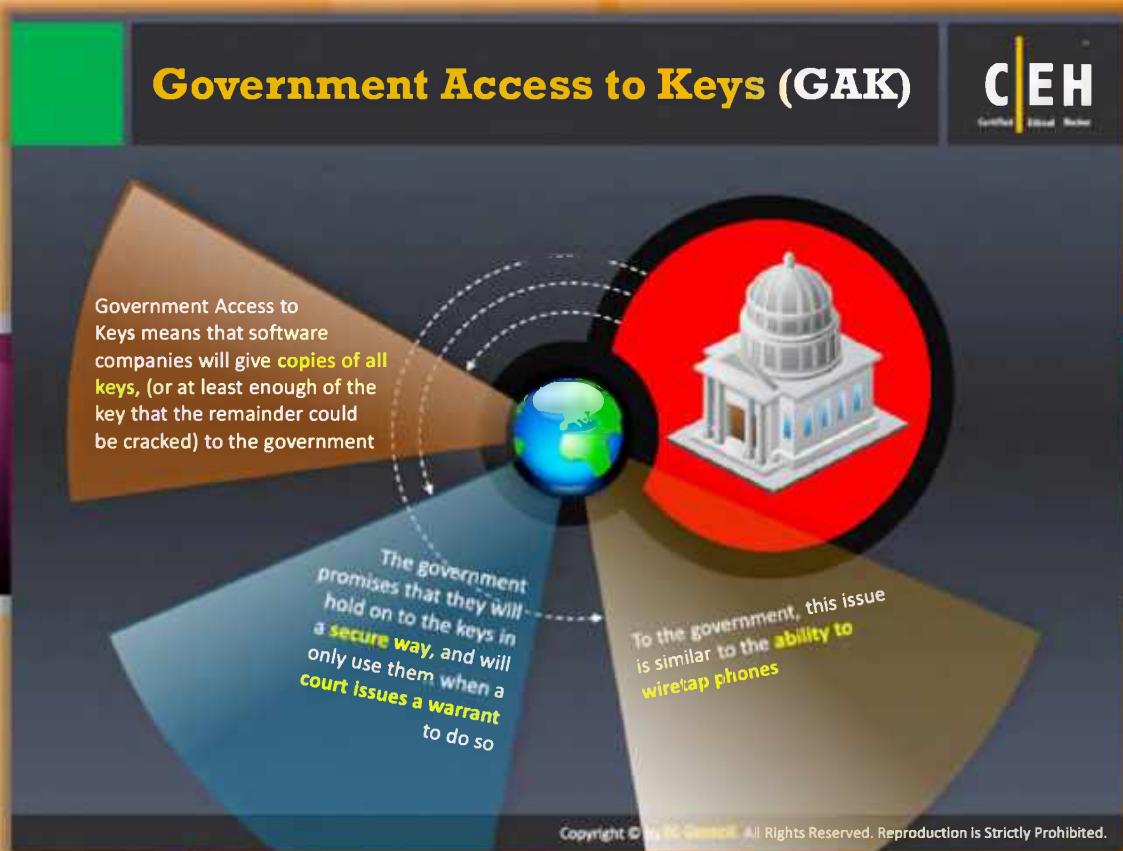
Asymmetric cryptography uses different keys for encryption and decryption. In this type of cryptography, an end user on a public or private network has a pair of keys: a public key for encryption and a private key for decryption. Here, a **private key** cannot be derived from the **public key**.

The asymmetric cryptography method has been proven to be secure against attackers. In asymmetric cryptography, the sender encodes the message with the help of a public key and the receiver decodes the message using a random key generated by the **sender's public key**.

Asymmetric Encryption



FIGURE 19.3: Asymmetric Encryption method



Government Access to Keys (GAK)

A key escrow encryption system provides the decrypting capability to certain authorized personnel, under stipulated conditions, and can decrypt the data.

The **data recovery keys** for encrypting and decrypting the data are not similar, but they inform a method to determine the encryption and decryption keys. They include a key escrow (used to refer the safeguard the data keys), key archive, key backup, and data recovery system.

Key recovery systems have gained **prominence** due to the desire of government intelligence and law enforcement agencies to guarantee they have access to the encrypted information without the knowledge or consent of encryption users.

A well-designed cryptosystem provides security by recovering the encrypted data without proper information about the correct key. The maintenance of such **high-security** measures may cause problems to the owner of the encrypted data if the owner loses the key.

The eventual goal of government-driven recovery encryption, as stated in the US Department of Commerce's recent encryption regulations, "Envisions a worldwide key management infrastructure with the use of key escrow and key recovery encryption items."

The Clipper Chip is a **hardware-based** cryptographic device used to secure private communications by simultaneously authorizing government agents to obtain the keys upon giving it, vaguely termed "**legal authorization**".

The keys are split between two government escrow agencies. This helps the government in accessing private communication channels. A device called **Clipper** is used to encrypt voice communications and a similar device called Capstone is used to encrypt the data.

The National Security Agency (NSA) is a secret US military intelligence agency responsible for capturing foreign government communications, and cracking the codes of protected transmissions that are developed with an algorithm known as **Skipjack**.

The Skipjack algorithm uses **80-bit keys**. Crypt analyzing requires searching through all keys, which makes it sixteen million times as hard to break as DES.

From the user's viewpoint, any key escrow system diminishes security. It puts the potential for access to the user's communications in the hands of **escrow** agencies, whose intentions, policies, security capabilities, and future cannot be known.

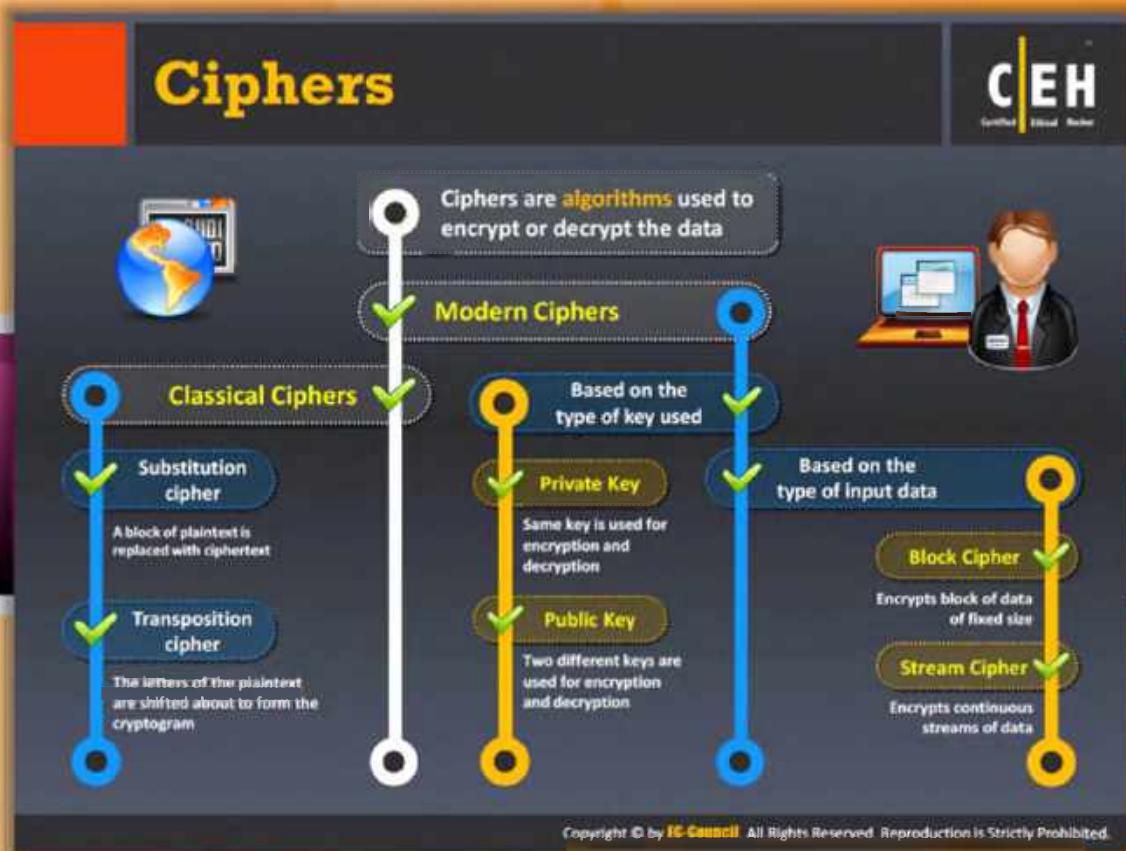


Module Flow

So far, we have discussed cryptography and the concepts associated with it. Now we will discuss encryption key concepts of cryptography. There are many mechanisms, i.e., encryption algorithms, that allow you to encrypt the plaintext.

Cryptography Concepts	Encryption Algorithms
Cryptography Tools	Public Key Infrastructure (PKI)
Email Encryption	Disk Encryption
Cryptography Attacks	Cryptanalysis Tools

This section describes ciphers and various encryption algorithms such as **AES, DES, RC4, RC5, RC6, DSA, RSA, MD5, and SSH**.



Ciphers

Cryptography refers to secret writing and a cipher is nothing more than an algorithm used for both **encryption** as well as **decryption**. The traditional method of encoding and decoding used to be in a different format, which provided numbering for each letter of the alphabet and used to encode the given message. If the attacker also knew the **numbering system**, he or she could decode it.

In cryptography, the cipher algorithm used for encoding is known as enciphering and decoding is known as deciphering.

Example:

a b c d e f g h...z are given in codes of numerical numbers, such as 1 2 3 4 5...26.

The message can be encoded based on this example and can be decoded as well. In a cipher, the message appears as plaintext but has been encoded through a key. Based on the requirements the key could be a symbol or some other form of text. If the message is highly confidential, then the key is restricted to the sender and recipient, but in some cases in open domains, some keys are shared without affecting the main data.

There are various types of ciphers:

Classical Ciphers

 Classical ciphers are the most basic type of ciphers that operate on **alphabet letters**, such as A-Z. These are usually implemented either by hand or with **simple mechanical devices**. These are not very reliable. There are two types of classical ciphers:

- ⊕ **Substitution cipher:** The units of plaintext are replaced with ciphertext. It replaces bits, characters, or blocks of characters with different bits, characters, or blocks.
- ⊕ **Transposition cipher:** The letters of the plaintext are shifted to form the **cryptogram**. The ciphertext is a permutation of the plaintext.

Modern Ciphers

 Modern ciphers are designed to withstand a **wide range of attacks**. Modern ciphers provide message secrecy, integrity, and authentication of the sender. The modern ciphers are calculated with the help of a one-way mathematical function that is capable of **factoring large prime numbers**. Modern ciphers are again classified into two categories based on the type of key and the input data. They are:

Based on the type of key used

- ⊕ **Private-key cryptography** (symmetric key algorithm): The same key is used for encryption and decryption.
- ⊕ **Public-key cryptography** (asymmetric key algorithm): Two different keys are used for encryption and decryption.

Based on the type of input data

- ⊕ **Block ciphers:** Refer to an algorithm operating on block (group of bits) of fixed size with an unvarying transformation specified by a symmetric key.
- ⊕ **Stream ciphers:** Refer to symmetric key ciphers. This is obtained by combining the plaintext digits with a key stream (pseudorandom cipher digit stream).

Data Encryption Standard (DES)

The algorithm is designed to **encipher** and **decipher** blocks of data consisting of **64 bits** under control of a 56-bit key

DES is the **archetypal block cipher** — an algorithm that takes a fixed-length string of plaintext bits and transforms it into a ciphertext bitstring of the same length

Due to the **inherent weakness** of DES with today's technologies, some organizations repeat the process three times (3DES) for added strength, until they can afford to update their equipment to AES capabilities

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Data Encryption Standard (DES)

DES is the name of the Federal information Processing Standard (FIPS) 46-3 that describes the data encryption algorithm (DEA). It is a **symmetric cryptosystem designed** for **implementation** in hardware and used for single-user encryption, such as to store files on a hard disk in encrypted form.

DES gives **72 quadrillion** or more possible encryption keys and chooses a random key for each message to be encrypted. Though DES is considered to be strong encryption, at present, triple DES is used by many organizations. **Triple DES** applies three keys successively.

Advanced Encryption Standard (AES)

AES is a **symmetric-key algorithm** for securing sensitive but unclassified material by U.S. government agencies

AES is an **iterated block cipher**, which works by repeating the same operation multiple times

It has a **128-bit block size**, with key sizes of 128, 192, and 256 bits, respectively for AES-128, AES-192, and AES-256



AES Pseudocode

```
Cipher (byte in[4*Nb], byte out[4*Nb],  
word w[Nb*(Nr+1)])  
begin  
    byte state[4,Nb]  
    state = in  
    AddRoundKey(state, w)  
    for round = 1 step 1 to Nr-1  
        SubBytes(state)  
        ShiftRows(state)  
        MixColumns(state)  
        AddRoundKey(state, w+round*Nb)  
    end for  
    SubBytes(state)  
    ShiftRows(state)  
    AddRoundKey(state, w+Nr*Nb)  
    out = state  
end
```

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Advanced Encryption Standard (AES)

The Advanced Encryption Standard (AES) is a **National Institute of Standards and Technology specification** for the encryption of electronic data. It can be used to encrypt digital information such as telecommunications, financial, and government data. AES consists of a symmetric-key algorithm, i.e., both encryption and decryption are performed using the same key.

It is an iterated block cipher that works by repeating the defined steps multiple times. This has a 128-bit block size, with key sizes of 128, 192, and 256 bits, respectively, for AES-128, AES-192, and AES-256.

AES Pseudo code

Initially, the cipher input is copied into the internal state and then an initial round key is added. The state is transformed by iterating a round function in a number of cycles. Based on the block size and key length, the number of cycles may vary. Once rounding is completed, the final state is copied into the **cipher output**. Cipher (byte in [4*Nb], byte out [4*Nb], word w[Nb*(Nr+1)])

```
begin  
    byte state[4, Nb]  
    state = in
```

```
AddRoundKey (state, w)
for round = 1 step 1 to Nr-1
    SubBytes(state)
    ShiftRows(state)
    MixColumns(state)
    AddRoundKey(state, w+round*Nb)
end for
SubBytes(state)
ShiftRows(state)
AddRoundKey(state, w+Nr*Nb)
out = state
end
```

RC4, RC5, RC6 Algorithms

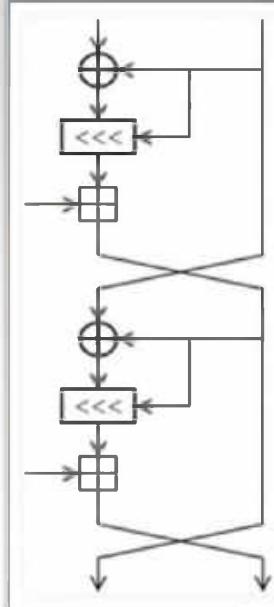
CEH
Certified Ethical Hacker

RC4
A variable **key size stream cipher** with byte-oriented operations, and is based on the use of a random permutation

RC5
It is a **parameterized algorithm** with a variable block size, a variable key size, and a variable number of rounds. The key size is 128-bits

RC6
RC6 is a symmetric key block cipher derived from RC5 with two additional features:

- Uses Integer multiplication
- Uses four 4-bit working registers (RC5 uses two 2-bit registers)



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



RC4, RC5, and RC6 Algorithms

The **encryption algorithms** developed by RSA Security are:



RC4

RC4 is a stream cipher for RSA Security, which Rivest designed. It is a variable key-size stream cipher with byte-oriented operations and is based on the use of a **random permutation**. According to some analysis, the period of the cipher is likely to be greater than 10100. For each output byte, eight to sixteen system operations are used, which means the cipher can run fast in software. Independent analysts have had a careful and critical look at the algorithm, and it is considered secure. Products like RSA **SecurPC** use this algorithm for file encryption. Rc4 is also used for safe communications like traffic encryption, which secures websites and from secure websites with **SSL protocol**.

RC5



RC5 is a **block cipher** known for its **simplicity**. Ronald Rivest designed it. This algorithm has a variable block size and key size and a variable number of rounds. The choices for the block-size are 32 bits, 64 bits, and 128 bits. The iterations range from 0 to 255; whereas the key sizes have a range from 0 to 2040 bits. It has three routines: key expansion, encryption, and decryption.



RC6

It is a block cipher that is based on RC5. Like in RC5, the block size, the key size, and the number of rounds are variable in the RC6 algorithm. The key-size ranges from 0 bits to 2040. In addition to RC5, RC6 has two more features, which are the addition of integer multiplication and the usage of four **4-bit** working registers as an alternative to RC5's two 2-bit registers.

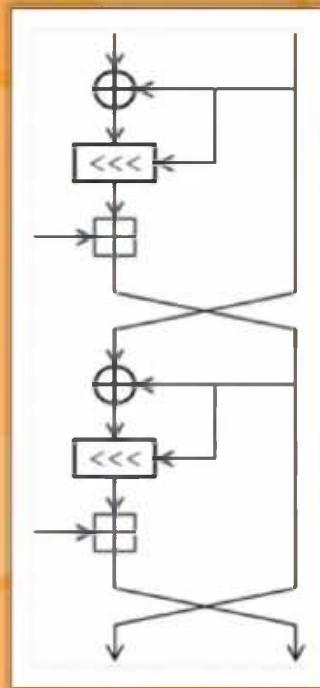


FIGURE 19.4: RC5 block cipher

The DSA and Related Signature Schemes

Digital Signature Algorithm

FIPS 186-2 specifies the Digital Signature Algorithm (DSA) that may be used in the generation and verification of digital signatures for sensitive, unclassified applications



Digital Signature

The digital signature is computed using a set of rules (i.e., the DSA) and a set of parameters such that the identity of the signatory and integrity of the data can be verified

Each entity creates a public key and corresponding private key

1. Select a prime number q such that $2^{159} < q < 2^{160}$
2. Choose t so that $0 \leq t \leq 8$
3. Select a prime number p such that $2^{511+64t} < p < 2^{512+64t}$ with the additional property that q divides $(p-1)$
4. Select a generator α of the unique cyclic group of order q in \mathbb{Z}_p^*
5. To compute α , select an element g in \mathbb{Z}_p^* and compute $\alpha = g^{(p-1)/q} \pmod{p}$
6. If $\alpha = 1$, perform step five again with a different g
7. Select a random a such that $1 \leq a \leq q-1$
8. Compute $y = \alpha^a \pmod{p}$

The public key is (p, q, α, y) . The private key is a .



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



The DSA and Related Signature Schemes

A digital signature is a mathematical scheme used for the authentication of a digital message. Digital Signature Algorithm (DSA) is intended for its use in the U.S. Federal Information Processing Standard (FIPS 186) called the **Digital Signature Standard** (DSS). DSA was actually proposed by the National Institute of Standards and Technology (NIST) in August 1991. NIST made the U.S. Patent 5,231,668 that covers DSA available worldwide freely. It is the first digital signature scheme recognized by any government.

A digital signature algorithm includes a signature generation process and a **signature verification process**.

Signature Generation Process: The private key is used to know who has signed it.

Signature Verification Process: The public key is used to verify whether the given digital signature is genuine or not.

As to the popularity of online shopping grows, e-payment systems and various other electronic payment modes rely on various systems like DSA.

Benefits of DSA:

- ⊖ Less chances of forgery as it is in the case of **written signature**.
- ⊖ Quick and easy method of business transactions.
- ⊖ Fake currency problem can be drastically reduced.

DSA, with its uses and benefits, may bring revolutionary changes in the future.

RSA (Rivest Shamir Adleman)

C|EH
Certified Ethical Hacker

-  RSA is an **Internet encryption and authentication system** that uses an algorithm developed by Ron Rivest, Adi Shamir, and Leonard Adleman
-  RSA encryption is widely used and is one of the **de-facto encryption standard**
-  It uses **modular arithmetic** and **elementary number theories** to perform computations using two large prime numbers

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



RSA (Rivest Shamir Adleman)

RSA is a public-key cryptosystem. It uses modular arithmetic and elementary number theories to perform computations using two large prime numbers. **RSA encryption** is widely used and is the de-facto encryption standard.

Ron Rivest, Adi Shamir, and Leonard Adleman formulated RSA, a public key cryptosystem for encryption and authentication. It is usually used with a secret key cryptosystem, like DES. The RSA system is widely used in a variety of products, platforms, and industries. Many operating systems like Microsoft, Apple, Sun, and Novell build the RSA algorithms into the existing versions. It can also be found on hardware secured telephones, on Ethernet network cards, and on smart cards. Consider that Alice uses the RSA technique to send Bob a message. If **Alice desires to communicate with Bob**, she encrypts the message using a randomly chosen DES key and sends it to Bob. Then she will look up Bob's public key and use it to encrypt the DES key. The RSA digital envelope, which is sent to Bob by Alice, consists of a DES-encrypted message and RSA-encrypted DES key. When Bob receives the digital envelope, he will decrypt the DES key with his private key, and then use the DES key to decrypt the message itself. This system combines the **high speed of DES with the key management convenience of the RSA system**.

The working of RSA is as follows: Two large prime numbers are taken (say "a" and "b"), and their product is determined ($c = ab$, where "c" is called the modulus). A number "e" is chosen such that it is less than "c" and relatively prime to $(a-1)(b-1)$, which means that "e" and $(a-1)(b-1)$ have no common factors other than 1.

1) have no common factors except 1. Apart from this, another number "f" is chosen such that $(ef - 1)$ is divisible by $(a-1)(b-1)$. The values "e" and "f" are called the public and private exponents, respectively. The public key is the pair (c, e) ; the private key is the pair (c, f) . It is considered to be difficult to obtain the private key "f" from the public key (c, e) . However, if someone can factor "c" into "a" and "b", then he or she can decipher the private key "f". The security of the RSA system is based on the assumption that such factoring is difficult to carry out, and therefore, the cryptographic technique is safe.

Example of RSA Algorithm

C|EH
Certified Ethical Hacker

```
P = 61    <= first prime number (destroy this after computing E and D)
Q = 53    <= second prime number (destroy this after computing E and D)
PQ = 3233 <= modulus (give this to others)
E = 17    <= public exponent (give this to others)
D = 2753  <= private exponent (keep this secret!)
Your public key is (E,PQ).
Your private key is D.

The encryption function is: encrypt(T) = (T^E) mod PQ
                           = (T^17) mod 3233

The decryption function is: decrypt(C) = (C^D) mod PQ
                           = (C^2753) mod 3233

To encrypt the plaintext value 123, do this:
encrypt(123) = (123^17) mod 3233
               = 337587917446653715596592958817679803 mod 3233
               = 855

To decrypt the cipher text value 855, do this:
decrypt(855) = (855^2753) mod 3233
               = 123
```



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Example of RSA Algorithm

RSA retains its security through the apparent difficulty in **factoring large composites**. Yet there is a possibility of discovering the polynomial time factoring algorithm using the advance number theory. There are three factors that can **aggravate** the path towards compromising RSA security. The advances include factoring technique, computing power, and decrease in the expenditure of the hardware. The working of RSA as explained before is illustrated in the following example. For P = 61 and Q = 53, PQ = 3233. Taking a public exponent, E = 17, and a private exponent, D = 2753, it can be encrypted into plain text 123 as shown as follows:

P = 61 <= first prime number (destroy this after computing E and D)

Q = 53 <= second prime number (destroy this after computing E and D)

PQ = 3233 <= modulus (give this to others)

E = 17 <= public exponent (give this to others)

D = 2753 <= private exponent (keep this secret!)

Your public key is (E,PQ).

Your private key is D.

The encryption function is: $\text{encrypt}(T) = (T^E) \bmod PQ$
 $= (T^{17}) \bmod 3233$

The decryption function is: $\text{decrypt}(C) = (C^D) \bmod PQ$
 $= (C^{2753}) \bmod 3233$

To encrypt the plaintext value 123, do this:

$$\begin{aligned}\text{encrypt}(123) &= (123^{17}) \bmod 3233 \\ &= 337587917446653715596592958817679803 \bmod 3233 \\ &= 855\end{aligned}$$

To decrypt the cipher text value 855, do this:

$$\begin{aligned}\text{decrypt}(855) &= (855^{2753}) \bmod 3233 \\ &= 123\end{aligned}$$

The RSA Signature Scheme

CEH
Certified Ethical Hacker

Algorithm Key generation for the RSA signature scheme

SUMMARY: each entity creates an RSA public key and a corresponding private key. Each entity A should do the following:

1. Generate two large distinct random primes p and q , each roughly the same size.
2. Compute $n = pq$ and $\phi = (p - 1)(q - 1)$.
3. Select a random integer e , $1 < e < \phi$, such that $\gcd(e, \phi) = 1$.
4. Use the extended Euclidean algorithm (Algorithm 2.107) to compute the unique integer d , $1 < d < \phi$, such that $ed \equiv 1 \pmod{\phi}$.
5. A's public key is (n, e) . A's private key is d .

Algorithm RSA signature generation and verification

SUMMARY: entity A signs a message $m \in \mathcal{M}$. Any entity B can verify A's signature and recover the message m from the signature.

1. **Signature generation:** Entity A should do the following:
 - (a) Compute $\tilde{m} = R(m)$, an integer in the range $[0, n - 1]$.
 - (b) Compute $s = \tilde{m}^d \pmod{n}$.
 - (c) A's signature for m is s .
2. **Verification:** To verify A's signature s and recover the message m , B should:
 - (a) Obtain A's authentic public key (n, e) .
 - (b) Compute $\tilde{m} = s^e \pmod{n}$.
 - (c) Verify that $\tilde{m} \in \mathcal{M}_R$; if not, reject the signature.
 - (d) Recover $m = R^{-1}(\tilde{m})$.

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



The RSA Signature Scheme

RSA is used for both public key encryption and for a digital signature (to sign a message). The RSA signature scheme is the first technique used to generate **digital signatures**. It is a deterministic digital signature scheme that provides message recovery from the signature itself. It is the most practical and versatile technique available.

RSA involves both a public key and a private key. The public key, as the name indicates, means any person can use it for **encrypting messages**. The messages that are encrypted with the public key can only be decrypted with the help of the private key.

Consider that John encrypts his document M using his private key S_A , thereby creating a signature $S_{john}(M)$. John sends M along with the signature $S_{john}(M)$ to Alice. Alice decrypts the document using Alice's public key, thereby verifying **John's signature**.



RSA key generation

The procedure for RSA key generation is common for all the RSA-based signature schemes. To generate an RSA key pair, i.e., both an **RSA public key** and corresponding private key, each entity A should do the following:

- Select two large distinct primes' p and q arbitrarily, each of roughly the same bit length
- Compute $n=pq$ and $\phi=(p-1)(q-1)$

- ⊕ Choose a random integer e , $1 < e < \phi$ such that $\text{get}(e, \phi) = 1$
- ⊕ Use the extended **Euclidean algorithm** in order to compute the unique integer d , $1 < d < \phi$ such that $ed \equiv 1 \pmod{\phi}$
- ⊕ The public key of A is (n, e) and private key is d

Destroy p and q at the end of the key generation

The RSA signature is generated and verified in the following way.



Signature generation

In order to sign a message m , A does the following:

- ⊕ Compute $m^* = R(m)$ an integer in $[0, n-1]$
- ⊕ Compute $s = m^{*d} \pmod{n}$
- ⊕ A's signature for m is s



Signature verification

In order to verify A's signature s and recover message m , B should do the following:

- ⊕ Obtain A's authentic public key (e, n)
- ⊕ Compute $m^* = s^e \pmod{n}$
- ⊕ Verify that m^* is in M_R ; if not, reject the signature
- ⊕ Recover $m = R^{-1}(m^*)$

Message Digest (One-way Hash) Functions

Hash functions calculate a unique fixed-size bit string representation called a message digest of any arbitrary block of information.

If any given bit of the function's input is changed, every output bit has a 50 percent chance of changing.

It is computationally infeasible to have two files with the same message digest value.

Note: Message digests are also called one-way hash functions because they cannot be reversed.

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Message Digest (One-way Hash) Functions

Message digest functions distill the information contained in a file (small or large) into a single large number, typically between **128- and 256-bits in length**. Message digest functions calculate a unique fixed-size bit string representation called hash value of any arbitrary block of information. The best message digest functions combine these mathematical properties. Every bit of the message digest function is influenced by every bit of the function's input. If any given bit of the function's input is changed, every output bit has a **50 percent** chance of changing. Given an input file and its corresponding message digest, it should be infeasible to find another file with the same message digest value.

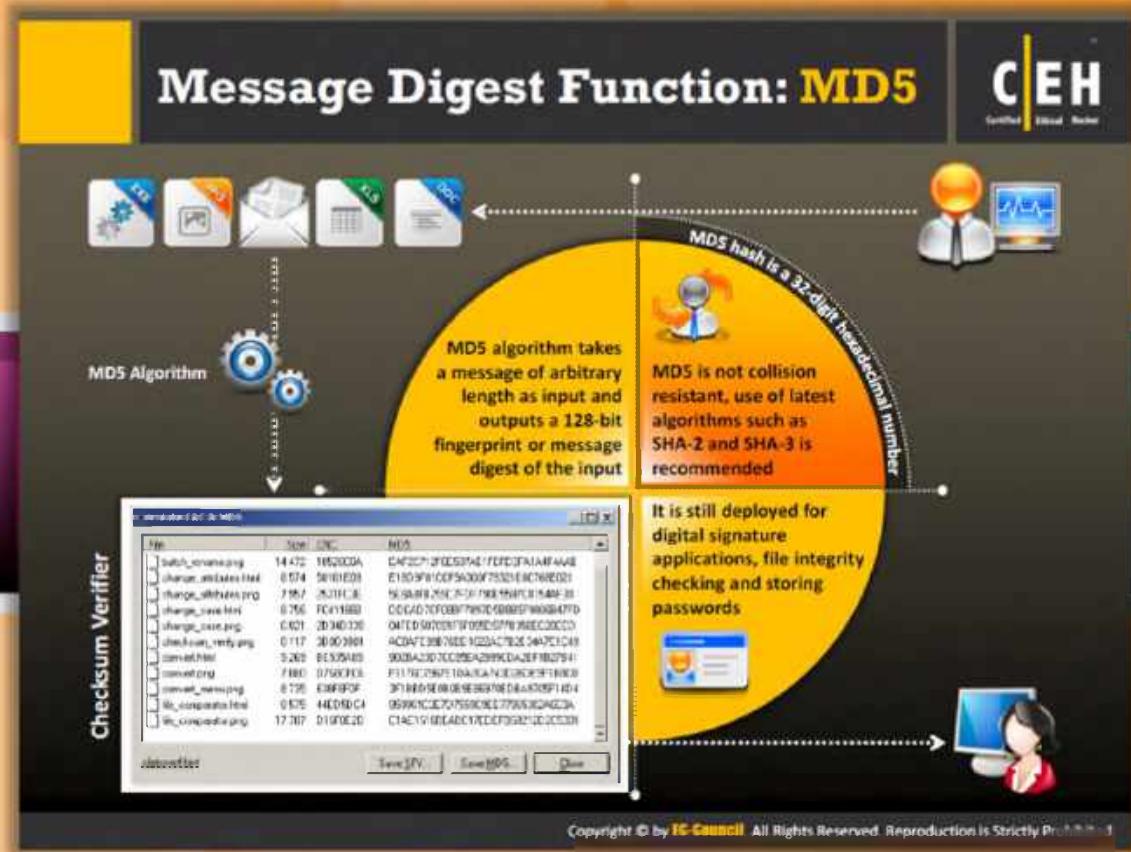
Message digests are also called one-way hash functions because they produce values that are difficult to invert, resistant to attack, mostly unique, and widely distributed.

Message digest functions:

- ⊕ HMAC
- ⊕ MD2
- ⊕ MD4
- ⊕ MD5
- ⊕ SHA



FIGURE 19.5: SHA1 a Message digest function



Message Digest Function: MD5

H is a hash function that is a transformation that accepts a variable of any size as an input, m, and returns a string of a certain size. This is called the hash value h. i.e. $h=H(m)$. The fundamental requirements for the **cryptographic hash** functions are:

- ⌚ Input of any length
- ⌚ Output of a fixed length

And $H(x)$, can be easily computed for any value of x and it must be one-way (i.e., it cannot be inverted and it has an infeasible computation for the given input) and collision free. H is considered to be a weak collision free hash function if the given message x is **infeasible** to find a message y, so that $H(x)=H(y)$. It is a collision free hash function if it is infeasible to find any two messages x and y such that $H(x)=H(y)$.

The main role of a cryptographic hash function is to provide digital signatures. Hash functions are relatively faster than digital signature algorithms; hence, its characteristic feature is to calculate the signature of the document's hash value, which is smaller than the document. In addition, a digest can be used publicly without mentioning the contents of the document and the source of the document.

MD2, MD4, and MD5 algorithms that **Rivest** developed are **message-digest algorithms** that are used in digital signature applications, where the document is compressed securely before being

signed with the private key. The algorithms mentioned here can be of variable length but with the resultant message digest of 128-bit.

The structures of all three algorithms appear to be similar, though the design of MD2 is reasonably different from MD4 and MD5. MD2 was designed for the 8-bit machines, whereas the MD4 and MD5 were designed for the 32-bit machines. The message is added with extra bits to make sure that the length of the bits is divisible by 512. A **64-bit binary message** is added to the message.

Development of attacks on versions of MD4 has progressed rapidly and **Dobbertin** showed how collisions for the full version of MD4 could be found in under a minute on a typical PC. MD5 is relatively secure but is slower than MD4. This algorithm has four different rounds, which are designed with slight differences than that of MD4, but both the message-digest size and padding requirements remain the same.



Brute Force of MD5

The effectiveness of the hash function can be defined by checking the output produced when an arbitrary input message is randomized. There are two types of **brute-force attacks** for one-way hash function: Normal brute force and birthday attack.

Examples of a few message digests are:

- ⌚ echo "There is CHF1500 in the blue bo" | md5sum
e41a323bdf20eadaf3f0e4f72055d36
- ⌚ echo "There is CHF1500 in the blue box" | md5sum
7a0da864a41fd0200ae0ae97af3279d
- ⌚ echo "There is CHF1500 in the blue box." | md5sum
2db1ff7a70245309e9f2165c6c34999d
- ⌚ echo "There is CHF1500 in the blue box." | md5sum
86c524497a99824897ccf2cd74ede50f

The same text always produces the same MD5 code.

File	Size	CRC	MD5
batch_rename.png	14 472	18528C04	EAF2C712F6E537AE1FEFD3FA1A4F4AAB
change_attributes.html	8 574	58101E09	E18D9F81CCF9A300F79321E8C768E021
change_attributes.png	7 957	2531FC3E	5E8A8FB259C7FDF790E5597C8154AF38
change_case.html	8 756	FC41186B	DDCAD7CF08BF7897D5B8B5F9806B47FD
change_case.png	6 821	2D34D339	04FED507091F5F095D977B358EC20EED
checksum_verify.png	8 117	3D8D9801	AC8AFE99B76BD1022AC7B2E34A7E1C49
convert.html	9 269	BE535A89	902BA23D7CC95EA2999CDA2EF1B27B41
convert.png	7 080	D760CF6	F1176C7967E1DA2CA743D26DE9F180C0
convert_menu.png	8 735	638F8F0F	3F1BB5E0B0B9E86970EDBA9705F14D4
file_comparator.html	8 575	44ED5DC4	959961C3E7D7559C9EE77965302A6E0A
file_comparator.png	17 787	D16F0E2B	C1AE1516BEABC17EDEFB58212D2C5331

FIGURE 19.6: Checksum verifier

Secure Hashing Algorithm (SHA)

C|EH
Certified Ethical Hacker

It is an algorithm for generating cryptographically secure one-way hash, published by the **National Institute of Standards and Technology** as a **U.S. Federal Information Processing Standard**

SHA1

- It produces a 160-bit digest from a message with a maximum length of $(2^{64} - 1)$ bits, and resembles the MD5 algorithm

SHA2

- It is a family of two similar hash functions, with different block sizes, namely SHA-256 that uses 32-bit words and SHA-512 that uses 64-bit words

SHA3

- SHA-3 uses the sponge construction in which message blocks are XORed into the initial bits of the state, which is then invertibly permuted

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Secure Hashing Algorithm (SHA)

The Secure Hash Algorithm (SHA), specified in the **Secure Hash Standard (SHS)**, was developed by NIST, and published as a federal information-processing standard (FIPS PUB 180). It is an algorithm for generating a cryptographically secure one-way hash. SHA is part of the Capstone Project. Capstone is the U.S. government's long-term project to develop a set of standards for publicly available cryptography, as authorized by the Computer Security Act of 1987. The basic organizations that are responsible for Capstone are NIST and the NSA. SHA is similar to the MD4 message-digest algorithm family of hash functions, which was developed by Rivest.

The algorithm accepts a message of **264 bits** in length and a **160-bit** message output digest is produced, that is designed to complicate the searching of the text, which is similar to the given hash. The algorithm is slightly slower than MD5, but the larger message digest makes it more secure against brute-force collision and inversion attacks.

The following are the cryptographic hash functions designed by the **National Security Agency (NSA)**:



SHA1

SHA1 produces a 160-bit digest from a message with a maximum length of $(2^{64} - 1)$ bits, and resembles the MD5 algorithm.



SHA2

SHA2 is a family of two similar hash functions, with different block sizes, namely SHA-256 that uses 32-bit words and SHA-512 that uses 64-bit words.



SHA3

SHA3 is a future hash function standard still in development, chosen in a public review process from non-government designers.

Comparison of SHA functions (SHA0, SHA1 & SHA2)

Algorithm and variant	Output size (bits)	Internal hash sum (bits)	Size of block (bits)	Maximum size of message (bits)	Size of word (bits)	Rounds	Operations	Collision found
SHA-0	160	160	512	$2^{64}-1$	32	80	+ , and , , xor, rot	or Yes
SHA-1	160	160	512	$2^{64}-1$	32	80	+ , and , xor, rot	or, Theoretical attacks (2^{51})
SHA-2	SHA-256/224	256/224	256	512	$2^{64}-1$	32	+ , and , or , xor, shr, rot	None
	SHA-512/384	512/384	512	1024	$2^{128}-1$	128	+ , and , or , xor, shr, rot	None

TABLE 19.1: Comparison between SHA-0, SHA-1 & SHA-2 functions



What Is SSH (Secure Shell)?

Secure Shell is a program that is used to log onto another computer over the network, to transfer files from one computer to another. It offers good authentication and a secure communication channel over insecure media. It might be used as a replacement for telnet, login, rsh, and rcp. In SSH2, **sftp** is a replacement for **ftp**. In addition, SSH offers secure connections and secure transferring of TCP connections. SSH1 and SSH2 are completely different protocols. SSH1 encrypts the user's server and hosts keys to authenticate where SSH2 only uses host keys, which are different packets of keys. SSH2 is more secure than SSH1. It should be noted that the **SSH1** and **SSH2** protocols are in fact different and not compatible with each other. SSH2 is more secure and has an improved performance than SSH1 and is also more portable than SSH1.

The SSH1 protocol is not being developed anymore, as SSH2 is the standard. Some of the main features of SSH1 are as follows:

- ⌚ SSH1 is more vulnerable to attacks due to the presence of structural weaknesses
- ⌚ It is an issue of the man-in-the-middle attack
- ⌚ It is supported by many platforms
- ⌚ It supports hosts authentication

- ➲ It supports varied authentication
- ➲ Performance of SSH2 is better than SSH1

SSH communications security maintains SSH1 and SSH2 protocols.

It authenticates with the help of one or more of the following:

- ➲ Password (the /etc/passwd or /etc/shadow in UNIX)
- ➲ User public-key (RSA or DSA, depending on the release)
- ➲ Kerberos (for SSH1)
- ➲ Host-based (.rhosts or /etc/hosts. equiv in SSH1 or public key in SSH2)

Secure Shell protects against:

- ➲ A remote host sending out packets that pretend to come from another trusted host (IP spoofing). SSH protects against a spooper on the local network, who can pretend to be the user's router to the outside.
- ➲ A host pretending that an IP packet comes from another **trusted host (IP source routing)**.
- ➲ An attacker forging domain name server records (DNS spoofing).
- ➲ Capturing of passwords and other data by the intermediate hosts.
- ➲ Exploitation of data by the people who control the intermediate hosts.
- ➲ Attacking by listening to X authentication data and spoofing connections to the X11 server.



FIGURE 19.7: Secure shell tunneling

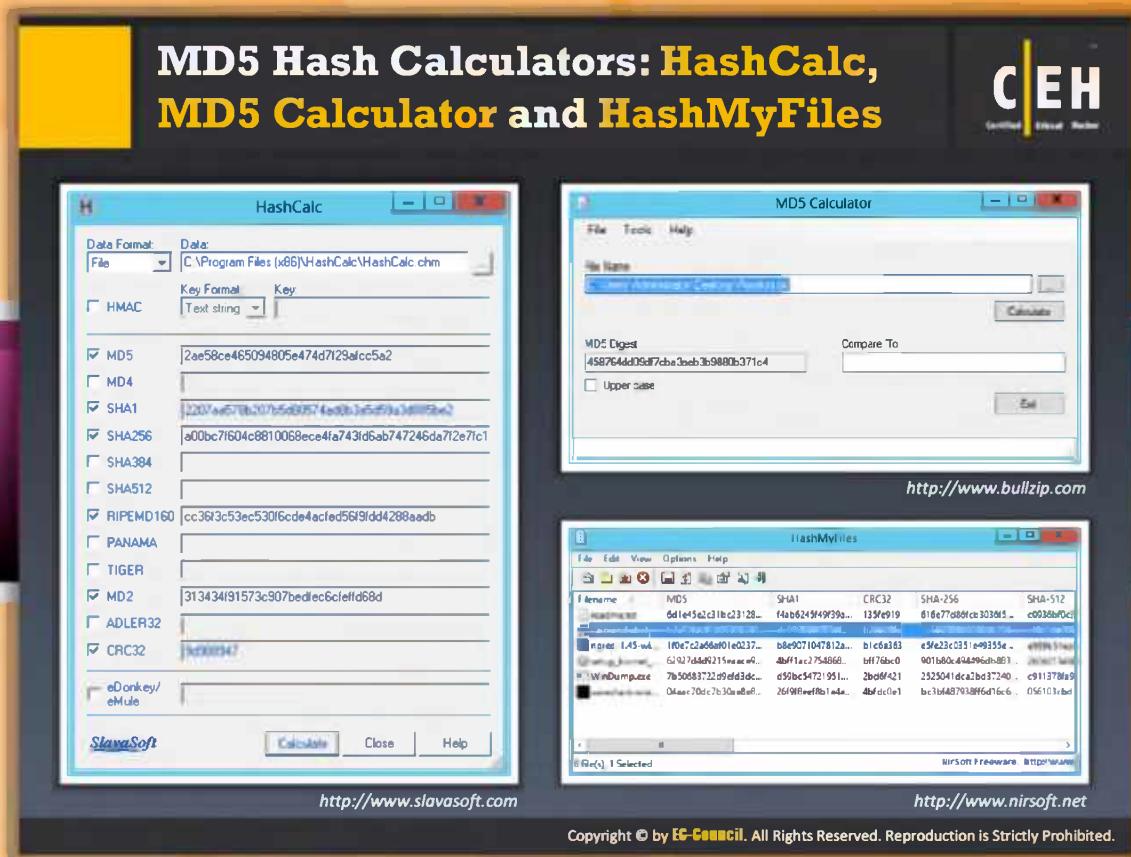


Module Flow

So far, we have discussed cryptography concepts and various encryption algorithms. Now it is time to discuss how cryptography is usually performed. There are many cryptographic tools readily available in the market that can help you to secure your data..

Cryptography Concepts	Encryption Algorithms
Cryptography Tools	Public Key Infrastructure (PKI)
Email Encryption	Disk Encryption
Cryptography Attacks	Cryptanalysis Tools

This section lists and describes various cryptographic tools.



MD5 Hash Calculators: HashCalc, MD5 Calculator, and HashMyFiles

Hashing is one form of cryptography in which a message digest function is used to convert plaintext into its equivalent hash value. This message digest function uses different hash algorithms to **convert plaintext** into hash values. Many MD5 hash calculators are readily available in the market. Examples of **MD5** hash calculators include:



HashCalc

Source: <http://www.slavasoft.com>

The HashCalc utility allows you to compute message digests, checksums, and HMACs for files, as well as for text and hex strings. It allows you to calculate hash values using different types of hashing algorithms such as **MD2, MD4, MD5, SHA-1, SHA-2 (256, 384, 512), RIPEMD-160, PANAMA, TIGER, ADLER32, and CRC32**. You just need to select the file and hash algorithm for calculating the hash value of a particular file.

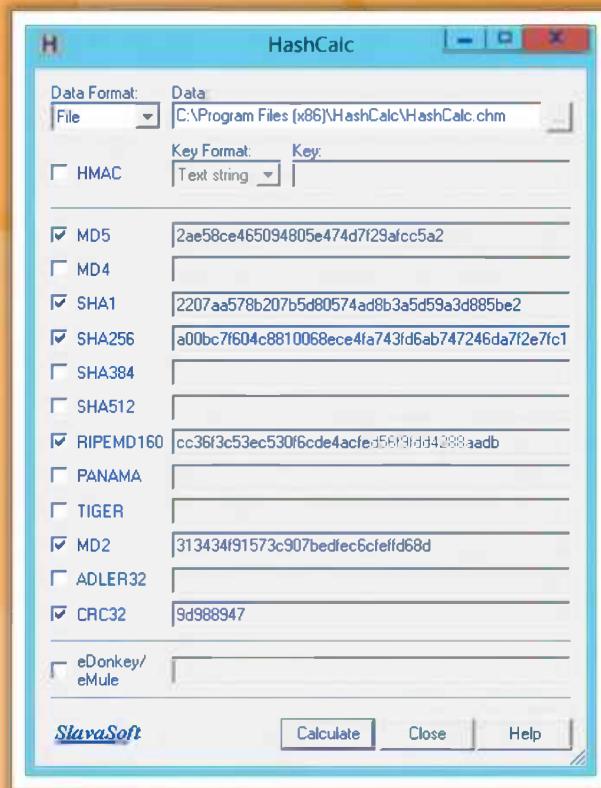


FIGURE 19.8: HashCalc screenshot

MD5 Calculator

Source: <http://www.bullzip.com>

MD5 Calculator allows you to calculate the MD5 hash value of the selected file. The **MD5 Digest field** of the utility contains the calculated hash value. You just need to select a file of which the hash value needs to be calculated. You can also compare two hash values with this **tool**.

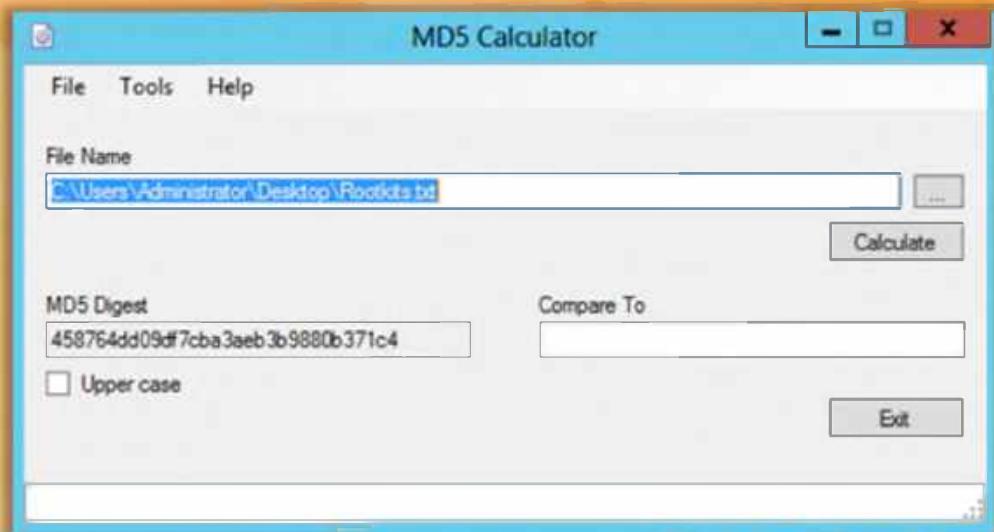


FIGURE 19.9: MD5 Calculator calculating MD5 hash value



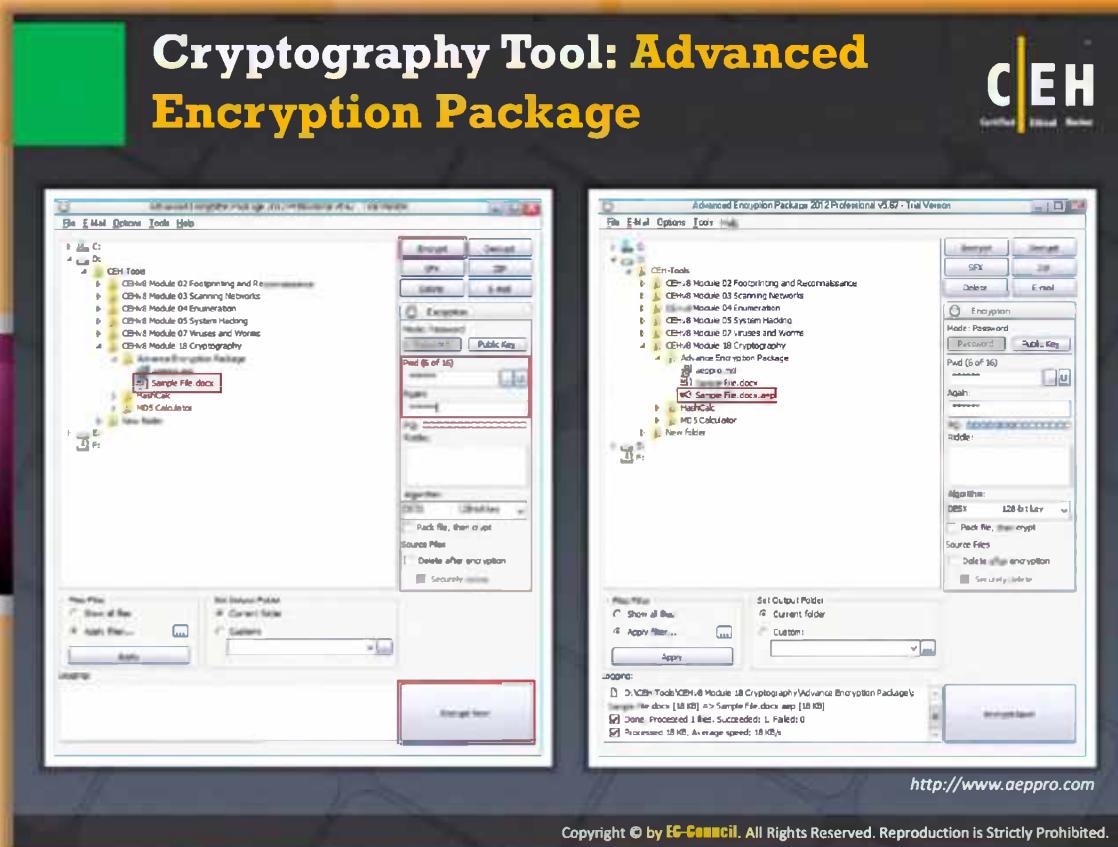
HashMyFiles

Source: <http://www.nirsoft.net>

The HashMyFiles utility allows you to calculate the MD5 and SHA1 hashes of one or more files. You can copy the **MD5/SHA1** hashes list into the clipboard, or save it into a text/html/xml file. It can also be launched from the context menu of Windows Explorer, and display the MD5/SHA1 hashes of the selected file or folder.

Filename	/	MD5	SHA1	CRC32	SHA-256	SHA-512
readme.txt		6d1e45e2c31bc23128...	f4ab6245f49f39a...	135fe919	616e77d88fc3036f5...	c0936bf0c3...
cain and abel_...	b2a72fadf1d0550b743...	de8908a9f285ef...	b2eed8fa	ce5ed388b8388dc254...	cf8c1de709	
ngrep-1.45-wi...	1f0e7c2a66af01e0237...	b8e9071047812a...	b1c6a363	e5fe23c0351e49355e...	e989b51ea1...	
setup_kismet_...	62927d4d9215eaace9...	4bff1ac2754868...	bff76bc0	901b80c494496db883...	2656013468	
WinDump.exe	7b50683722d9efd3dc...	d59bc54721951...	2bd6f421	2525041dca2bd37240...	c911378fa9...	
wireshark-win...	04aac70dc7b30ae8e8...	26f9f8ee8b1a4a...	4bfdc0e1	bc3bf487938ff6d16c6...	056103cbd1...	

FIGURE 19.10: HashMyFiles screenshot



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Cryptography Tool: Advanced Encryption Package

Source: <http://www.aepro.com>

Advanced Encryption Package is file encryption software that helps you maintain the privacy of your information by allowing you to password-protect files. It is able to perform encryption, decryption, and self-decrypting file creation, file Delete/Wipe, Zip management, encryption key management, and file emailing.

Its feature includes:

- ④ **Strong and proven algorithms** are used to protect your sensitive documents
- ④ It can encrypt files as well as text
- ④ Performs secure file deletion
- ④ Ability to create **encrypted self-extracting file** to send it as **email** attachment

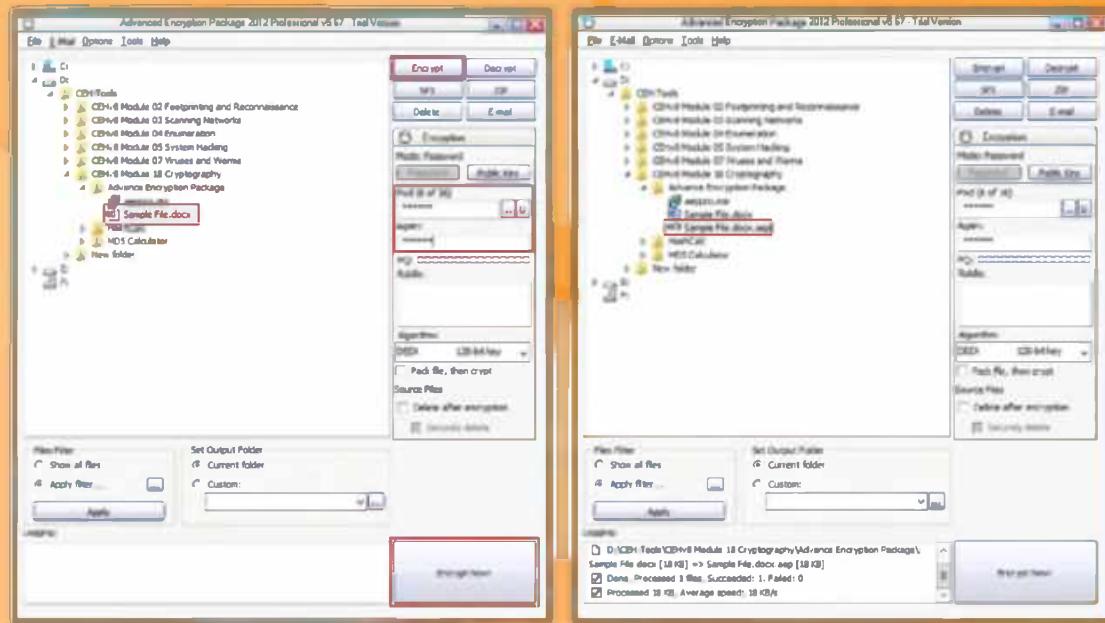


FIGURE 19.11: Advanced Encryption Package protecting files using passwords



Cryptography Tool: BCTextEncoder

Source: <http://www.jetico.com>

BCTTextEncoder allows you to **encrypt** and **decrypt** the **confidential messages** for secure email or chat communications. It uses public key encryption methods as well as password-based encryption and strong and approved symmetric and public key algorithms for **data encryption**. You simply need to choose the text you want to encrypt and specify the password and then click the button to encode it.

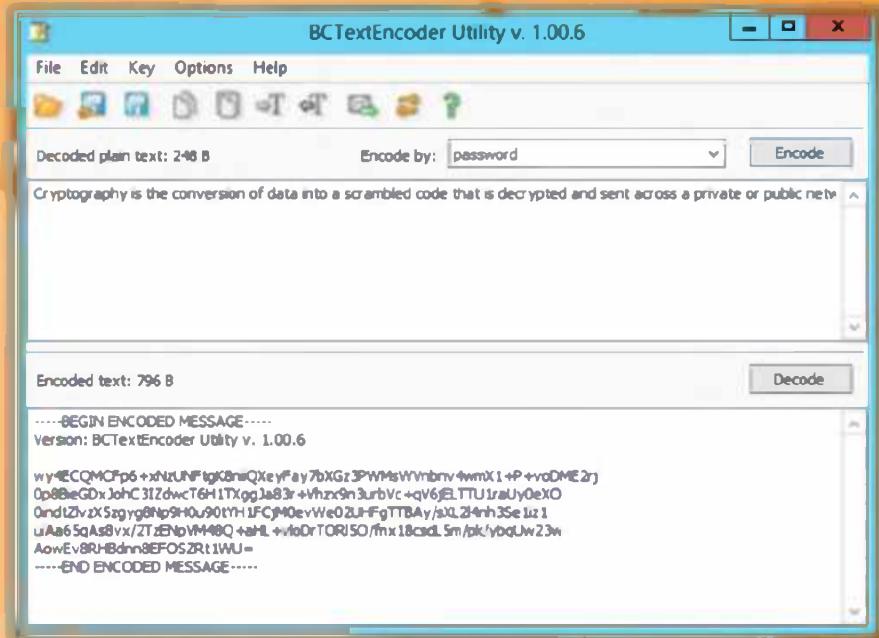


FIGURE 19.12: BCTextEncoder encrypting and decrypting confidential messages

Cryptography Tools

 CommuniCrypt File Encryption Tools http://www.communicrypt.com	 NCrypt XL http://www.littleelite.net
 Steganos LockNote https://www.steganos.com	 ccrypt http://ccrypt.sourceforge.net
 AxCrypt http://www.axantum.com	 WinAES http://fatlyz.com
 AutoKrypt http://www.hiteksoftware.com	 EncryptOnClick http://www.2brightsparks.com
 CryptoForge http://www.cryptoforge.com	 GNU Privacy Guard http://www.gnupg.org

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Cryptography Tools

There are various cryptographic tools that you can use for encrypting and decrypting your information, files, etc. These tools implement different types of available **encryption algorithms**:

- ⌚ CommuniCrypt File Encryption Tools available at <http://www.communicrypt.com>
- ⌚ Steganos LockNote available at <https://www.steganos.com>
- ⌚ AxCrypt available at <http://www.axantum.com>
- ⌚ AutoKrypt available at <http://www.hiteksoftware.com>
- ⌚ CryptoForge available at <http://www.cryptoforge.com>
- ⌚ NCrypt XL available at <http://www.littleelite.net>
- ⌚ Ccrypt available at <http://ccrypt.sourceforge.net>
- ⌚ WinAES available at <http://fatlyz.com>
- ⌚ EncryptOnClick available at <http://www.2brightsparks.com>
- ⌚ GNU Privacy Guard available at <http://www.gnupg.org>



Module Flow

So far, we have discussed cryptography, various **encryption algorithms**, and the use of encryption algorithms in cryptography. In addition to the cryptographic security mechanisms discussed so far, there is one more infrastructure intended to **exchange data** and **money** over the Internet securely: PKI (Public Key Infrastructure).

Cryptography Concepts	Encryption Algorithms
Cryptography Tools	Public Key Infrastructure (PKI)
Email Encryption	Disk Encryption
Cryptography Attacks	Cryptanalysis Tools

This section provides information about Public Key Infrastructure (PKI) and the role of each components of **PKI** in the **security public** key encryption. Let's start with what is Public Key Infrastructure (PKI)?

Public Key Infrastructure (PKI)

Public Key Infrastructure (PKI) is a **set of hardware, software, people, policies, and procedures** required to create, manage, distribute, use, store, and revoke **digital certificates**.

Certificate Management System
Generates, distributes, stores, and verifies certificates

Digital Certificates
Establishes credentials of a person when doing online transactions

End User
Requests, manages, and uses certificates

Certificate Authority (CA)
Issues and verifies digital certificates

Registration Authority (RA)
Acts as the verifier for the certificate authority

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

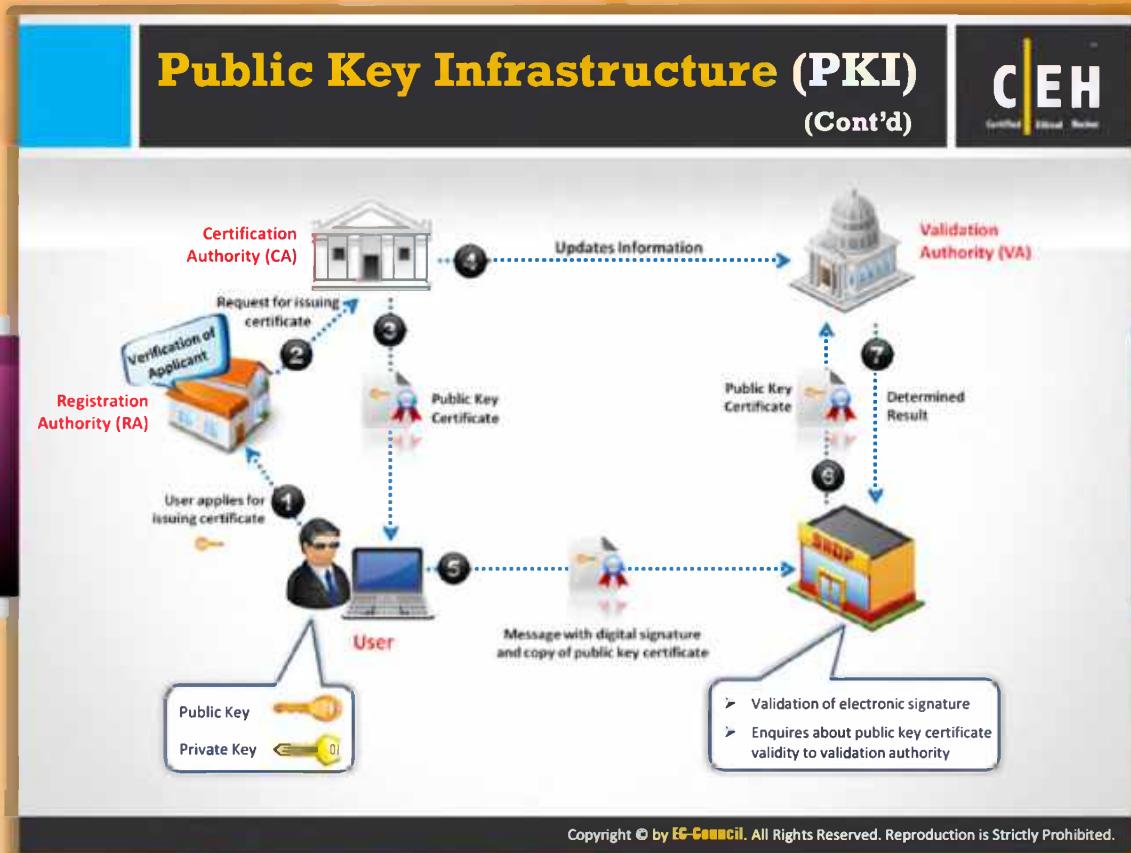


Public Key Infrastructure (PKI)

Public Key Infrastructure (PKI) is a **security architecture** developed to increase the confidentiality of information being exchanged over the insecure Internet. It includes hardware, software, people, policies, and procedures required to create, manage, distribute, use, store, and revoke digital certificates. In **cryptography**, the PKI helps to bind public keys with corresponding user identities by means of a certificate authority (CA). The following are the components of PKI:

- ➊ A certificate authority (CA) that issues and verifies **digital certificates**
- ➋ A certificate management system for generation, distribution, storage, and verification of certificates
- ➌ One or more directories where the certificates (with their public keys) are held
- ➍ A registration authority (RA) that acts as the verifier for the **certificate authority**

Cryptographic keys can be delivered securely between users by PKI.



Public Key Infrastructure (PKI) (Cont'd)

The public key cryptosystem uses a pair of a public key and a private key to assure secure communication over the Internet. In public key cryptosystem authentication, it is important to connect the correct person and the public key. This is accomplished with the help of Public Key Infrastructure (PKI). **Asymmetric** (public key) **cryptography** is the foundation technology of PKI, when sender and receiver agreed upon a secret communication using public key encryption with a digital signature.

The figure that follows shows how a message gets digitally signed by the organization involved in authentication and certification by means of PKI. In public key cryptosystems, the correspondence between a public key and the private key is taken care by the certification authority (CA), i.e., based on the public key the **CA determines the owner of the respective private key**. Initially, the user requests the certification authority for binding his or her public key; a certification authority digitally signs it and issues a public key certificate to the user. It binds the user's identity with the user's public key. In between the user and the CA, there exists an organization, the Registration Authority (RA). The job of the RA is to verify the identity of the user requesting the certificate face-to-face. There exists another authority in PKI, i.e., the validation authority (VA). The job of the VA is to check whether the certificate was issued by **trustworthy** a CA or not, i.e., is it valid or not. The sender and receiver can then exchange a secret message using public key cryptography.

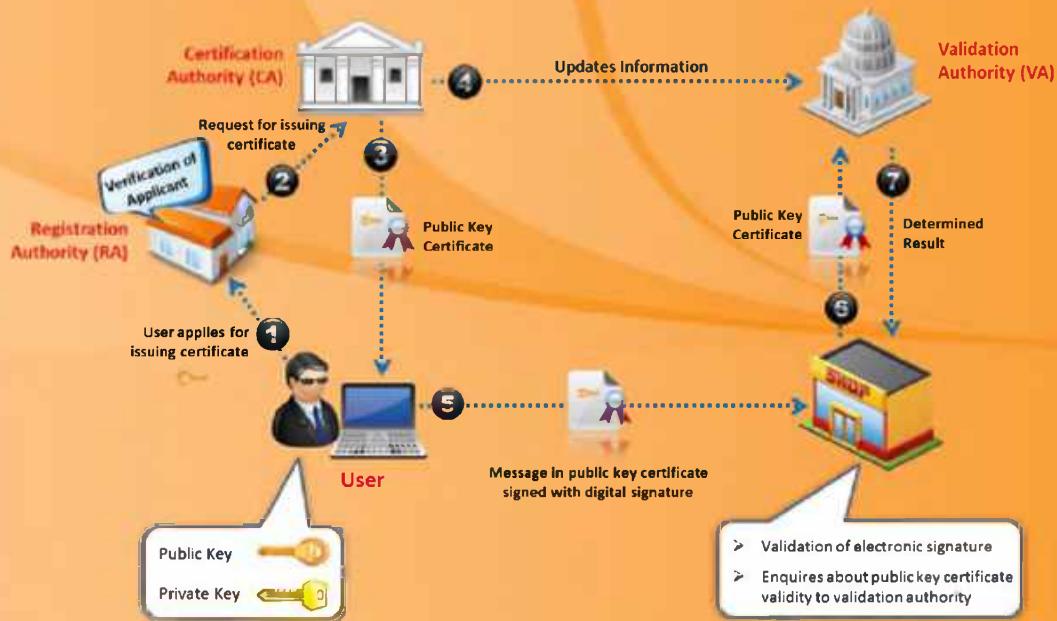


FIGURE 19.13: Public Key Infrastructure (PKI)

Certification Authorities

The image displays four screenshots of certification authority websites arranged in a 2x2 grid:

- Comodo**: The first screenshot shows the Comodo website with a banner stating "The First To Bring You a Full Line of 2048-bit Certificates". The URL <http://www.comodo.com> is listed below.
- Thawte**: The second screenshot shows the Thawte website with a banner stating "online security trusted by millions around the world". The URL <http://www.thawte.com> is listed below.
- Norton**: The third screenshot shows the Norton website with a banner stating "Same check. New name. Still the gold standard.". The URL <http://www.verisign.com> is listed below.
- Entrust**: The fourth screenshot shows the Entrust website with a banner stating "SECURITY ON:SSL". The URL <http://www.entrust.net> is listed below.

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Certification Authorities

Certification authorities are the entities that issue digital certificates. The following are some of the **certificate authorities**:



Comodo

Source: <http://www.comodo.com>

Comodo offers a complete range of PKI digital certificates with strong SSL encryption available. It ensures standards of confidentiality, system reliability, and pertinent business practices as judged through qualified independent audits. The PKI (Public Key Infrastructure) management solutions offered by Comodo include **Comodo Certificate Manager** and **Comodo EPKI Manager**.

Available Digital Certificates:

- ⊕ Extended validation (EV)-SSL
- ⊕ Multi-domain EV SSL
- ⊕ Wildcard SSL
- ⊕ Unified communications (UC)
- ⊕ Intel Pro Series

- ⌚ General purpose SSL
- ⌚ Secure Email - S/MIME
- ⌚ Client authentication
- ⌚ Code signing

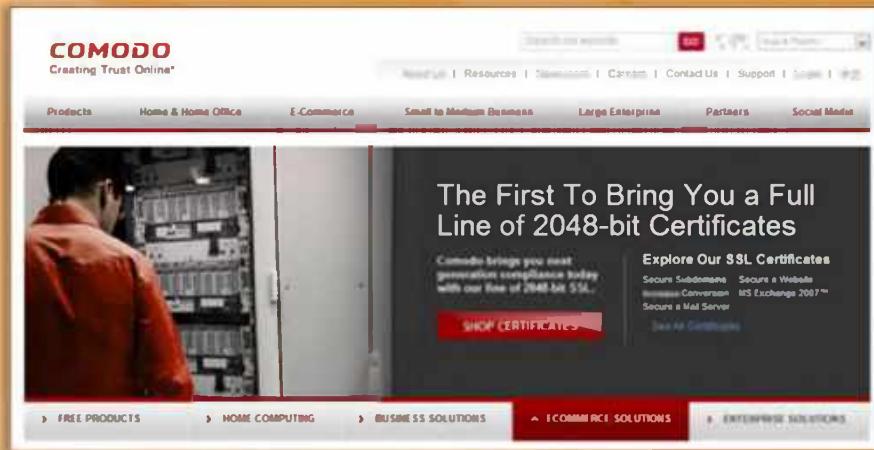


FIGURE 19.14: Comodo screenshot



thwate

Source: <http://www.thawte.com>

thawte is a Certification Authority. thwate offers SSL and code signing digital certificates to secure servers, provides data encryption, authenticates users, protects privacy, and assures online identities through stringent authentication and verification processes. The **SSL certificates** offered by thwate include Wildcard SSL Certificates, SAN /UC Certificates, SGC SuperCerts, and Extended Validation SSL Certificates.



FIGURE 19.15: thawte screenshot



Verisign

Source: <http://www.verisign.com>

VeriSign Authentication Services, now part of Symantec Corp. (NASDAQ: SYMC), provides solutions that allow companies and consumers to **engage in communications** and **commerce** online with confidence.

SSL Certificates:

- ⌚ Secure Site Pro with EV
- ⌚ Secure Site with EV
- ⌚ Secure Site Pro
- ⌚ Secure Site
- ⌚ Managed PKI for SSL
- ⌚ SSL for the Enterprise
- ⌚ SSL Partner Programs
- ⌚ Symantec Certificate Intelligence Center

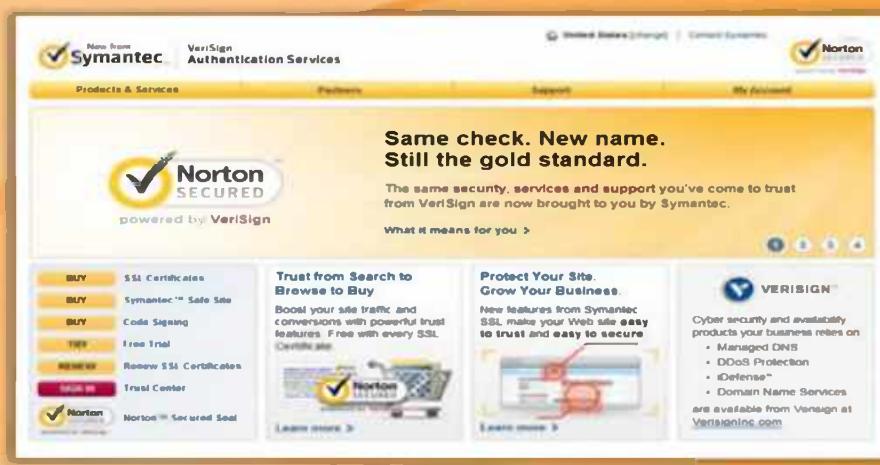


FIGURE 19.16: Verisign screenshot

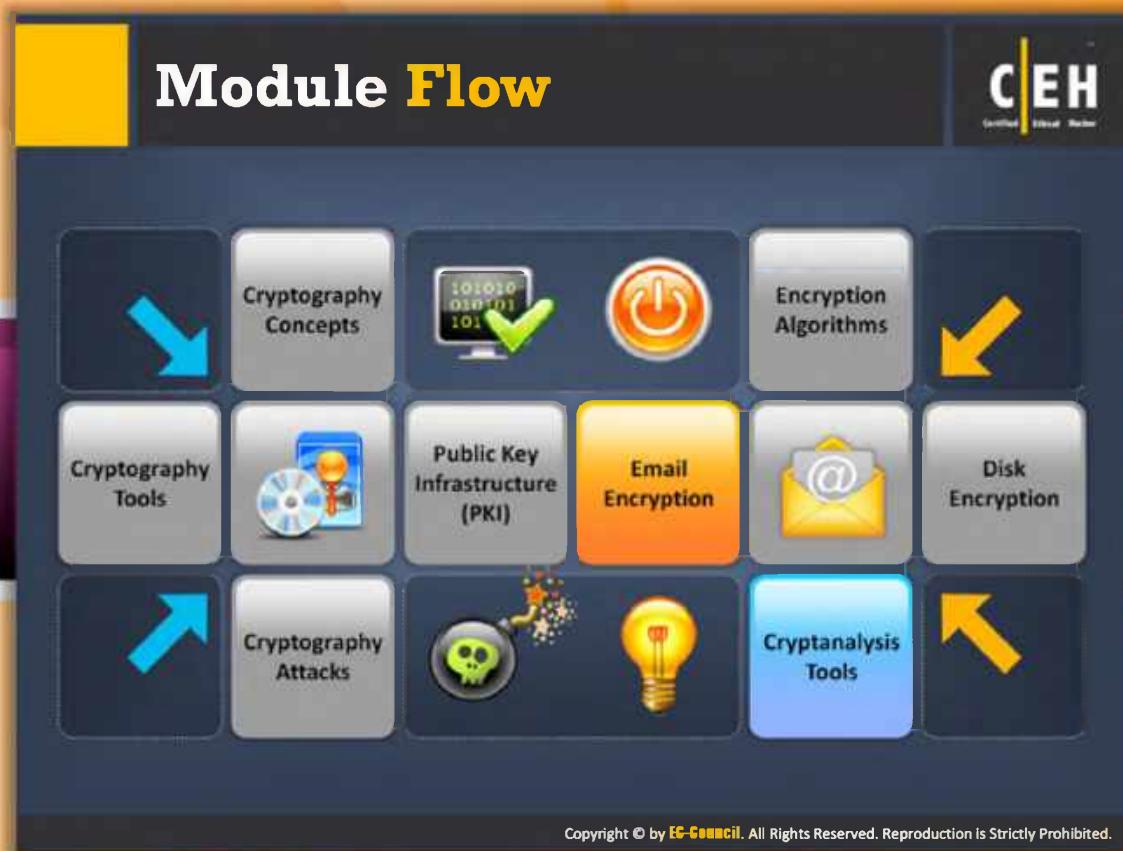


Source: <http://www.entrust.net>

Entrust provides identity-based security solutions that empower enterprises, consumers, citizens, and the web. Entrust's solutions include strong authentication, fraud detection, digital certificates, SSL, and PKI. Entrust can deploy appropriate security solutions to help protect digital identities and information at multiple points to address **ever-evolving threats**.



FIGURE 19.17: Entrust screenshot

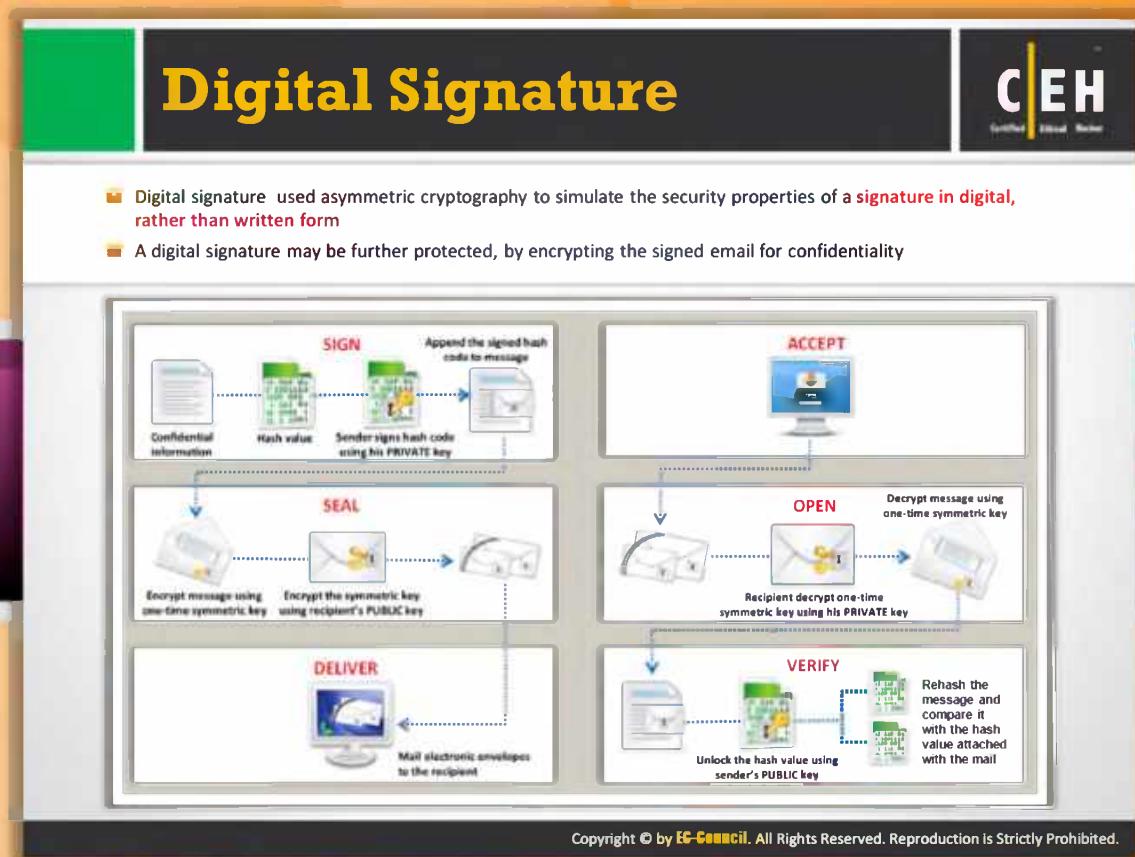


Module Flow

At present, most businesses use email as the major source of communication as it is simple and easy to communicate or share information. These emails may contain **sensitive information** about their projects, updates, etc. If this information falls into the wrong hands, then the organizations may face **huge losses**. This risk can be avoided by encrypting the email messages. Email encryption is the means to transfer the plaintext message into an unreadable form.

 Cryptography Concepts	 Encryption Algorithms
 Cryptography Tools	 Public Key Infrastructure (PKI)
 Email Encryption	 Disk Encryption
 Cryptography Attacks	 Cryptanalysis Tools

This section focuses on various email security mechanisms such as **digital signatures, SSL, and TLS**.



Digital Signature

A digital signature is a cryptographic means of authentication. Public key cryptography, which uses an **asymmetric key algorithm**, is used for creating the digital signature. The two types of keys in public key cryptography are the private key (which is known only to the signer and used to create the digital signature) and the public key (which is more widely known and is used by a relying party to verify the digital signature). A **hash function** is a process, or an algorithm, that is used in creating and verifying a digital signature. This algorithm creates a digital representation of a message, which is also known as a "**fingerprint**." This fingerprint is of a "hash value" of a standard length, which is much smaller than the message, but is unique to it. If any change is made to the message, it will automatically produce a different hash result; it is not possible to derive the original message from the hash value in case of a secure hash function, which is also known as a **one-way hash function**.

The hash result of the original message and the hash function that is used to create the digital signature are required to verify the digital signature. With the help of the public key and the new result, the verifier checks:

- ⊕ If the digital signature is created with the **related private key**. If the new hash result is the same as the original hash result, which was converted into a digital signature during the **signing process**.

To correlate the key pair with the respective signer, the certification authority presents a certificate that is an electronic record of the public as the subject of the certificate, and confirms the identity of the signer as the related private key owner. The future signer is called the subscriber. The main function of a certificate is to bind a pair of public and private keys to a particular subscriber. The recipient of the certificate relies on a digital signature created by the subscriber named in the certificate. The public key listed can be used to verify that the private key is used to create the related digital signature.

The certification authority digitally signs the certificate to assure the authenticity of both the public key and the subscriber's identity. The authority's digital signature on the certificate can be verified with the help of the public key of the certification authority recorded in another certificate, which belongs to another **certification's authority**. This certificate can be authenticated with the help of another public key recorded in another certificate and so on.

The repository can be made to publish the certificate; the public key and its identity are available for verification of the certificate. The retrieval and verification of the digital signature is made with the help of an online database called repositories, which holds the certificates and other information. The certification authority may suspend or revoke the certificate.

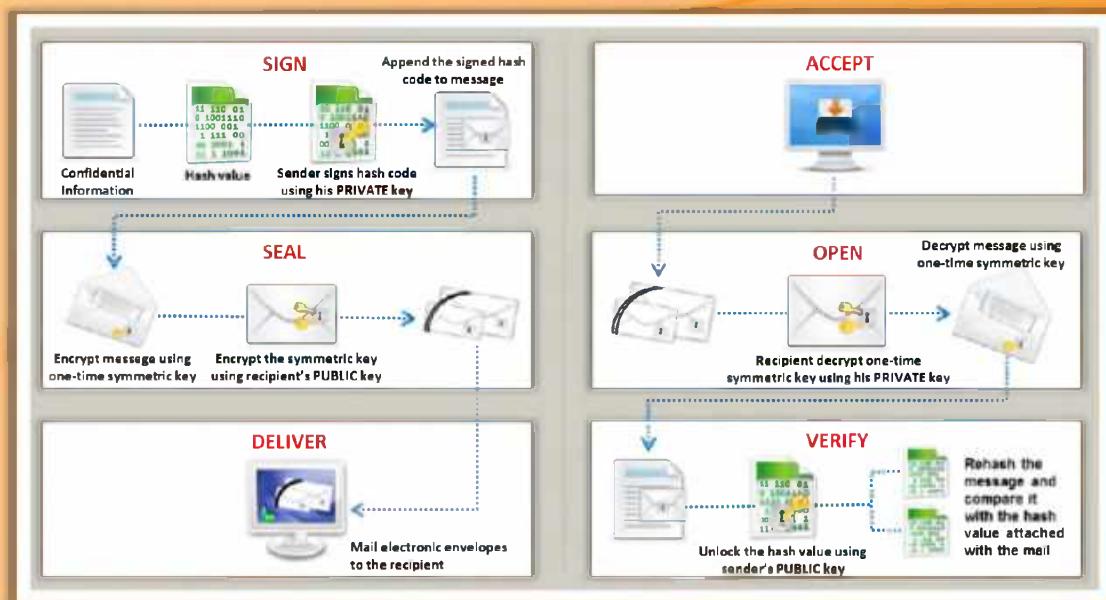
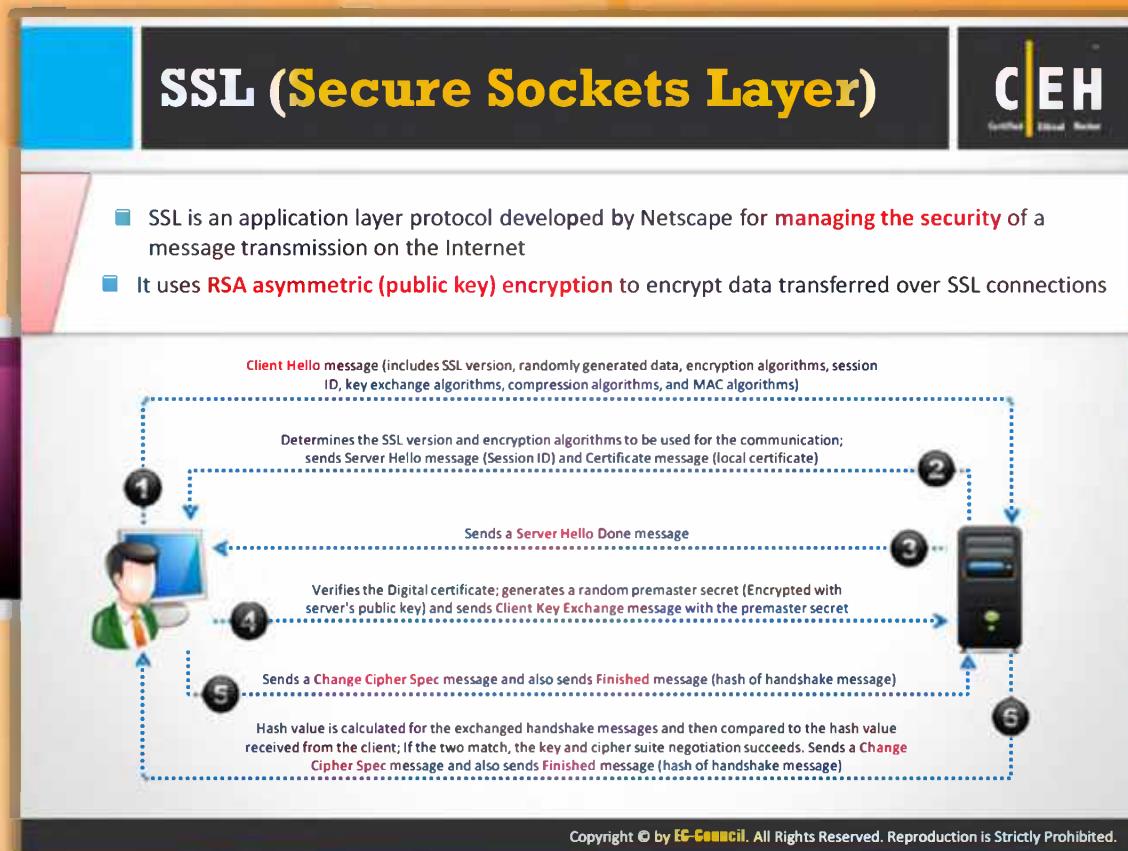


FIGURE 19.18: Digital signatures



SSL (Secure Sockets Layer)

SSL is **acronym** for Secured Sockets Layer, developed by Netscape. It is a protocol for sending private documents over the Internet. It works with the help of the private key to encrypt data that is transferred over an SSL connection. The main motive behind designing the SSL protocol is to provide privacy between two communicating applications, such as a client and a server. Moreover, the protocol is designed to authenticate the server and the client; SSL requires a **reliable transport protocol** such as **TCP** for **data transmission** and **reception**.

Any application-layer protocol that is higher than **SSL**, such as **HTTP**, **FTP**, and **TELNET**, can be layered on top of SSL transparently. The SSL acts as an arbitrator between the encryption algorithm and session key, and also verifies the destination server before the transmission and reception of data. The complete data of the application protocol is encrypted, to ensure security. It also offers channel security which has three basic properties:

- It has a private channel, where the messages are encrypted after the simple handshake that defines the secret key.
- The channel is authenticated. The server endpoints are always authenticated but the client endpoints are optionally **authenticated**.
- The channel is reliable. The transmission has an integrity check.

An SSL session is responsible for the **SSL handshake protocol** to organize the states of the server and clients, thus ensuring the consistency of the protocol state machines (the states are not exactly parallel).

There are two different types of states: operating and pending. In addition to the two states, two additional states are also maintained; the read and write states. When the server or client obtains the cipher spec message, the message is copied into a current read state from the pending read state. In a similar way, when the data is transmitted from the server or client, it transmits a changed cipher spec message, and copies the message into the write current state from the pending write state. After the completion of the **handshake arbitration**, the server and client exchange the changed spec message and the communication is based on the newly agreed upon cipher spec. An SSL may include many secure connections, and it might have multiple concurrent sessions. The elements included in **session** state are as follows:



Session Identifier

Session identifier is a random sequence of bytes transmitted by the server to identify an active or **presumable session** state:

- ⌚ Peer Certificate – X509.v3[X509] is the certificate of the peer and may be null.
- ⌚ Compression Method – Is the algorithm used to compress data prior to encryption.
- ⌚ Cipher Spec – Enumerates the **bulk data encryption** and **MAC algorithms**. It also defines cryptographic attributes like the size of the hash.
- ⌚ Master Secret – Is the **48-byte** secret shared between the client and server.
- ⌚ Is Resumable – A flag specifies whether a new session can be started.

The elements of the connection state are as follows:

- ⌚ Server and client random – Is the sequences of bytes, which are selected by the server and the client for every connection.
- ⌚ Server write MAC secret – Is the secret used in MAC operations on data written by the server.
- ⌚ Client write MAC secret – Is the secret used in MAC operations on data written by the client.
- ⌚ Server write key – Is the huge cipher key for data encrypted by the server and decrypted by the client.
- ⌚ Client write key – Is the cipher key for data encrypted by the client and decrypted by the server.
- ⌚ Initialization vectors – In CBC (Cipher Block Chain) mode when the block cipher is used, an initialization vector is managed for every key. It is started by the SSL handshake protocol and is used to make the first cipher text. The last **cipher text block** of every text is used with the subsequent record.

- ④ Sequence numbers – Every party maintains a different and unique sequence of numbers for the transmission and reception of messages for every connection. The appropriate sequence is set to **zero** depending on the party that sends and receives cipher spec.



SSL Handshake Protocol Flow

The SSL handshake protocol works on top of the SSL record layer. These processes that are executed in the three handshake protocol are summarized as follows:

- ④ The client sends a **hello message** to the server and the server must respond to the hello message with a hello message, or else the connection will fail due to the **occurrence of a fatal error**. The attributes that are established due to the server and client hello are: protocol version, session ID, cipher suite, and compression method.
- ④ After the connection is established, the server sends a certificate to the client for authentication. In addition, a **server-key exchange message** might be sent. If the server is authenticated, the client may be requested for the certificate, if that is appropriate to the **cipher suite** selected.
- ④ The server sends a hello done message, to inform that the handshake phase is complete and waits for the client's response.
- ④ If the client receives a certificate request message, the client must respond to the message by sending a certificate message or “no certificate” alert. The client-key exchange message is sent and the content of the message depends on the public-key algorithm between the server hello and client hello. If the certificate sent by the client has signing ability, a **digitally signed certificate** verifies the message, and is transmitted.
- ④ The client transmits the changed cipher spec message and copies the pending cipher spec into the current cipher spec. The client sends a message to initiate the completion of the message under the new algorithm, keys, and secrets. In response the server replies by sending its own changed cipher spec message, transfers the pending cipher spec to the current cipher spec, and initiates the completion of the message under the new cipher spec. This is the point of completion of the handshake and the server starts to exchange the application layer data.

The message of the previous session or the **replica** of an existing session is as follows:

The client initiates the communication by sending a hello message with the session I of the session that is to be resumed. The server checks its cache to look for the match of the session ID; if it finds a match it re-establishes the session under the specified session state with same session ID. This is the point where both the server and the client exchange the changed spec messages and proceed directly to the finished messages. After re-establishment, the server and the client exchange the data at the application layer. If the session I is not found, the server creates a **new session ID**, and the SSL client and server carry out a complete handshake.



FIGURE 19.19: Depicting SSL Handshake Protocol Flow

Transport Layer Security (TLS)

CEH Certified Ethical Hacker

- TLS is a protocol to establish a secure connection between a client and a server and ensure privacy and integrity of information during transmission
- It uses the RSA algorithm with 1024 and 2048 bit strengths

TLS Handshake Protocol
It allows the client and server to authenticate each other, select encryption algorithm, and exchange symmetric key prior to data exchange

TLS Record Protocol
It provides secured connections with an encryption method such as Data Encryption Standard (DES)

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Transport Layer Security (TLS)

TLS is a protocol to establish a secure connection between a client and a server and ensure privacy and integrity of information during transmission. It is a cryptographic protocol intended to provide information security over the Internet. The **TLS encrypts** the network connection segments at the application layer for the transport layer. It uses asymmetric cryptography for key exchange, symmetric encryption for confidentiality, and message authentication codes for message integrity. With the help of TLS, you can reduce some of the risks such as tampering, message forgery mail communications, and eavesdropping during transmission of electronic mails or information.

TLS protocol consists of two layers:

- ➊ TLS record protocol
- ➋ TLS handshake protocol



TLS Record Protocol

The TLS record protocol provides secure communications. It is intended for encryption, authentication, and compression (optional) of packets. Once the **handshake**

process is done, then record layer functions can be called at any time whenever there is a need to send or receive data. It is responsible for securing application data and also verifying its integrity and origin of the data. **TLS Record Protocol** manages the following:

- ➊ Dividing and reassembling messages
- ➋ Compressing and decompressing blocks (optional)
- ➌ Applying MAC (Message Authentication Code) and verifying incoming messages based on MAC
- ➍ Encrypting and decrypting messages

The outgoing **encrypted data** from the record protocol is sent to TCP layer for transport.



TLS Handshake Protocol

The TLS handshake protocol is responsible for peers to agree upon **security parameters** for the record layer, authentication. This also negotiates a session consisting of session identifier, peer certificate, compression method, cipher spec, master secret, and information about resuming a connection. The figure that follows shows the process of client-authenticated TLS handshake:

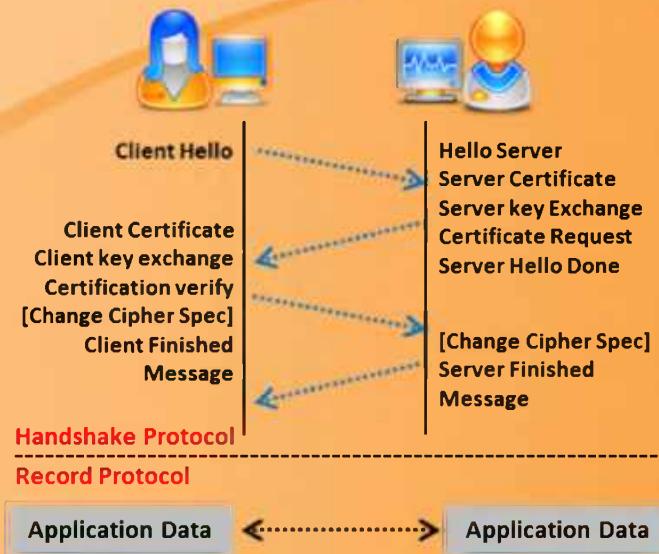


FIGURE 19.20: Showing the client-authenticated TLS handshake process

A handshake protocol exchanges a series of messages between a client and a server for a secure connection. Initially, the client sends a “hello” to the server. The server, in response to the client, sends “hello.” During this period, the security capabilities including protocol version, compression method, cipher suite, session ID, and initial random number have been established. Then the server may send a certificate and key exchange and requests a certificate. Now, the server signals the end of the hello message. In response to the certificate request by the server, the client sends the certificate and key exchange. The client then sends certificate verification. Both the client and server exchange their **cipher suite** and finish the handshake protocol.

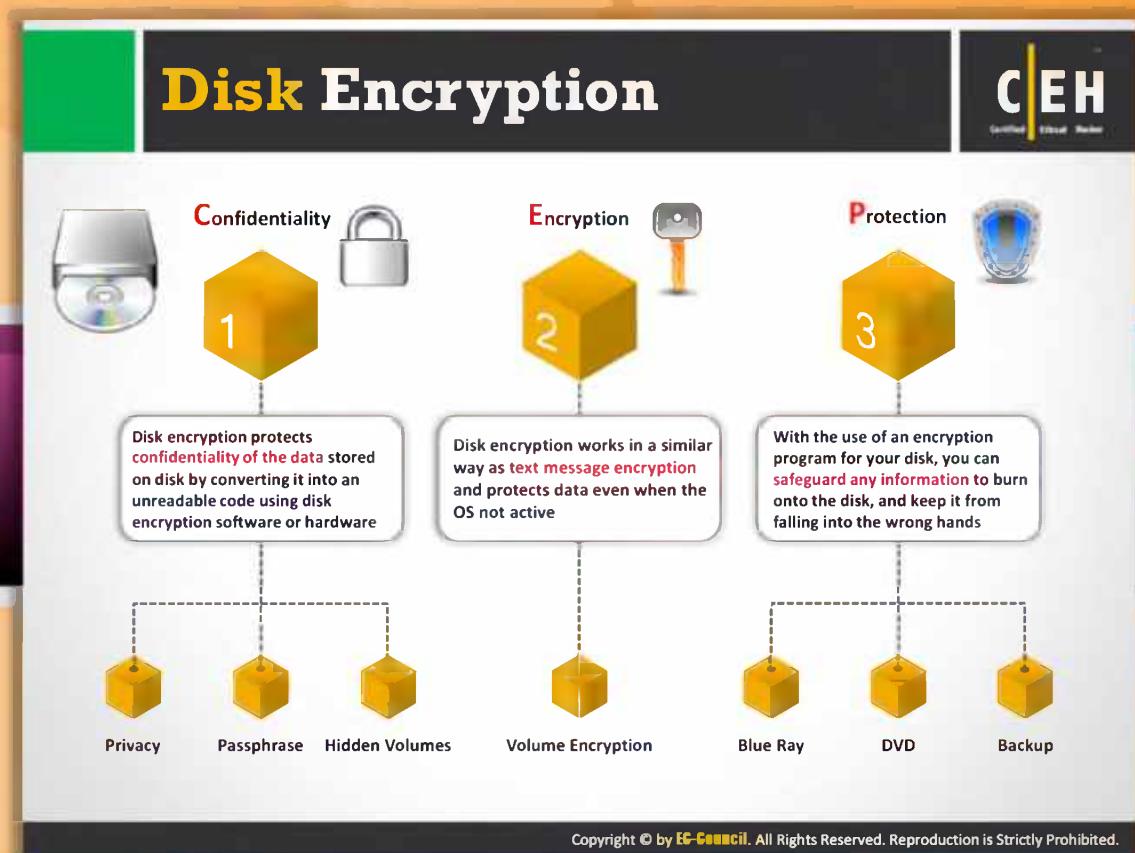


Module Flow

So far, we have discussed cryptography, the need for cryptography, **cryptographic encryption algorithms**, cryptography tools, PKI, and email encryption. In addition to all these encryption methods, there is one more encryption method: disk encryption.

Cryptography Concepts	Encryption Algorithms
Cryptography Tools	Public Key Infrastructure (PKI)
Email Encryption	Disk Encryption
Cryptography Attacks	Cryptanalysis Tools

This section describes disk encryption and disk encryption tools.



Disk Encryption

Disk encryption is the process of securing data by transferring it into unreadable code that cannot be deciphered by unauthorized persons. You can use **disk encryption software** or hardware to encrypt every bit of information that is written on the disk.

Disk encryption works similar to text message encryption. With the use of an encryption program for the user's disk, the user can safeguard any, and all, information burned onto the disk and save it from falling into **wrong hands**.

A computer disk is a round plate onto which data is recorded and/or burned. If the user needs to store information on a disk, and keep it safe, it is recommended that an encryption program be used. Encryption software, for disks, scrambles the information burned on the disk into an illegible code. It is only after the disk information is **decrypted**, that it can be read and/or used.

Encryption for disks is useful when the user needs to send sensitive information through the mail. For instance, the user needs to mail his or her friend a disk, but cannot take the risk of it being stolen and the information is being compromised. In this case, the user could simply encrypt the information on the disk and then rest assured, even if the **disk** is lost or stolen, the information on it would not be compromised.

In addition, disk encryption can also be useful in protecting the real-time exchange of information from being compromised. When the exchange of information is made in an

encrypted form, the chances of the information being compromised are minimized. The only way the attacker can access the information is by decrypting the message, which can only be done via the **authentication process**.

Furthermore, the **encryption software** installed on one's system ensures the security of the system. Thus, it is recommended to install encryption software on systems that hold valuable information and/or are exposed to unlimited data transfer in order to protect the data and information from compromise.



Disk Encryption Tool: TrueCrypt

Source: <http://www.truecrypt.org>

TrueCrypt is software that allows you to establish and maintain an encrypted volume (data storage device). No data stored on an encrypted volume can be read (decrypted) without using the correct password/keyfile(s) or correct encryption keys. The entire **file system** is encrypted (e.g., file names, folder names, contents of every file, free space, meta data, etc).

Main Features:

- ➊ Creates a virtual encrypted disk within a file and mounts it as a real disk
- ➋ Encrypts an entire partition or storage device such as USB flash drive or hard drive
- ➌ Encrypts a partition or drive where Windows is installed (pre-boot authentication)
- ➍ Encryption can be hardware-accelerated on modern processors
- ➎ Provides plausible deniability, in case an adversary forces you to reveal the password
- ➏ Hidden volume (**steganography**) and hidden operating system

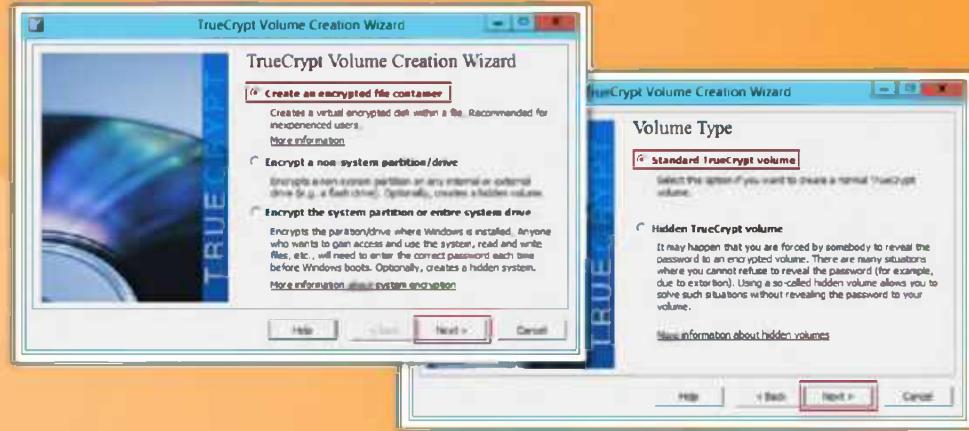
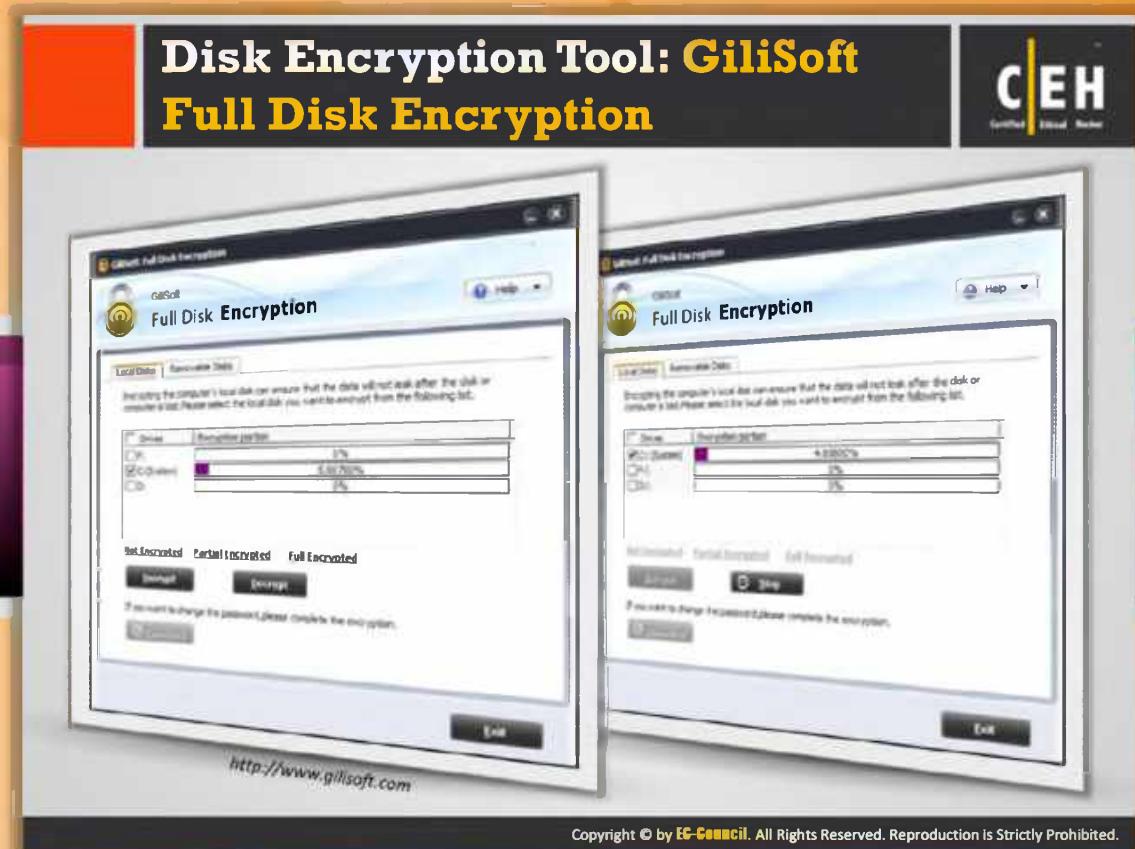


FIGURE 19.21: TrueCrypt Screenshot



Disk Encryption Tool: GiliSoft Full Disk Encryption

Source: <http://www.gilisoft.com>

GiliSoft Full Disk Encryption allows you to encrypt all disk partitions, including the system partition. Through password protecting a disk, disk partition, or operating system launch, the program disables any **unauthorized** reading/writing activity on your disk or PC and restricts access and launch of specific disks and files. It provides automatic security for all information on endpoint hard drives, including user data, operating system files, and temporary and erased files. For maximum data protection, **multi-factor pre-boot** authentication ensures user identity, while encryption prevents data loss from theft.

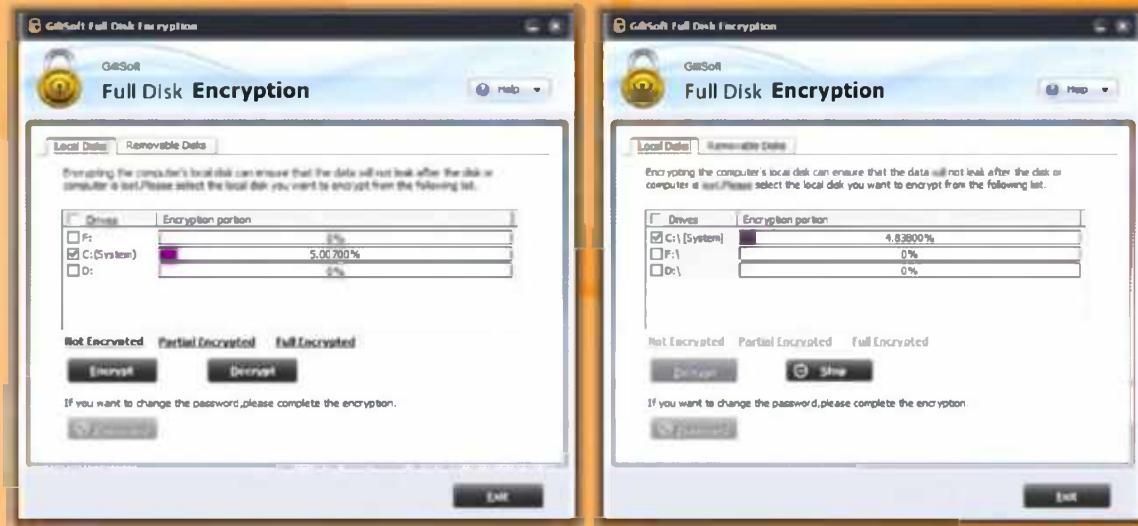


FIGURE 19.22: GiliSoft Full Disk Encryption screenshot

The slide features a yellow header bar with the title "Disk Encryption Tools" in white. To the right of the title is the CEH logo, which consists of the letters "CEH" in a stylized font with vertical bars above them, and the text "Certified Ethical Hacker" below it. Below the header is a grid of eight items, each containing a small icon, the tool name, and its website address. The items are:

- DriveCrypt (<http://www.securstar.com>)
- SafeBit Disk Encryption (<http://www.safebit.net>)
- ShareCrypt (<http://www.securstar.com>)
- DiskCryptor (<http://diskcryptor.net>)
- PocketCrypt (<http://www.securstar.com>)
- alertsec (<http://www.alertsec.com>)
- Rohos Disk Encryption (<http://www.rohos.com>)
- Symantec Drive Encryption (<http://www.symantec.com>)
- R-Crypto (<http://www.r-tt.com>)
- DriveCrypt Plus Pack (<http://www.securstar.com>)

At the bottom of the slide, a copyright notice reads: "Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited."

Disk Encryption Tools

In addition to TrueCrypt and GiliSoft Full Disk Encryption, there are many other disk encryption tools that allow you to fully encrypt all data. A list of **disk encryption tools** is mentioned below as follows. All these tools have a common goal, i.e., encrypting a disk partition. But environment or purpose may change. If one tool is intended to create a virtual encrypted disk of the **target disk partition**, then the other may be intended to encrypt data on Pocket PCs running Windows Mobile and so on:

- ➊ DriveCrypt available at <http://www.securstar.com>
- ➋ ShareCrypt available at <http://www.securstar.com>
- ➌ PocketCrypt available at <http://www.securstar.com>
- ➍ Rohos Disk Encryption available at <http://www.rohos.com>
- ➎ R-Crypto available at <http://www.r-tt.com>
- ➏ SafeBit Disk Encryption available at <http://www.safebit.net>
- ➐ DiskCryptor available at <http://diskcryptor.net>
- ➑ alertsec available at <http://www.alertsec.com>
- ➒ Symantec Drive Encryption available at <http://www.symantec.com>

- DriveCrypt Plus Pack available at <http://www.securstar.com>

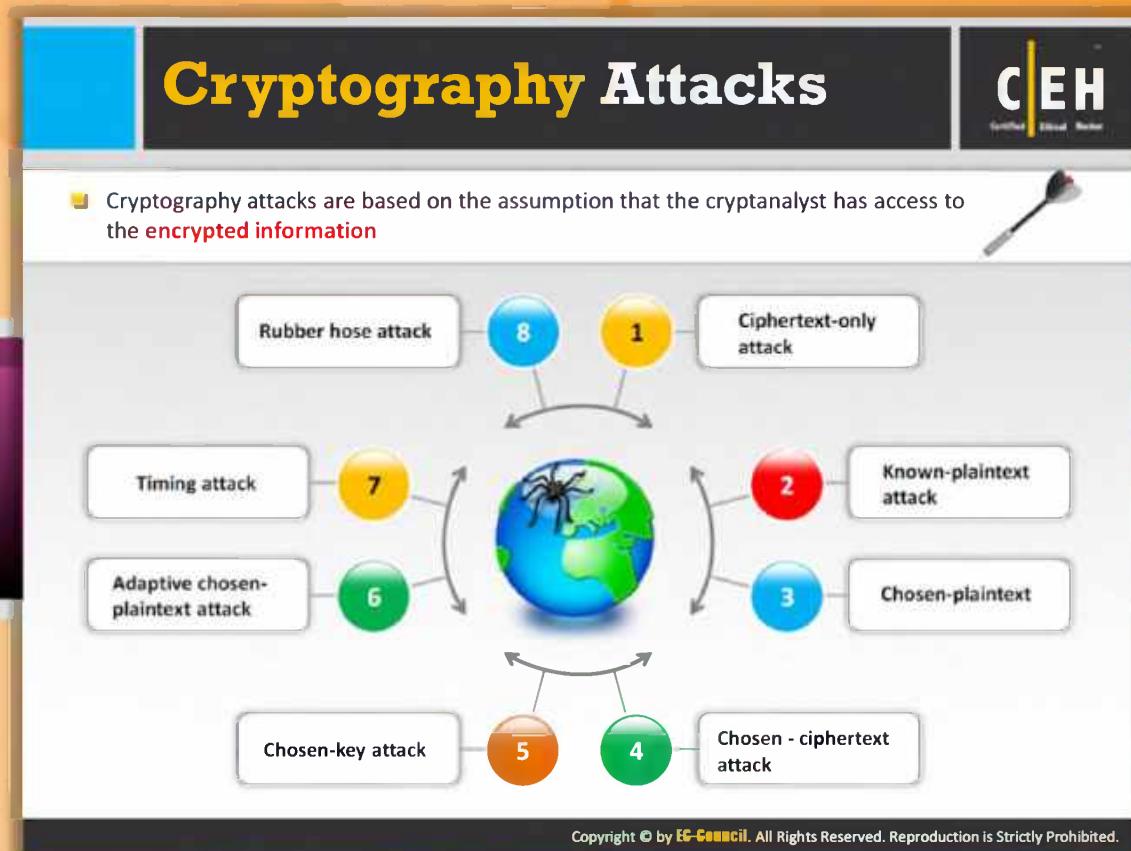


Module Flow

So far, we have discussed cryptography concepts, various cryptography mechanisms, and encryption algorithms. Now it's time to discuss how cryptography systems can be exploited by an external user.

Cryptography Concepts	Encryption Algorithms
Cryptography Tools	Public Key Infrastructure (PKI)
Email Encryption	Disk Encryption
Cryptography Attacks	Cryptanalysis Tools

This section focuses on various types of cryptography attacks, code breaking methodologies, and other kinds of attacks that exploit cryptography systems.



Cryptography Attacks

Cryptographic attacks are the means by which the attacker **decrypts** the ciphertext (breaks the ciphertext) without the knowledge of the key. In these attacks, the attacker subverts the cryptographic system's security by exploiting the loopholes in code, cipher, cryptographic protocol or key management scheme. **Cryptography attacks** are based on the assumption that the cryptanalyst has knowledge of the information encrypted. Attackers have found various attacks for defeating the cryptosystem and they are categorized into eight types:

- ⊕ Ciphertext only attack
- ⊕ Known-plaintext attack
- ⊕ Chosen-plaintext
- ⊕ Chosen-ciphertext attack
- ⊕ Chosen key attack
- ⊕ Adaptive chosen-plaintext attack
- ⊕ Timing attack
- ⊕ Rubber hose attack

Cryptography Attacks (Cont'd)

Ciphertext-only Attack
Attacker has access to the cipher text; goal of this attack to **recover encryption key** from the ciphertext



Adaptive Chosen-plaintext Attack
Attacker makes a **series of interactive queries**, choosing subsequent plaintexts based on the information from the previous encryptions



Chosen-plaintext Attack
Attacker defines his own plaintext, feeds it into the cipher, and analyzes the resulting ciphertext



Known-plaintext Attack
Attacker has **knowledge of some part of the plain text**; using this information the key used to generate ciphertext is deduced so as to decipher other messages



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Cryptography Attacks (Cont'd)

Attackers gain access to the content of the encrypted message through **cryptanalysis by defeating the cryptographic security algorithms**, even without the knowledge of encryption details. Though the algorithms are strong and are resistant to all attacks, the demands of practical cryptosystem easily introduce vulnerabilities. These vulnerabilities are the sources of various cryptography attacks. As discussed previously, there are eight types of cryptography attacks. All these attacks try either to retrieve the key or expose the plaintext. These attacks are distinguished based on the information available to the cryptanalyst to mount an attack. The main goal of attackers in all the cases is to decrypt the new pieces of **encrypted** message without additional information.



Ciphertext only attack

A ciphertext only attack is one of the basic types of active attacks because it is very easy for the attacker to get ciphertext by **sniffing** the traffic of any individual. In this type of attack, the attacker will have access only to ciphertexts of several messages, all of which were encrypted using the same encryption algorithm. Finding the key used for encryption is the main objective of the attacker as it allows the attacker to decode all the messages encrypted with the respective key.



Adaptive chosen-plaintext attack

An adaptive chosen-ciphertext is the **collaborative version** of the **chosen-plaintext** attack. In this type of attack, the attacker chooses further ciphertexts based on prior results. Here the cryptanalyst not only chooses the plaintext that is encrypted but can also modify his or her choice based on the results of the previous encryption.



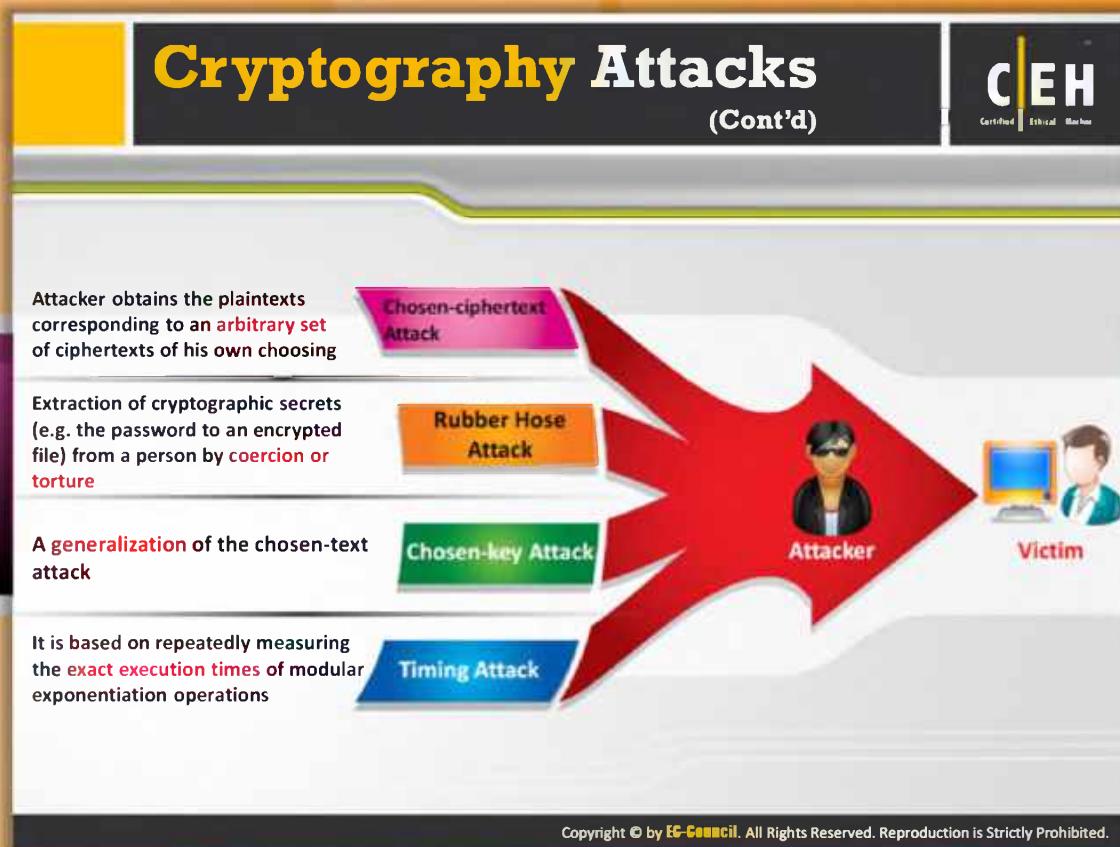
Chosen-ciphertext attack

In a chosen-ciphertext attack, the attacker chooses some part of ciphertext to be decrypted and tries to find out the corresponding decrypted plaintext. This is usually done with the help of a decryption oracle (a machine that decoded the text without disclosing the key). Basically, this type of attack is applicable to **public-key cryptosystems**. This attack is harder to perform when compared to other attacks, and the attacker needs to have complete control of system containing cryptosystem in order to carry out this attack.



Rubber hose attack

In a rubber hose attack, the attacker extracts the secret key from the user by **threatening, blackmailing**, or torturing him or her until the key is handed over.



Cryptography Attacks (Cont'd)



Chosen-plaintext

This is more powerful than a plaintext attack. In this type of attacker, the attacker not only has access to the **ciphertext** and associated plaintext for several messages, but also chooses the plaintext that is encrypted, and obtains the resulting ciphertext.



Known-plaintext attack

In a known-plaintext attack, the attacker has access to the ciphertext of one or more messages as well as access to the respective plaintext. With the help of both these items, the **cryptographic** key can easily be extracted. The attacker can recover the remaining encrypted, zipped files with the help of the extracted key.

In general, most people start their messages with the same type of beginning notes such as greetings and close with the same type of ending such as specific salutations, contact information, name, etc. Attackers can use this as an advantage to launch known-plaintext attacks. Here the attacker has some plaintext (i.e., the data that are the same on each message) and can capture an encrypted message, and therefore capture the **ciphertext**. Once the few parts of the message are discovered, the remaining can easily be accomplished with the help of reverse engineering, frequency analysis, or **brute force** attempts.



Chosen key attack

A chosen key attack is a generalization of the **chosen-text attack**. In this attack, the attacker has some knowledge about the relationship between the different keys, but cannot choose the key.



Timing Attack

A timing attack also is known as a side channel attack. In this type of attack, the attacker tries to compromise a cryptosystem by analyzing the time taken to execute **cryptographic algorithms**.

Code Breaking Methodologies



Code Breaking Methodologies

The strength of an encryption algorithm is measured, in large part by cryptanalysts, by using various code breaking techniques. The various **code-breaking** techniques that are available are:

- Brute-Force
 - Frequency Analysis
 - Trickery and Deceit
 - One-Time Pad



Brute-Force

Code-breakers, or cryptanalysts, want to recover the plaintext of a message without knowing the required key in advance. They may first try to recover the key, or go after the message itself. One of the familiar ways of the **cryptanalytic** technique is brute-force attack or an exhaustive search, (where the keys are guessed by trying every possible combination).

The efficiency of the brute-force depends on the hardware configuration. Usage of faster processors means testing more keys per second. Michael Weiner, put forth a brute-force attack

on the DES with the help of specially designed computers with **cryptographers** sounding the old standard's death knell.

Moreover, the combination of advanced factoring and the faster computers used in the recent attacks on **RSA-129**, makes algorithms appear weak. The NSA that has top computing power is the center of the brute-force attack.



Frequency Analysis

Frequency analysis of the letters makes the brute-force method not a suitable method for attacking the cipher. For example the letter "e" is the common word in the English language and the letter "k" appears commonly in the ciphertext, it can be concluded reasonably that $k=e$, and so on.

Encrypted source codes are more exposed to the attacks because few words like "**#define**," "**struct**," "**else**," and "**return**" are repeated frequently. Frequency analysis was first used by papal courts in the Middle Age, which built frequency tables for Latin and Italian words. Sophisticated cryptosystems are required to maintain the security of the messages.



Trickery and Deceit

There has always been a need for a high level of mathematical and cryptographic skills, but trickery and deceit have a long history in **code-breaking** as well the value of the encrypted data must be below the cost entitled to break the algorithm. In the modern world, computers are faster and cheaper, therefore it would be better to check the limits of these two parameters.



One-time Pad

It is considered that any cipher can be cracked if sufficient time and resources are provided. But there is an exception called a one-time pad, which is considered to be unbreakable even after infinite resources are provided.

A one-time pad contains many **non-repeating** groups of letters or number keys, which are chosen randomly. These are then pasted together on a pad.

Bob encrypts only one plaintext character with the pad and Alice decrypts each and every character of the ciphertext with the help of the same key characters from an identical pad. After the use, the characters are securely removed from the pad. The major drawback of the one-time **padding** is the length of the pads. The length of key is same as the length of the message, which makes it impossible to encrypt and send large messages.

The Soviet spies commonly used one-time pads during the Cold War. The agent carried the encrypted message to the field, leaving the identical pad at the headquarters. The well-known, one-time padding was used on the communication lines between **Moscow** and **Washington**.

Brute-Force Attack



Attack Scheme

Defeating a cryptographic scheme by trying a large number of possible keys until the correct encryption key is discovered




Brute-Force Attack

Brute-force attack is a high resource and time intensive process, however, more certain to achieve results




Success Factors

Success of brute force attack depends on length of the key, time constraint, and system security mechanisms




Power/Cost	40 bits (5 char)	56 bit (7 char)	64 bit (8 char)	128 bit (16 char)
\$ 2K (1 PC. Can be achieved by an individual)	1.4 min	73 days	50 years	10^{20} years
\$ 100K (this can be achieved by a company)	2 sec	35 hours	1 year	10^{19} years
\$ 1M (Achieved by a huge organization or a state)	0.2 sec	3.5 hours	37 days	10^{18} years

Estimate Time for Successful Brute-force Attack

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Brute-force Attack

It is very difficult to crack cryptographic systems as they have no practical weaknesses to exploit. But, it is not impossible. Cryptographic systems use **cryptographic algorithms** to encrypt a message. These cryptographic algorithms use a key to encrypt or decrypt messages. In cryptography, this key is the important parameter that specifies the transformation of plaintext to ciphertext and vice versa. If you are able to guess or find the key used for decryption then you can decrypt the messages and read it in clear text; **128-bit keys** are commonly used and considered strong. From security perspectives to avoid the key being guessed, the cryptographic systems use randomly generated keys. This makes you put a lot of effort in guessing the key. But you still have a choice to determine the key used for encryption or decryption. Attempt to decrypt the message with all possible keys until you discover the key used for encryption. This method of discovering a key is usually called a brute-force attack. In a brute-force attack, the attacker tries every possible key until the message can be decrypted. But this needs a huge amount of processing power for determining the key used to secure cryptographic communications. For any **non-flawed protocol**, the average time needed to find the key in a brute-force attack depends on the length of the key. If the key length is small, then it will take less time to find the key. If key length is larger, then it will take more time to discover the key. A **brute-force attack** will be successful if and only if enough time is given for discovering the key. However, the time is relative to the length of the key.

The difficulty of a brute-force attack depends on various issues, such as:

- ❑ Length of the key
- ❑ The numbers of possible values each component of the key can have
- ❑ The time it takes to attempt each key
- ❑ If there is any mechanism, which locks the attacker out after a certain number of failed attempts

For example, if a system could brute force a **DES 56-bit key** in one second, then for an AES 128-bit key it takes approximately **149 trillion** years to brute force. To perform a brute-force attack, the time is doubled for every additional bit of key length; the reason behind it is that the number of potential keys is doubled.

A brute-force attack is, however, more certain to achieve results.

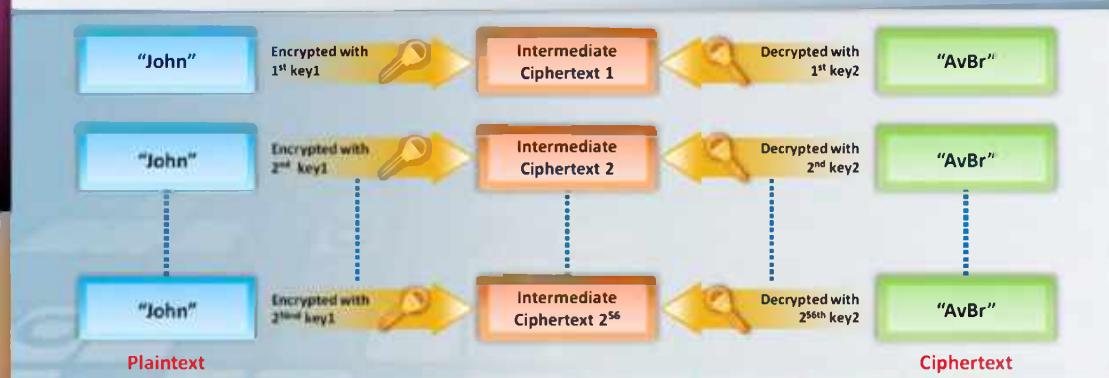
Estimate Time for Successful Brute – Force Attack

Power/Cost	40 bits (5 char)	56 bit (7 char)	64 bit (8 char)	128 bit (16 char)
\$ 2K (1 PC. Can be achieved by an individual)	1.4 min	73 Days	50 Years	10^{20} Years
\$ 100K (this can be achieved by a company)	2 Sec	35 Hours	1 Year	10^{19} Years
\$ 1M (Achieved by a huge organization or a state)	0.2 Sec	3.5 Hours	37 Days	10^{18} Years

TABLE 19.2: Time estimation for successful Brute-Force Attack

Meet-in-the-Middle Attack on Digital Signature Schemes

- The attack works by encrypting from one end and decrypting from the other end, thus meeting in the middle
- It can be used for forging signatures even on digital signatures that use multiple-encryption scheme



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Meet-in-the-Middle Attack on Digital Signature Schemes

A meet-in-the-middle attack is the best attack method for cryptographic algorithms using multiple keys for encryption. This attack reduces the number of brute force permutations needed to decode text that has been encrypted by more than one key and is conducted mainly for forging signatures on mixed type digital signatures. A **meet-in-the-middle** attack uses space-time trade-off; it is also known as birthday attack because it exploits the mathematics behind the **birthday paradox**. It takes less time than an exhaustive attack. It is called a meet-in-the-Middle attack because this attack works by encrypting from one end and decrypting from the other end, thus meeting in the middle.

In the meet-in-the-middle attack, the attacker uses a known plaintext message. The attacker has access to both the plaintext as well as the respective encrypted text.

Consider an example where the plain text is "John" and the resulting double DES encrypted message is "AvBr."

In order to recover both the keys, i.e. key1 and key2, that are used for encryption, the attacker performs a brute-force attack on key1 using all 2^{56} different Single DES possible keys to encrypt the plaintext of "John" and saves each key and the resulting intermediate ciphertext in a table. The attacker conducts brute force on key2, decrypting "AvBr" up to 2^{56} times. The attack is successful, when the second brute-force attack gives the same result as that of the

intermediate ciphertext present in the ciphertext table after **first brute-force attack**. Once the match is found, both keys can be determined and the attack is complete. This attack at most takes 2^{56} plus or maximum 2^{57} total operations. This enables the attacker to gain access to the data easily when compared with the Double DES.

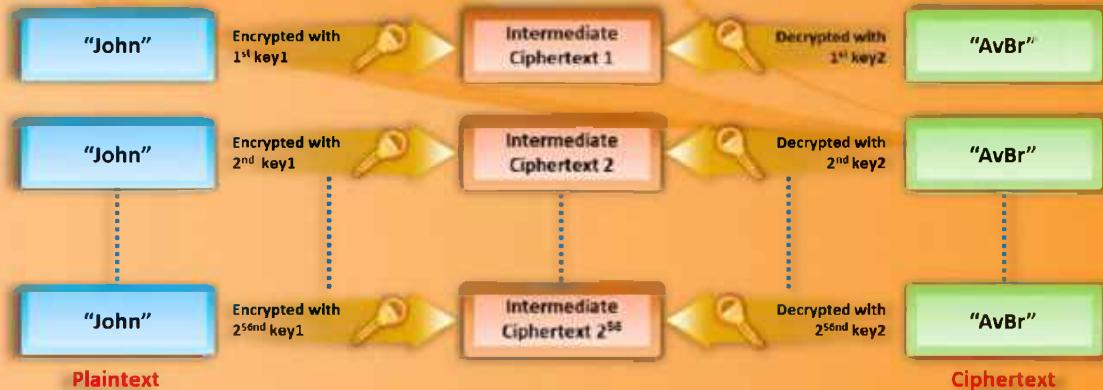
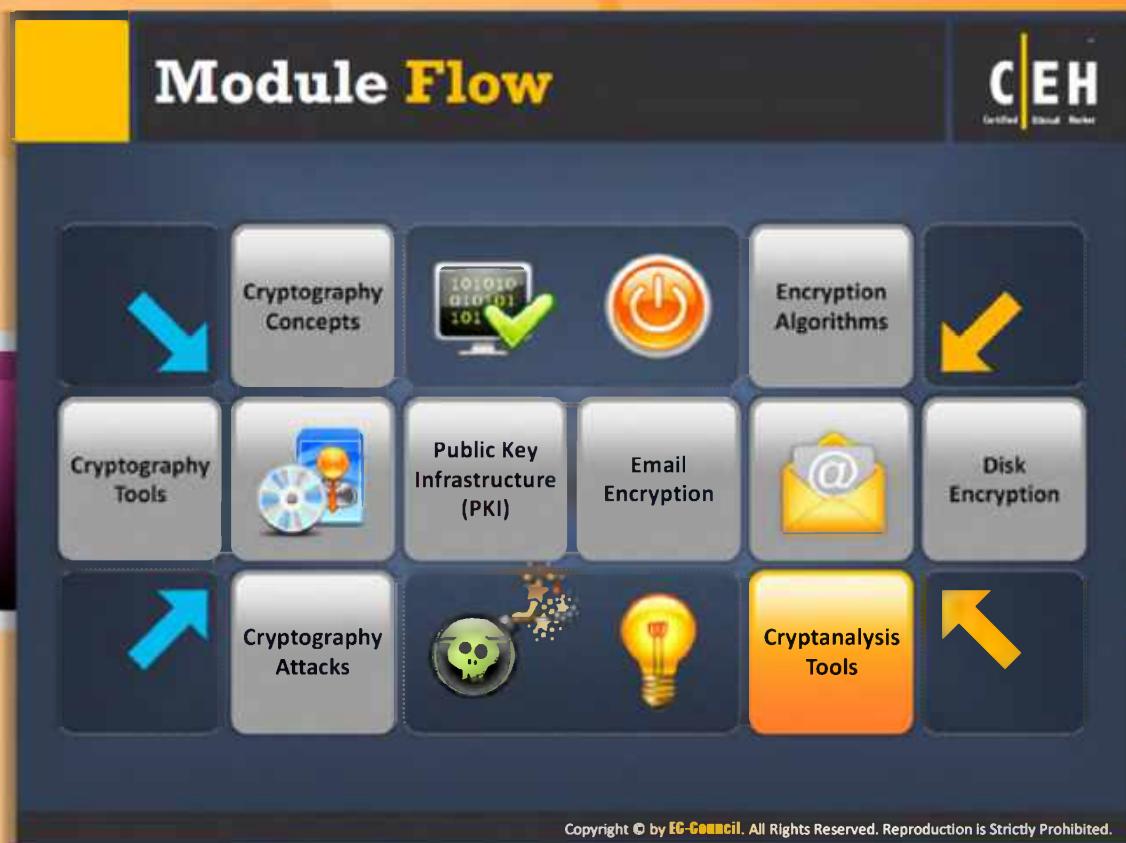


TABLE 19.23: Example illustrating Meet-in-the-middle attack

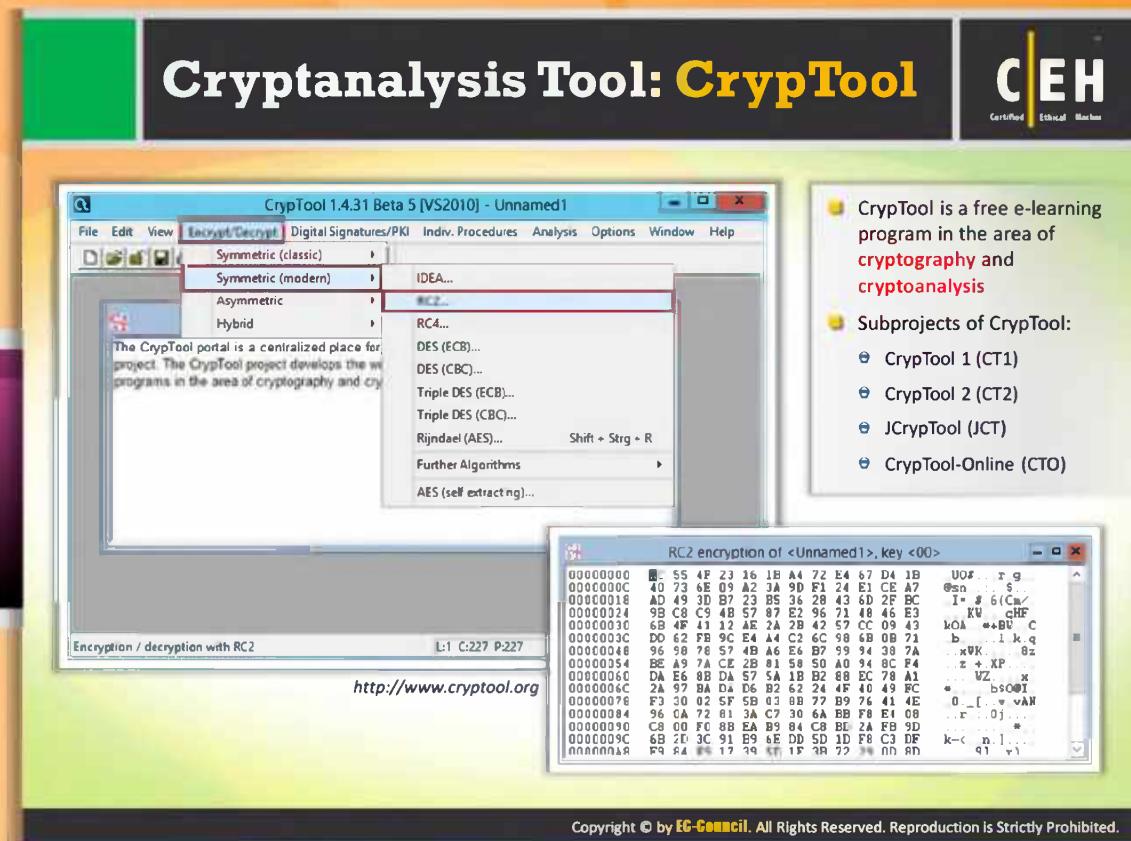


Module Flow

So far, we have discussed all cryptography concepts, various cryptographic encryption algorithms, tools that help in cryptography, email and disk encryption, and how cryptographic mechanisms can be compromised. Now it's time to discuss cryptanalysis tools that help you in breaking old ciphers.

Cryptography Concepts	Encryption Algorithms
Cryptography Tools	Public Key Infrastructure (PKI)
Email Encryption	Disk Encryption
Cryptography Attacks	Cryptanalysis Tools

This section describes and lists cryptanalysis tools.



Cryptanalysis Tool: CrypTool

Source: <http://www.cryptool.org>

The CrypTool project develops e-learning programs in the area of cryptography and cryptanalysis. It consists of four different subprojects: They are ([CT1](#), [CT2](#), [JCT](#), [CTO](#)) related to the **CrypTool software** in various facets for different purposes.

- ⊖ CrypTool 1 (CT1) was the first version of CrypTool. It was released in 1998 and allows to experiment with different cryptographic algorithms. CT 1 has two successors.
- ⊖ CrypTool 2 (CT2) supports visual programming and execution of cascades of cryptographic procedures.
- ⊖ JCrypTool (JCT) which is platform-independent.
- ⊖ CrypTool-Online (CTO) was released in spring 2009. This tool allows trying out different algorithms in a browser/smartphone.
- ⊖ Another subproject is the international crypto cipher challenge "MTC3," offering cryptographic riddles of different levels.

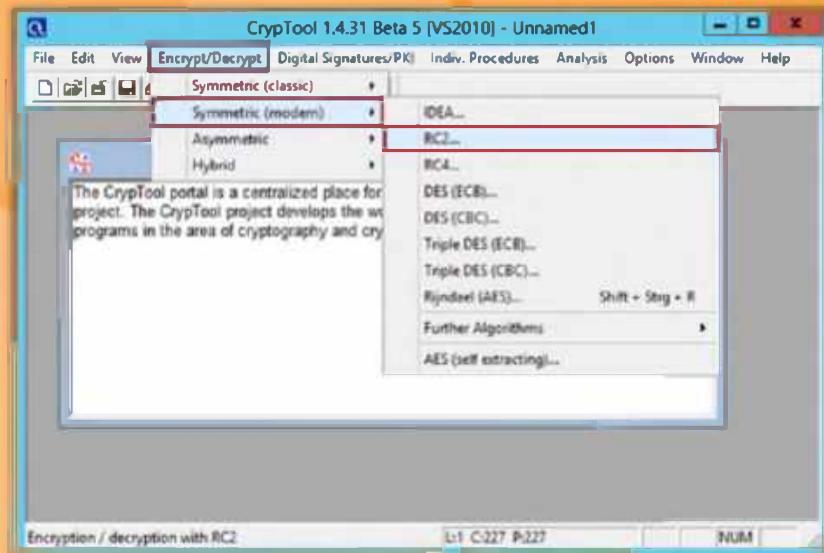


TABLE 19.24: CrypTool Screenshot

RC2 encryption of <Unnamed1>, key <00>	
00000000	EC 55 4F 23 16 1B A4 72 E4 67 D4 1B U0\$...r.g..
0000000C	40 73 6E 09 A2 3A 9D F1 24 E1 CE A7 @sn....@...
00000018	AD 49 3D B7 23 B5 36 28 43 6D 2F BC I= \$ 6(Cm/
00000024	9B C8 C9 4B 57 87 E2 96 71 48 46 E3 . KV.. qHF
00000030	6B 4F 41 12 AE 2A 2B 42 57 CC 09 43 KOA. *+BW.C
0000003C	DD 62 FB 9C E4 A4 C2 6C 98 6B 0B 71 .b....1.k.q
00000048	96 98 78 57 4B A6 E6 B7 99 94 38 7A .xVK. .8z
00000054	BE A9 7A CE 2B 81 58 50 A0 94 8C F4 .z.+ XP....
00000060	DA E6 8B DA 57 5A 1B B2 88 EC 78 A1 ...WZ....x
0000006C	2A 97 BA DA D6 B2 62 24 4F 40 49 FC *....bSO@I.
00000078	F3 30 02 5F 5B 03 8B 77 B9 76 41 4E .0._[..v.vAN
00000084	96 0A 72 81 3A C7 30 6A BB F8 E4 08 ..r.: 0j....
00000090	C8 00 F0 8B EA B9 84 C8 BD 2A FB 9D *
0000009C	6B 2D 3C 91 B9 6E DD 5D 1D F8 C3 DF k-<.n.]... 91 r)
000000A8	F9 R4 F9 17 39 SD 1F 3R 72 29 0D 0D

TABLE 19.24: RC2 encryption Screenshot

Cryptanalysis Tools



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

 CryptoBench http://www.addario.org	 AlphaPeeler http://alphapeeler.sourceforge.net
 JCrypTool http://www.cryptool.org	 Draft Crypto Analyzer http://www.literatecode.com
 Ganzúa http://ganza.sourceforge.net	 Linear Hull Cryptanalysis of PRESENT http://www.ecrypt.eu.org
 Crank http://crank.sourceforge.net	 mediggo http://code.google.com
 EverCrack http://evercrack.sourceforge.net	 SubCypher http://www.esclepiusllc.com



Cryptanalysis Tools

In addition to CrypTool, many tools that allow you to perform cryptanalysis are available:

- ⌚ [CryptoBench available at http://www.addario.org](http://www.addario.org)
- ⌚ [JCrypTool available at http://www.cryptool.org](http://www.cryptool.org)
- ⌚ [Ganzúa available at http://ganza.sourceforge.net](http://ganza.sourceforge.net)
- ⌚ [Crank available at http://crank.sourceforge.net](http://crank.sourceforge.net)
- ⌚ [EverCrack available at http://evercrack.sourceforge.net](http://evercrack.sourceforge.net)
- ⌚ [AlphaPeeler available at http://alphapeeler.sourceforge.net](http://alphapeeler.sourceforge.net)
- ⌚ [Draft Crypto Analyzer available at http://www.literatecode.com](http://www.literatecode.com)
- ⌚ [Linear Hull Cryptanalysis of PRESENT available at http://www.ecrypt.eu.org](http://www.ecrypt.eu.org)
- ⌚ [Mediggo available at http://code.google.com](http://code.google.com)
- ⌚ [SubCypher available at http://www.esclepiusllc.com](http://www.esclepiusllc.com)

Online MD5 Decryption Tools



The slide displays a grid of eight online MD5 decryption tools, each with an icon and a link:

 MD5 Decrypt http://www.md5decrypt.org	 OnlieHashCrack.com http://www.onlinehashcrack.com
 MD5Cracker http://md5crack.com	 MD5Decrypter.co.uk http://www.md5decrypter.co.uk
 MD5 Hash Cracker http://www.tmto.org	 Md5.My-Addr.com http://md5.my-addr.com
 Hash Cracker http://www.hash-cracker.com	 cmd5.org http://www.cmd5.org
 MD5Decrypter http://www.md5decrypter.com	 Crypt and Decrypt Online Tool Conversion http://myeasywww.appspot.com

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Online MD5 Decryption Tools

Online MD5 decryption tools allow you to read the encrypted messages in clear text. All you need to do is submit the MD5 hash of the message that you want to read to an online MD5 decryptor. It decrypts the MD5 hash value and simply gives you the original message that has been encrypted. These tools eliminate the need for installing **MD5 decryptors**. Many online MD5 decryption tools are readily available:

- ⌚ [MD5 Decrypt available at http://www.md5decrypt.org](http://www.md5decrypt.org)
- ⌚ [MD5Cracker available at http://md5crack.com](http://md5crack.com)
- ⌚ [MD5 Hash Cracker available at http://www.tmto.org](http://www.tmto.org)
- ⌚ [Hash Cracker available at http://www.hash-cracker.com](http://www.hash-cracker.com)
- ⌚ [MD5Decrypter available at http://www.md5decrypter.com](http://www.md5decrypter.com)
- ⌚ [OnlieHashCrack.com available at http://www.onlinehashcrack.com](http://www.onlinehashcrack.com)
- ⌚ [MD5Decrypter.co.uk available at http://www.md5decrypter.co.uk](http://www.md5decrypter.co.uk)
- ⌚ [Md5.My-Addr.com available at http://md5.my-addr.com](http://md5.my-addr.com)
- ⌚ [cmd5.org available at http://www.cmd5.org](http://www.cmd5.org)
- ⌚ [Crypt and Decrypt Online Tool Conversion available at http://myeasywww.appspot.com](http://myeasywww.appspot.com)

Module Summary



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

- ❑ Cryptography is the conversion of data into a scrambled code that is sent across a private or public network and decrypted by its recipients
- ❑ Using Public Key Infrastructure (PKI), anyone can send a confidential message using public information, which can only be decrypted with a private-key in the sole possession of the intended recipient
- ❑ AES is a symmetric-key algorithm for securing sensitive but unclassified material by U.S. government agencies
- ❑ Cryptography attacks are based on the assumption that the cryptanalyst has access to the encrypted information
- ❑ Public Key Infrastructure (PKI) is a set of hardware, software, people, policies, and procedures required to create, manage, distribute, use, store, and revoke digital certificates

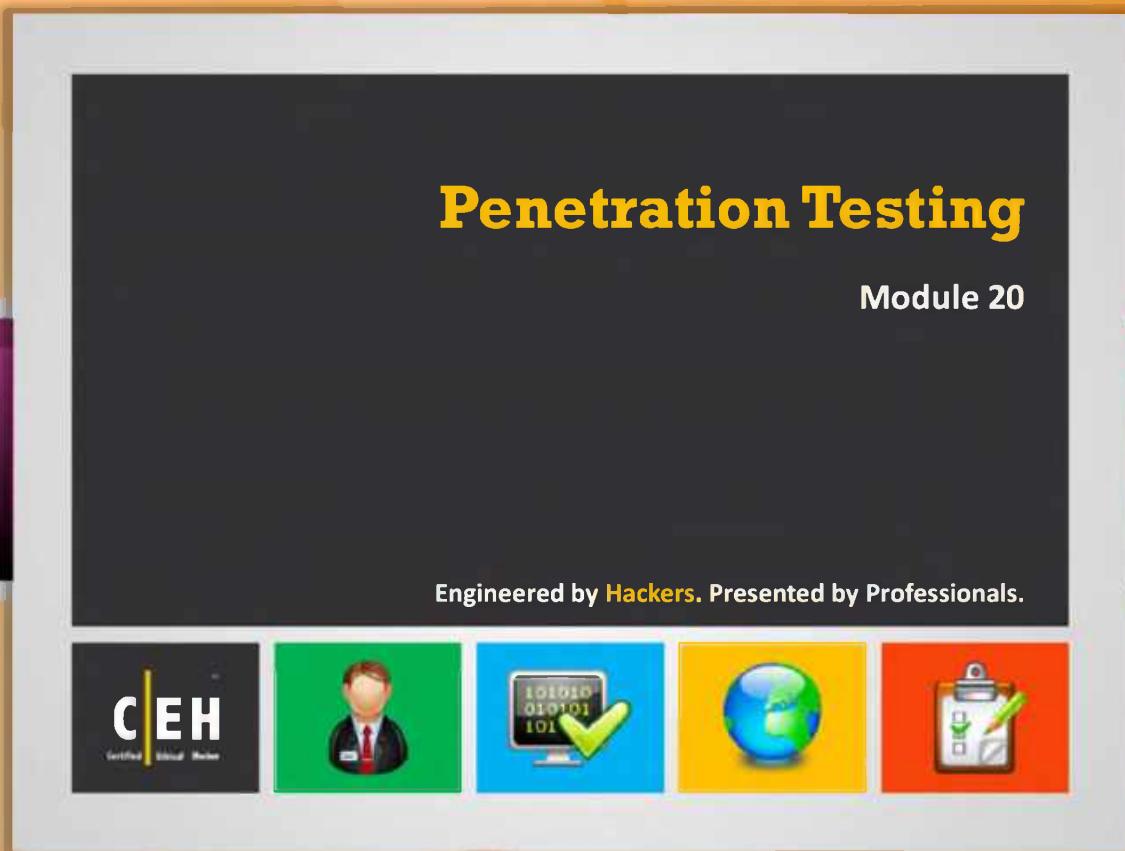


Module Summary

- ❑ Cryptography is the conversion of data into a scrambled code that is **decrypted** and sent across a private or public network.
- ❑ Using Public Key Infrastructure (PKI), anyone can send a confidential message using public information, which can only be decrypted with a private key in the sole possession of the intended recipient.
- ❑ RSA encryption is widely used and is a **de-facto encryption** standard.
- ❑ The MD5 algorithm is intended for digital signature applications, where a large file must be compressed securely before being encrypted.
- ❑ The **SHA algorithm** takes a message of arbitrary length as input and outputs a 160-bit message digest of the input.
- ❑ Secure Sockets Layer (SSL) is a protocol for transmitting private documents via the Internet.
- ❑ RC5 is a fast block cipher designed by RSA Security.

Penetration Testing

Module 20



Ethical Hacking and Countermeasures v8

Module 20: Penetration Testing

Exam 312-50

Security News

City of Tulsa Cyber Attack Was Penetration Test, Not Hack

The City of Tulsa, Oklahoma last week began notifying residents that their personal data may have been accessed -- but it now turns out that the **attack was a penetration test by a company the city had hired.**

"City officials didn't realize that the apparent breach was caused by the security firm, Utah-based SecurityMetrics, until after 90,000 letters had been sent to people who had applied for city jobs or made crime reports online over the past decade, warning them that their personal identification information might have been accessed," writes Tulsa World's Brian Barber. "The mailing cost the city \$20,000, officials said."

"**An additional \$25,000 was spent on security consulting services to add protection measures to the website,**" FOX23 News reports.

<http://www.esecurityplanet.com>

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Security News

City of Tulsa Cyber Attack Was Penetration Test, Not Hack

Source: <http://www.esecurityplanet.com>

The City of Tulsa, Oklahoma last week began notifying residents that their personal data may have been accessed -- but it now turns out that the attack was a penetration test by a company the city had hired.

"City officials didn't realize that the apparent breach was caused by the security firm, Utah-based SecurityMetrics, until after 90,000 letters had been sent to people who had applied for city jobs or made crime reports online over the past decade, warning them that their personal identification information might have been accessed," writes Tulsa World's Brian Barber. "The mailing cost the city \$20,000, officials said."

"An additional \$25,000 was spent on security consulting services to add protection measures to the website," FOX23 News reports.

"The third-party consultant had been hired to perform an assessment of the city's network for vulnerabilities," write NewsOn6.com's Dee Duren and Lacie Lowry. "The firm used an unfamiliar testing procedure that caused the City to believe its website had been compromised. 'We had

to treat this like a cyber-attack because every indication initially pointed to an attack,' said City Manager Jim Twombly."

"The chief information officer who failed to determine that the hack was actually part of a penetration test has been placed on administrative leave with pay," writes Softpedia's Eduard Kovacs. "In the meantime, his position will be filled by Tulsa Police Department Captain Jonathan Brook."



Copyright 2012 QuinStreet Inc

By Jeff Goldman

<http://www.esecurityplanet.com/network-security/city-of-tulsa-cyber-attack-was-penetration-test-not-hack.html>

Module Objectives



The slide features two columns of objectives. A large orange arrow points from the left column to the right column. Below the columns are three icons: a whiteboard, a document with a yellow seal, and a blue folder.

■ Security Assessments	■ Pre-Attack Phase
■ Vulnerability Assessment	■ Attack Phase
■ Penetration Testing	■ Post-Attack Phase
■ What Should be Tested?	■ Penetration Testing Deliverable Templates
■ ROI on Penetration Testing	■ Pen Testing Roadmap
■ Types of Penetration Testing	■ Web Application Testing
■ Common Penetration Testing Techniques	■ Outsourcing Penetration Testing Services

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



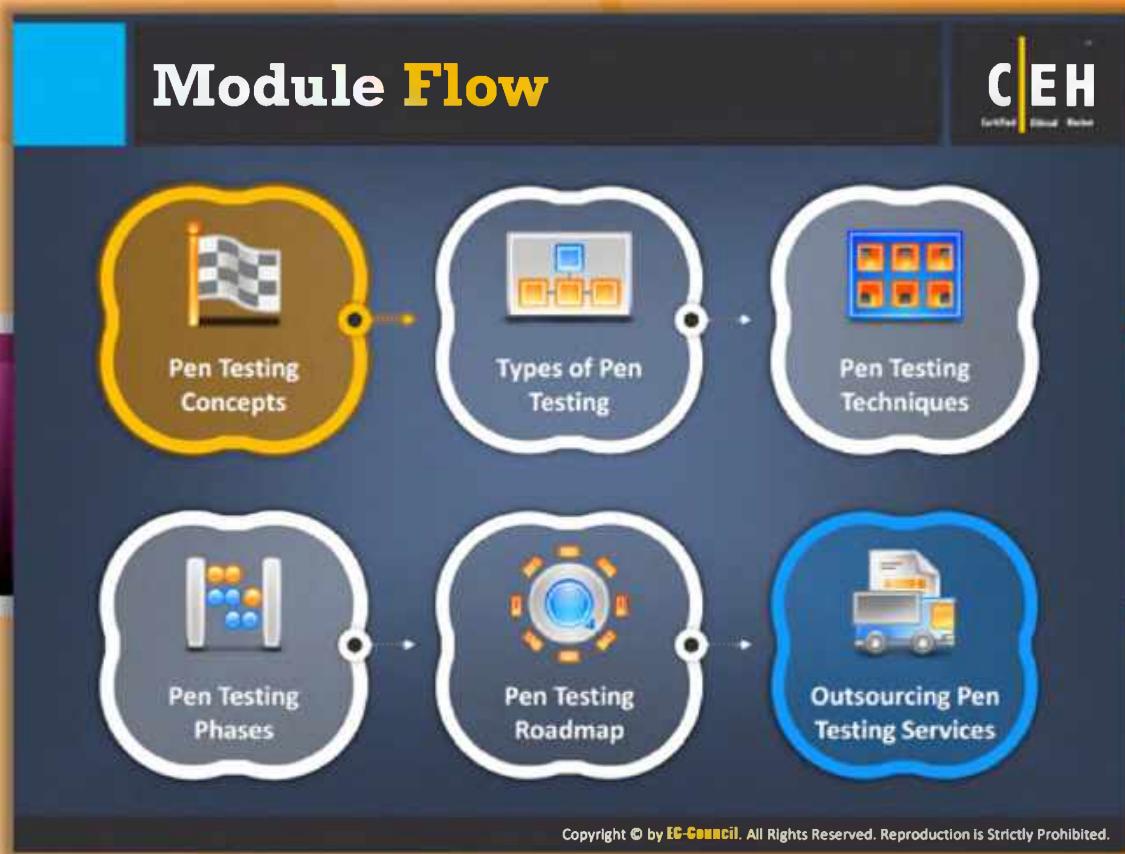
Module Objectives

All the modules discussed so far concentrated on various penetration testing techniques specific to the respective element (web application, etc.), mechanism (IDS, firewall, etc.), or phase (reconnaissance, scanning, etc.). This module summarizes all the **penetration tests**. This module helps you in evaluating the security of an organization and also guides you to make your network or system more secure with its **countermeasures**.

The module will make you familiarize with:

- Security Assessments
- Vulnerability Assessments
- Penetration Testing
- What Should be Tested
- ROI on Penetration Testing
- Types of Penetration Testing
- Common Penetration Testing Techniques

- Pre-attack Phase
- Attack Phase
- Post-attack Phase
- Penetration Testing Deliverable Templates
- Pen Testing Roadmap
- Web Application Testing
- Outsourcing Penetration Testing Services

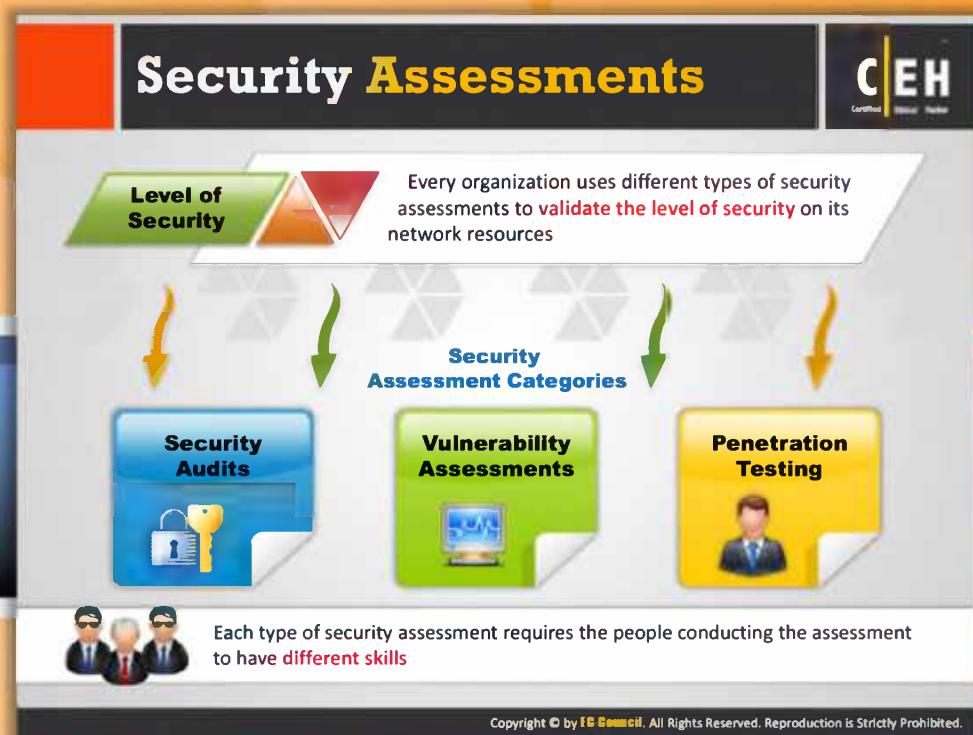


Module Flow

For better understanding of penetration testing, this module is divided into various sections. Let's begin with penetration testing concepts.

Pen Testing Concepts	Types of Pen Testing
Pen Testing Techniques	Pen Testing Phases
Pen Testing Roadmap	Outsourcing Pen Testing Services

This section starts with basic concept of penetration testing. In this section, you will learn the role of penetration testing in the security assessment and why vulnerability assessment alone is not enough to detect and remove vulnerabilities in the network. Later in this section, you will examine why penetration testing is necessary, how to perform a good penetration test, how to determine testing points, testing locations, and so on.



Security Assessments

Every organization uses different types of security assessments to validate the level of security on its network resources. Organizations need to choose the assessment method that suits the requirements of its situation most appropriately. People conducting different types of security assessments must possess different skills. Therefore, pen testers—if they are employees or outsourced security experts—must have a thorough experience of penetration testing. Security assessment categories include **security audits**, **vulnerability assessments**, and **penetration testing** or **ethical hacking**.



Security Assessment Categories

The security assessment is broadly divided into three categories:

1. **Security Audits:** IT security audits typically focus on the people and processes used to design, implement, and manage security on a network. There is a baseline involved for processes and policies within an organization. In an IT security audit, the auditor and the organization's security policies and procedures use the specific baseline to audit the organization. The **IT management** usually initiates IT security audits. The National Institute of Standards and Technology (NIST) has an IT security audit manual and associated toolset to conduct the audit; the NIST Automated Security Self-Evaluated Tool (ASSET) can be downloaded at <http://csrc.nist.gov/asset/>.

In a computer, the **security audit technical assessment** of a system or application is done manually or automatic.

You can perform a manual assessment by using the following techniques:

- ⌚ Interviewing the staff
- ⌚ Reviewing application and operating systems access controls
- ⌚ Analyzing physical access to the systems.

You can perform an automatic assessment by using the following techniques:

- ⌚ Generating audit reports
- ⌚ Monitoring and reporting the changes in the files

2. **Vulnerability Assessments:** A vulnerability assessment helps you in identifying **security vulnerabilities**. To perform a vulnerability assessment you should be a very skilled professional. Through proper assessment, threats from hackers (outsiders), former employees, internal employees, etc. can be determined.
3. **Penetration Testing:** Penetration testing is the act of testing an **organization's security** by simulating the actions of an attacker. It helps you in determining various levels of vulnerabilities and to what extent an external attacker can damage the network, before it actually occurs.

Security Audit

C|EH
Certified Ethical Hacker

- 1**
A security audit is a systematic evaluation of an **organization's compliance** to a set of established information security criteria
- 2**
Security audit includes assessment of a system's software and hardware configuration, physical security measures, data handling processes, and user practices against a **checklist of standard policies and procedures**
- 3**

A security audit ensures that an organization has and deploys a set of **standard information security policy**
- 4**

It is generally used to achieve and demonstrate compliance to **legal and regulatory requirements** such as HIPPA, SOX, PCI-DSS, etc.

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Security Audit

A security audit is a systematic, measurable technical assessment of how the security policy is employed by the organization. A security audit is conducted to maintain the security level of the particular organization. It helps you to identify attacks that pose a threat to the network or attacks against resources that are considered valuable in risk assessment. The security auditor is responsible for **conducting security audits on the particular organization**. The security auditor works with the full knowledge of the organization, at times with considerable inside information, in order to understand the resources to be audited.

- ➊ A security audit is a systematic evaluation of an organization's compliance to a set of established information security criteria.
- ➋ The security audit includes assessment of a system's software and hardware configuration, physical security measures, data handling processes, and user practices against a checklist of **standard policies** and procedures.
- ➌ A security audit ensures that an organization has and deploys a set of standard information security policies.
- ➍ It is generally used to achieve and demonstrate compliance to legal and regulatory requirements such as **HIPPA, SOX, PCI-DSS**, etc.

The slide has a dark blue header bar with the title "Vulnerability Assessment" in white. In the top right corner is the CEH logo with the text "Certified Ethical Hacker". The main content area is divided into four colored boxes: blue, yellow, green, and red. A central globe icon is positioned between the blue and yellow boxes. The blue box contains "Network Scanning" information. The yellow box contains "Scanning Tools" information. The green box contains "Security Mistakes" information. The red box contains "Test Systems/Network" information. At the bottom of the slide, a copyright notice reads: "Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited."

Network Scanning	Scanning Tools
Vulnerability assessment scans a network for known security weaknesses	Vulnerability scanning tools search network segments for IP-enabled devices and enumerate systems, OS's, and applications

Security Mistakes	Test Systems/Network
Additionally, vulnerability scanners can identify common security configuration mistakes	Vulnerability scanners can test systems and network devices for exposure to common attacks



Vulnerability Assessment

A vulnerability assessment is a basic type of security. This assessment helps you in finding the known security weaknesses by scanning a network. With the help of vulnerability-scanning tools, you can search network segments for **IP-enabled devices** and **enumerate systems, operating systems, and applications**. Vulnerability scanners are capable of identifying device configurations including the OS version running on computers or devices, IP protocols and Transmission Control Protocol/User Datagram Protocol (TCP/UDP) ports that are listening, and applications that are installed on computers.

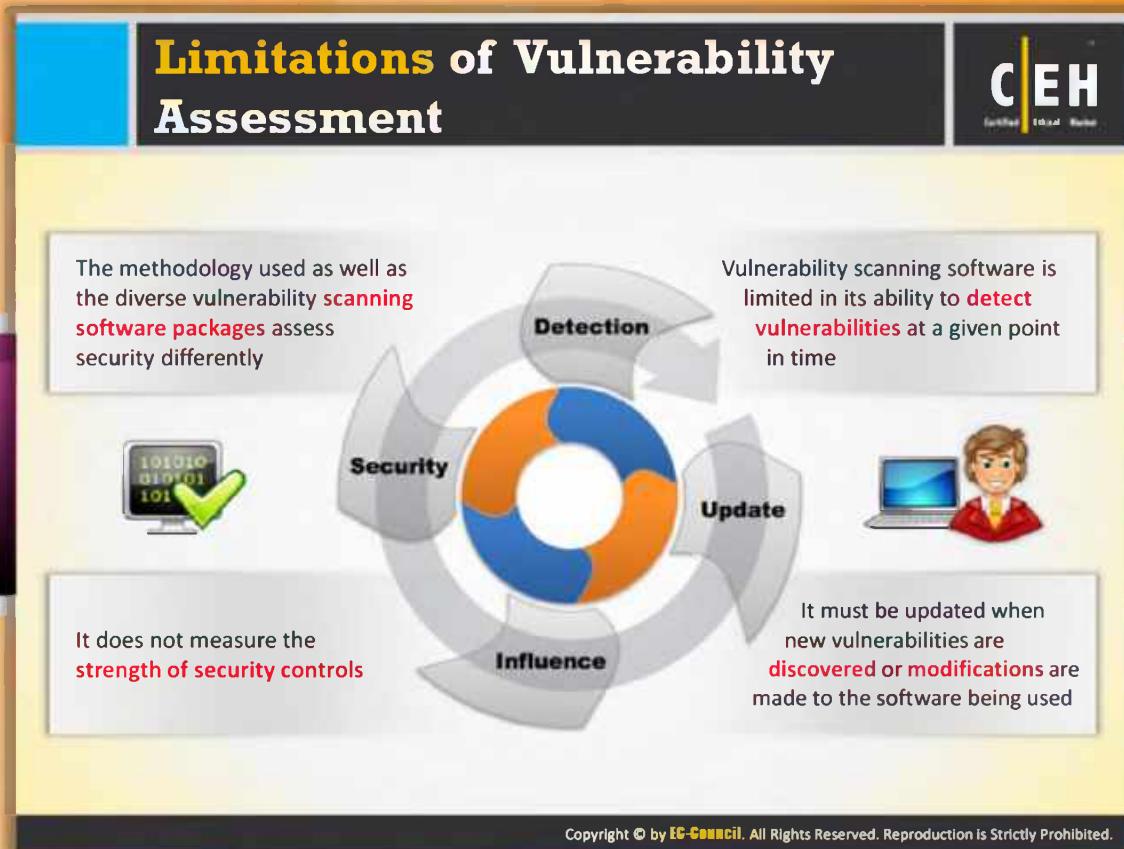
By using vulnerability scanners, you can also identify common security mistakes such as accounts that have weak passwords, files and folders with weak permissions, default services and applications that might need to be uninstalled, and mistakes in the security configuration of common applications. They can search for computers exposed to known or publicly reported vulnerabilities. The software packages that perform vulnerability scanning scan the computer against the Common Vulnerability and Exposures (CVE) index and security bulletins provided by the software vendor. The CVE is a **vendor-neutral** listing of reported security vulnerabilities in major operating systems and applications and is maintained at <http://cve.mitre.org/>.

Vulnerability scanners can test systems and network devices for exposure to common attacks. This includes common attacks such as the enumeration of security-related information and denial-of-service attacks. However, it must be noted that vulnerability scanning reports can

expose weaknesses in hidden areas of applications and frequently include many false positives. Network administrators who analyze vulnerability scan results must have sufficient knowledge and experience with the operating systems, network devices, and applications being scanned and their roles in the network.

You can use two types of automated vulnerability scanners depending upon the situation: network-based and host-based. Network-based scanners attempt to detect vulnerabilities from the outside. They are normally launched from a remote system, outside the organization, and without an authorized user access. For example, **network-based scanners examine** a system for such exploits as open ports, application security exploits, and buffer overflows.

Host-based scanners usually require a software agent or client to be installed on the host. The client then reports back the vulnerabilities it finds to the server. Host-based scanners look for features such as weak file access permissions, poor passwords, and logging faults.



Limitations of Vulnerability Assessment

Vulnerability scanning software allows you to detect limited vulnerabilities at a given point in time. As with any assessment software, which requires the signature file to be updated, vulnerability scanning software must be updated when new vulnerabilities are discovered or improvements made to the software are being used. The vulnerability software is only as effective as the maintenance performed on it by the software vendor and by the administrator who uses it. Vulnerability scanning software itself is not immune to software engineering flaws that might lead to **non-detection** of serious vulnerabilities.

Another aspect to be noted is that the methodology used might have an impact on the result of the test. For example, vulnerability scanning software that runs under the security context of the domain administrator will yield different results than if it were run under the security context of an authenticated user or a **non-authenticated** user. Similarly, diverse vulnerability scanning software packages assess security differently and have unique features. This can influence the result of the assessment. Examples of vulnerability scanners include **Nessus** and **Retina**.

Introduction to Penetration Testing

CEH
Certified Ethical Hacker

A pentest simulates methods that intruders use to gain unauthorized access to an organization's networked systems and then compromise them.

In the context of penetration testing, the tester is limited by resources - namely time, skilled resources, and access to equipment - as outlined in the penetration testing agreement.

Most attackers follow a common approach to penetrate a system.

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Introduction to Penetration Testing

This module marks a departure from the approach followed in earlier modules; here you will be encouraged to think “outside the box.” Hacking as it was defined originally portrayed a streak of genius or brilliance in the ability to conjure previously unknown ways of doing things. In this context, to advocate a methodology that can be followed to simulate a real-world hack through ethical hacking or penetration testing might come across as a contradiction. Penetration testing is a process of evaluating the security of the network by trying all possible attack vectors like an attacker does. The reason behind advocating a methodology in penetration testing arises from the fact that most attackers follow a common underlying approach when it comes to penetrate a system.

In the context of penetration testing, as a tester you will be limited by resources such as time, skilled resources, and access to equipment, as outlined in the penetration testing agreement. The paradox of penetration testing is the fact that the inability to breach a target does not necessarily indicate the absence of vulnerability. In other words, to maximize the returns from a penetration test, you must be able to apply your skills to the resources available in such a manner that the attack area of the target is reduced as much as possible.

A pen test simulates methods that intruders use to gain unauthorized access to an organization's networked systems and then compromise them. It involves using proprietary and

open source tools to test for known and unknown technical vulnerabilities in networked systems. Apart from automated techniques, penetration testing involves manual techniques for conducting targeted testing on specific systems to ensure that there are no **security flaws** that may have gone undetected earlier.

The main purpose behind **footprinting** pen testing is to gather data related to a target system or network and find out its vulnerabilities. You can perform this through various techniques such as DNS queries, network enumeration, network queries, **operating system identification**, organizational queries, ping sweeps, point of contact queries, port scanning, registrar queries, and so on.

Penetration Testing



Penetration testing that is not completed professionally can result in the **loss of services** and disruption of the business continuity

Penetration testing assesses the **security model** of the organization as a whole

It reveals **potential consequences** of a real attacker breaking into the network

A penetration tester is differentiated from an attacker only by his **intent and lack of malice**

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Penetration Testing

Penetration testing goes a step beyond vulnerability scanning in the category of security assessments. With vulnerability scanning, you can only examine the security of the individual computers, network devices, or applications, but penetration testing allows you to assess the security model of the network as a whole. Penetration testing can help you to reveal potential consequences of a real attacker breaking into the network to **network administrators, IT managers, and executives**. Penetration testing also reveals the security **weaknesses** that a typical vulnerability scanning misses.

A penetration test will not only point out vulnerabilities, it will also document how the weaknesses can be exploited and how several minor vulnerabilities can be escalated by an attacker to compromise a computer or network. Penetration testing must be considered as an activity that shows the holes in the security model of an organization. Penetration testing helps organizations to reach a balance between technical prowess and business functionality from the perspective of **potential security breaches**. This test can help you in disaster recovery and **business continuity planning**.

Most vulnerability assessments are carried out solely based on **software** and cannot assess security that is not related to technology. Both people and processes can be the source of security vulnerabilities as much as the technology can be. Using social engineering techniques, penetration tests can reveal whether employees routinely allow people without identification

to enter company facilities and where they would have physical access to computers. Practices such as patch management cycles can be evaluated. A penetration test can reveal process problems, such as not applying security updates until three days after they are released, which would give attackers a **three-day window** to exploit known vulnerabilities on servers.

You can differentiate a penetration tester from an attacker only by his or her intent and lack of malice. Therefore, employees or external experts must be cautioned against conducting penetration tests without proper authorization. Penetration testing that is not completed professionally can result in the loss of services and **disruption of business continuity**.

Management needs to give written approval for penetration testing. This approval should include a clear scoping, a description of what will be tested, and when the testing will take place. Because of the nature of penetration testing, failure to obtain this approval might result in committing computer crime, despite the best intentions.

Why Penetration Testing

The diagram features a central vertical axis with five circular markers, each accompanied by a blue square icon. To the left of the axis is a list of five reasons, and to the right is another list of five reasons.

- ① Identify the **threats** facing an organization's information assets
- ② Reduce an organization's expenditure on IT security and enhance **Return On Security Investment (ROSI)** by identifying and remediating vulnerabilities or weaknesses
- ③ Provide assurance with comprehensive assessment of organization's security including policy, procedure, design, and implementation
- ④ Gain and maintain certification to an **industry regulation** (BS7799, HIPAA etc.)
- ⑤ Adopt **best practices** in compliance to legal and industry regulations

- ⑥ For testing and validating the **efficiency** of security protections and controls
- ⑦ It focuses on high severity vulnerabilities and emphasizes **application-level security** issues to development teams and management
- ⑧ Providing comprehensive approach of **preparation steps** that can be taken to prevent upcoming exploitation
- ⑨ Evaluating the **efficiency** of **network security devices** such as firewalls, routers, and web servers
- ⑩ For **changing or upgrading** existing infrastructure of software, hardware, or network design

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Why Penetration Testing?

Penetration testing plays a vital role in evaluating and maintaining security of a system or network. It helps you in finding out the loopholes by deploying attacks. It includes both script-based testing as well as human-based testing on networks. A penetration test not only reveals network security holes, but also provides **risk assessment**. Let's see what you can do with the help of penetration testing:

- ① You can identify the threats facing an organization's information assets.
- ② You can reduce an organization's IT security costs and provide a better Return On IT Security Investment (ROSI) by identifying and resolving vulnerabilities and weaknesses.
- ③ You can provide an organization with assurance: a thorough and comprehensive assessment of organizational security covering policy, procedure, design, and implementation.
- ④ You can gain and maintain certification to an industry regulation (BS7799, HIPAA, etc.).
- ⑤ You can adopt best practices by conforming to legal and industry regulations.
- ⑥ You can test and validate the efficiency of security protections and controls.
- ⑦ It focuses on high-severity vulnerabilities and emphasizes application-level security issues to development teams and management.

- ⌚ It provides a comprehensive approach of preparation steps that can be taken to prevent upcoming exploitation.
- ⌚ You can evaluate the efficiency of network security devices such as firewalls, routers, and web servers.
- ⌚ You can use it for changing or **upgrading existing infrastructure** of software, hardware, or network design.



Comparing Security Audit, Vulnerability Assessment, and Penetration Testing

Although a lot of people use the terms security audit, vulnerability assessment, and penetration test interchangeably to mean security assessment, there are considerable differences between them.

Security Audit	Vulnerability Assessment	Penetration Testing
A security audit just checks whether the organization is following a set of standard security policies and procedures	A vulnerability assessment focuses on discovering the vulnerabilities in the information system but provides no indication if the vulnerabilities can be exploited or the amount of damage that may result from the successful exploitation of the vulnerability	Penetration testing is a methodological approach to security assessment that encompasses the security audit and vulnerability assessment and demonstrates if the vulnerabilities in system can be successfully exploited by attackers

TABLE 20.1: Comparison between Security Audit, Vulnerability Assessment, and Penetration Testing

What Should be Tested?



An organization should conduct a risk assessment operation before the penetration testing that will help to **identify the main threats**, such as:

Communications failure and e-commerce failure	Public facing systems; websites, email gateways, and remote access platforms	FTP, IIS, and web servers
1 Loss of confidential information	2 Mail, DNS, firewalls, and passwords	3 4 5

Note: Testing should be performed on all hardware and software components of a network security system

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



What Should be Tested?

It is always ideal to conduct a vulnerability assessment in an organization so that various potential threats can be known well before they occur. You can test various network or system components for **security vulnerabilities**, such as:

- ⌚ Communication failure
- ⌚ E-commerce failure
- ⌚ Loss of confidential information
- ⌚ Public facing systems websites
- ⌚ Email gateways
- ⌚ Remote access platforms
- ⌚ Mail
- ⌚ DNS
- ⌚ Firewalls
- ⌚ Passwords
- ⌚ FTP
- ⌚ IIS
- ⌚ Web servers

What Makes a Good Penetration Test?



Establishing the **parameters for the penetration test** such as objectives, limitations, and the justification of procedures

Hiring **skilled and experienced professionals** to perform the test

Choosing a **suitable set of tests** that balance cost and benefits

Following a methodology with **proper planning** and documentation

Documenting the result carefully and making it comprehensible for the client

Stating the **potential risks and findings** clearly in the final report

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



What Makes a Good Penetration Test?

Consider the following factors to perform a good penetration test:

- ➊ Establish the parameters for the penetration test such as objectives, limitations, and the justification of procedures. The establishment of these **parameters** helps you in know the purpose of conducting penetration test.
- ➋ Hire skilled and experienced professionals to perform the test. If the penetration testing is not done by the skilled and experienced professionals there are chances of damaging the live data and more harm can happen than the benefits.
- ➌ Choose a suitable set of tests that balance cost and benefits.
- ➍ Follow a methodology with proper planning and documentation. It is very important to document the test at each phase for the further references.
- ➎ Document the result carefully and making it comprehensible for the client.
- ➏ State the **potential risks** and findings clearly in the final report.

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



ROI on Penetration Testing

ROI (return on investment) is a traditional financial measure. It is used to determine the business results of for the future based on the calculations of **historical data**. The ROI is calculated based on three things:

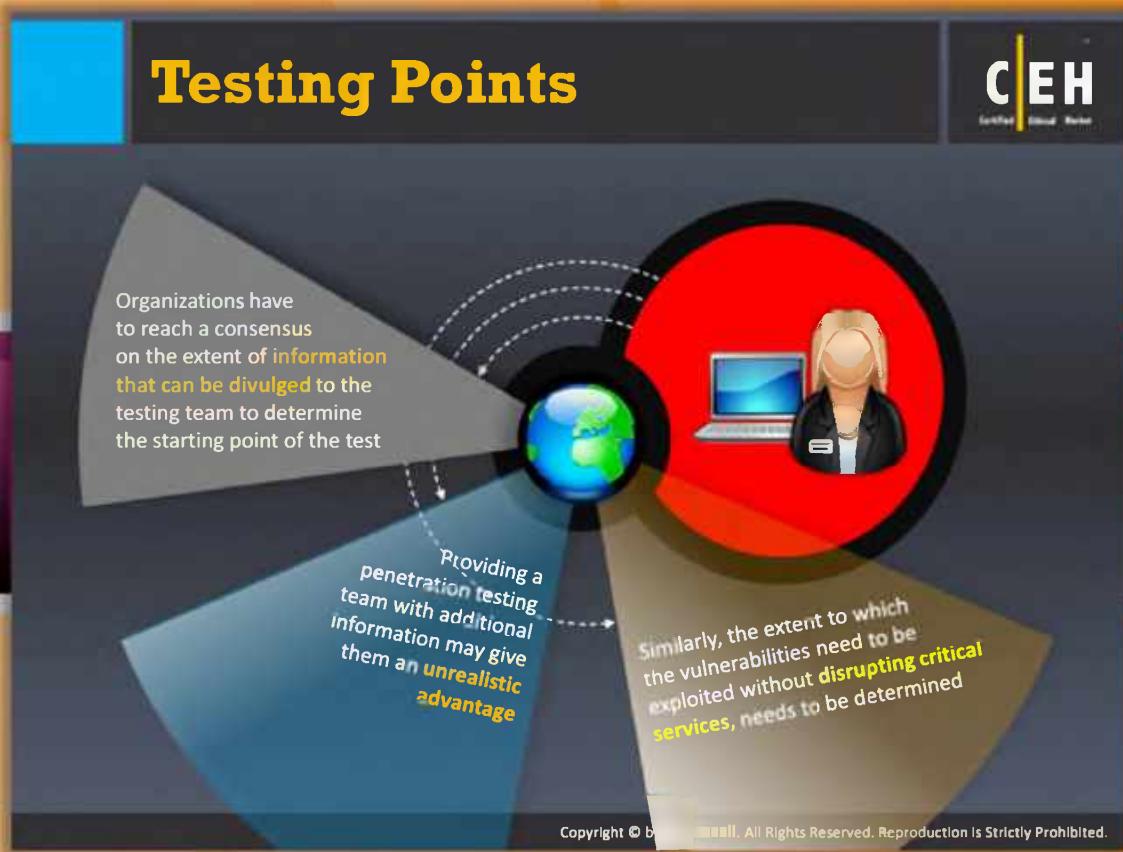
- ⌚ **Payback period:** In this method the time taken to get the pay back (getting the amount invested) on a particular project is calculated.
- ⌚ **Net present value:** Future benefits are calculated in the terms of today's money.
- ⌚ **Internal rate of return:** The benefits based on the interest rate.

So whenever a penetration test is conducted, a company checks what kinds of benefits are there associated with the penetration testing. What could be the costs to be incurred for the for penetration testing? Costs related to the hiring of skilled professionals?

All these things to be kept in view and penetration testing should be conducted through proper planning.

- ⌚ Penetration testing helps companies in identifying, understanding, and addressing vulnerabilities, which saves them a lot of money resulting in ROI.

- Demonstrate the ROI for a pen test with the help of a **business case scenario**, which includes the expenditure and the profits involved in it.



Testing Points

Every penetration test will have a start- and end-point, irrespective of whether it is zero knowledge or partial knowledge test. How does a pen test team or an organization determine this? While providing a **penetration-testing** team with information such as the exact configuration of the firewall used by the target network may speed up the testing, it can work negatively by providing the testers with an unrealistic advantage.

If the objective of the penetration effort is to find as much vulnerability as possible, it might be a good idea to opt for white box testing and share as much information as possible with the testers. This can help in detecting hidden vulnerabilities that are often undetected because of obscurity. On the other hand, if the purpose of the penetration test is to evaluate the effectiveness of the security posture of the organization—irrespective of any “**security by obscurity**” measures—withholding information will derive more realistic results.

Similarly, by making highly sensitive information, such as the names and user IDs of system administrators, the organization may be defeating the purpose of a comprehensive pen test. Therefore, balance must be reached between assisting the testing team in conducting their test faster and providing a more realistic testing environment by restricting information.

Some organizations may choose to get the initial **pen test** audited by a second pen test team so that there is a third party assurance on the results obtained.

Testing Locations

The pentest team may have a choice of doing the test either remotely or on-site

- A remote assessment may simulate an external hacker attack. However, it may miss assessing internal guards
- An on-site assessment may be expensive and may not simulate an external threat exactly

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



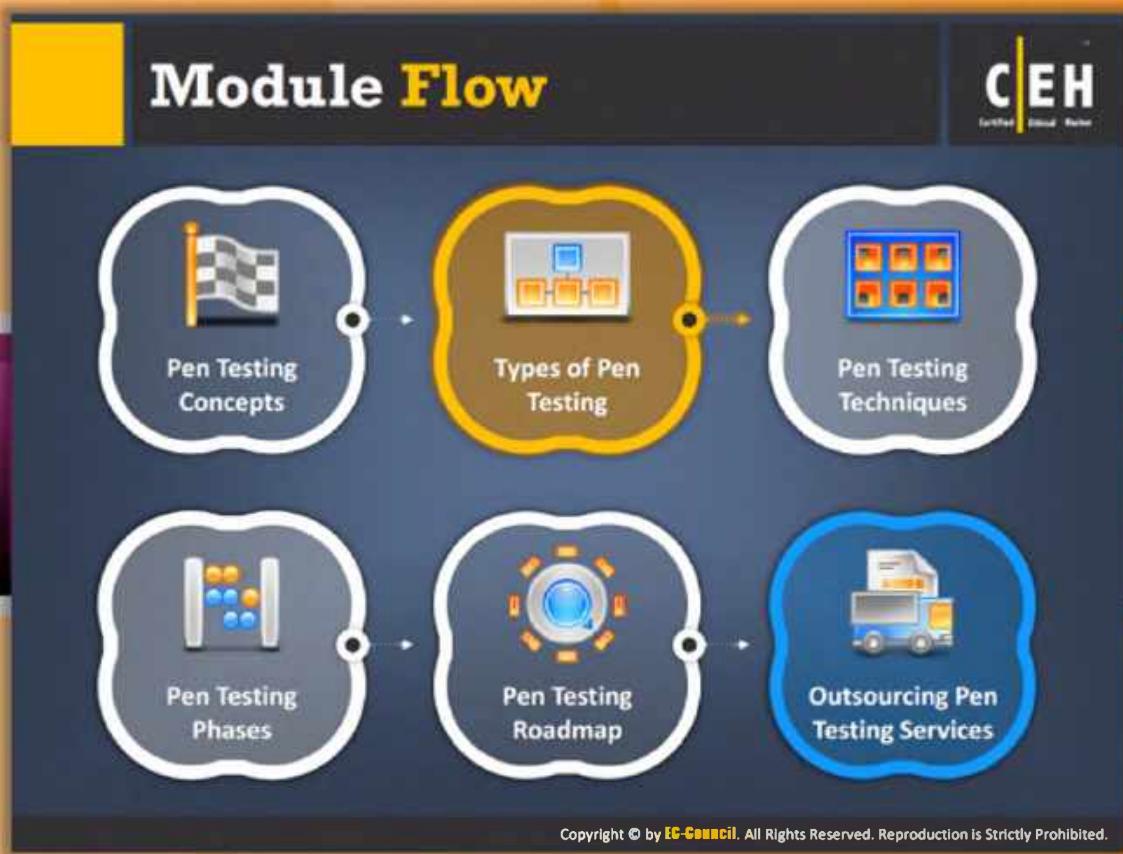
Testing Locations

The penetration test team may have a preference on the location from where they would probe the network. Alternatively, the organization may want the network to be assessed from a remote location. If the pen test team is based overseas, an **onsite assessment** may be expensive than a remote one.

The location of the assessment has an influence on the test results. Testing over the Internet may provide a more realistic test environment. However, the pen test team may learn little if there is a well-configured perimeter firewall and robust web application defenses. A purely external assessment may not be able to test any additional inner network defenses put in place to guard against an internal intruder.

Sometimes, the organization may have a network that is dispersed geographically across locations and that contains several systems. In this case, the organization may choose to prioritize locations or the team may choose locations depending on **critical applications**.

If a complete knowledge test is being undertaken, the pen test team can undertake an asset audit to determine which systems are critical to the business, and plan the test accordingly.



Module Flow

So far, we have discussed various pen testing concepts. Depending on the scope of operation and time required for conducting a pen test, the tester can choose the appropriate type of penetration testing. The selection of the particular type of penetration testing depends upon the type of resources to be protected against attacks. Now, we will discuss various types of pen testing.

Pen Testing Concepts	Types of Pen Testing
Pen Testing Techniques	Pen Testing Phases
Pen Testing Roadmap	Outsourcing Pen Testing Services

In this section, you will learn different types of penetration testing such as external testing, internal testing, Black-box, gray-box penetration testing, white-box penetration testing, announced/unannounced testing, automated testing, and manual testing.

Types of Penetration Testing

External Testing

External testing involves analysis of **publicly available information**, a network enumeration phase, and the behavior of the security devices analyzed



Internal Testing

Internal testing involves testing **computers and devices** within the company

- ⊕ Black-hat testing/zero-knowledge testing
- ⊕ Gray-hat testing/partial-knowledge testing
- ⊕ White-hat testing/complete-knowledge testing
- ⊕ Announced testing
- ⊕ Unannounced testing

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Types of Penetration Testing

Penetration testing is broadly divided into two types. They are:



External Testing

External penetration testing is the conventional approach to penetration testing. The testing is focused on the servers, infrastructure, and underlying software pertaining to the target. It may be performed with no prior knowledge of the site (black box) or with full disclosure of the topology and environment (white box). This type of testing will take in a comprehensive analysis of publicly available information about the target.



Internal Testing

Internal testing makes use of similar methods as the external testing, and it is considered to be a more versatile view of the security. Testing will be performed from several network access points, including both **logical** and **physical** segments.

It is critical to note that despite everything, information security is an ongoing process and penetration testing only gives a snapshot of the security posture of an organization at any given point in time.

Internal testing will be performed from a number of network access points, representing each logical and physical segment. The following tests come fall under **internal testing**:

- ⌚ Black-hat testing/zero-knowledge testing
- ⌚ Gray-hat testing/partial-knowledge testing
- ⌚ White-hat testing/complete-knowledge testing
- ⌚ Announced testing
- ⌚ Unannounced testing

External Penetration Testing

CEH
Certified Ethical Hacker

- External penetration testing involves a **comprehensive analysis** of company's externally visible servers or devices, such as:
 - Web Servers (Icon: Phone)
 - Mail Servers (Icon: Mailbox)
 - Firewalls (Icon: Firewall)
 - Routers (Icon: Router)
- It is the **traditional** approach to penetration testing
- It can be performed without **prior knowledge** of the target to be tested or with full disclosure of the target's **topology** and **environment**
- The goal of an external penetration testing is to demonstrate the **existence of known vulnerabilities** that could be exploited by an external attacker
- It helps the testers to check if system is **properly managed** and **kept up-to-date** protecting the business from information loss and disclosure

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



External Penetration Testing

A pen tester conducts external penetration test for determining the external threats to the network or system. The attacker can perform an external attack without accessing a system by using credentials or the appropriate rights. The main aim behind conducting this pen test is to **identify potential weaknesses** in the security of target network system.

External testing is focused on the servers, infrastructure, and underlying software pertaining to the target. It may be performed with no prior knowledge of the site (black box) or with full disclosure of the topology and environment (white box).

This type of testing will take in a comprehensive analysis of publicly available information about the target, a network enumeration phase where target hosts are identified and analyzed, and the behavior of security devices such as **screening network-filtering devices**. Vulnerabilities are then identified and verified, and the implications assessed. It is the traditional approach to penetration testing.

Internal Security Assessment



Internal penetration testing focuses on company's **internal resources such as DMZs, network connections, application services, etc. and comprehensive analysis of **threats and risks** that arise within the company**

The goal of internal penetration testing is to demonstrate the **exposure of information or other **organization assets** to an unauthorized user**

An internal security assessment follows a similar methodology to external testing, but provides a more **complete view of the site security**

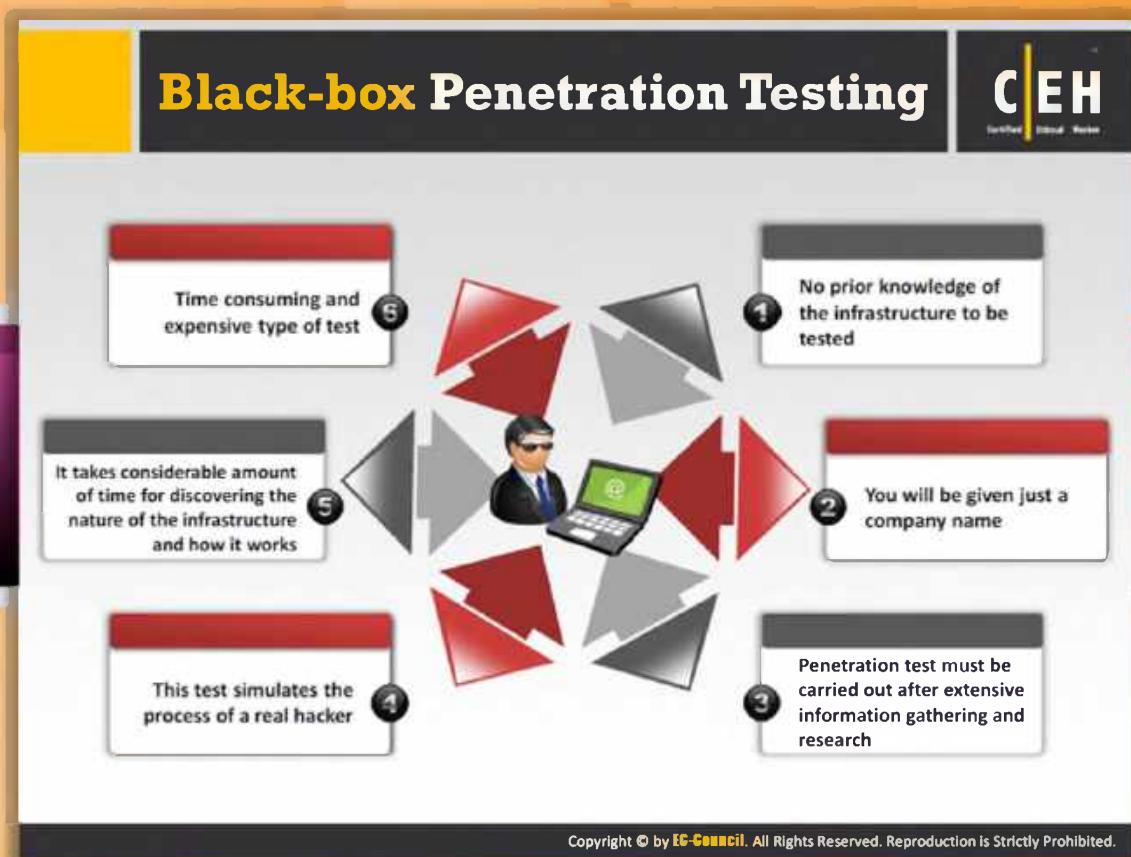
Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Internal Security Assessment

A pen tester conducts internal penetration testing in order to ensure nobody can access the system inside network by misusing user privileges. It is used to identify the weaknesses of computer system inside the particular network. The internal security assessment gives a clear view of the site's security. Internal security assessment has similar methodology like external penetration testing. The main purpose behind the **internal penetration testing** is to find out the various vulnerabilities inside the network. Risks associated with security aspects are carefully checked. Exploitation can be done by a hacker, a malicious employee, etc.:

- ➊ Testing will be performed from a number of network access points, representing each logical and physical segment.
- ➋ For example, this may include tiers and DMZs within the environment, the corporate network, or partner company connections.

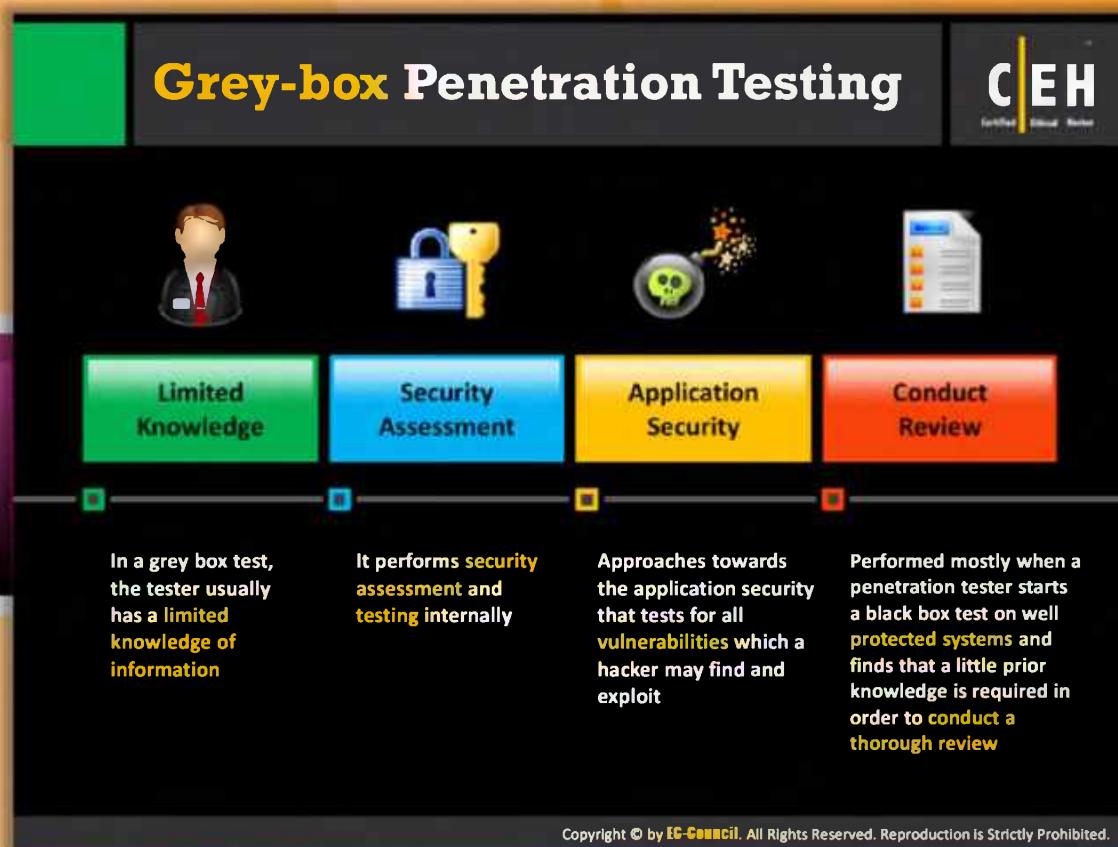


Black-box Penetration Testing

In black-box testing, a pen tester carries out the test without having any prior knowledge of the target. In order to simulate **real-world attacks** and minimize false positives, pen testers can choose to undertake black-hat testing (or a zero-knowledge attack, with no information or assistance from the client) and map the network while enumerating services, shared file systems and operating systems discreetly. Additionally, the pen tester can undertake war dialing to detect listening modems and war driving to discover vulnerable access points if it is legal and within the scope of the project.

The following points summarize the black-box pen testing:

- ➊ It does not require prior knowledge of the infrastructure to be tested
- ➋ Penetration test must be carried out after extensive information gathering and research
- ➌ It takes a considerable amount of time for the project to discover the nature of the **infrastructure** and how it connects and **interrelates**
- ➍ You will be given only a company name
- ➎ This test simulates the process of a real hacker
- ➏ Time consuming and expensive type of test



Gray-box Penetration Testing

In gray-box penetration testing, the test is conducted with limited knowledge about infrastructure, defense mechanism, and communication channels of the target on which test is to be conducted. It is simulation of those attacks that is performed by the insider or outsider with **limited access privileges**.

In this case, organizations would prefer to provide the pen testers with partial knowledge or information that hackers could find such as domain name server. This can save time and expenses of the organization. In gray-box testing, pen testers may also interact with system and network administrators.

White-box Penetration Testing

 Complete knowledge of the **infrastructure** that needs to be tested is known
This test simulates the process of **company's employees**

Information is provided such as

- Company infrastructure
- Network type
- Current security implementations
- IP address / firewall / IDS details
- Company policies do's and don'ts

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



White-box Penetration Testing

In white-box penetration testing, the test is conducted with full knowledge of infrastructure, defense mechanism, and communication channels of the target on which test is being conducted. This test simulates the insider attacker who has full privileges and unlimited access to the **target system**.

This type of penetration test is being conducted when the organization needs to assess its security against a specific kind of attack or a specific target. In this case, the complete information about the target is given to the pen testers. The information provided can include network topology documents, asset inventory, and valuation information. Typically, an organization would opt for this when it wants a complete **audit** of its security.

Announced/Unannounced Testing

CEH
Certified Ethical Hacker

Announced Testing

- Is an attempt to compromise systems on the client with the full **cooperation and knowledge** of the IT staff
- Examines the **existing security** infrastructure for possible vulnerabilities
- Involves the security staff on the penetration testing teams to **conduct audits**



Unannounced Testing

- Is an attempt to compromise systems on the client networks **without the knowledge** of IT security personnel
- Allows only the **upper management** to be aware of these tests
- Examines the security **infrastructure** and **responsiveness** of the IT staff



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Announced/Unannounced Testing

Announced testing is an attempt to access and retrieve pre-identified flag file(s) or to compromise systems on the client network with the full cooperation and knowledge of the IT staff. Such testing examines the existing security infrastructure and individual systems for possible vulnerabilities. Creating a **team-oriented environment** in which members of the organization's security staff are part of the penetration team allows for a targeted attack against the most worthwhile hosts.

Unannounced testing is an attempt to access and **retrieve pre-identified flag file(s)** or to compromise systems on the client network with the awareness of only the upper levels of management. Such testing examines both the existing security infrastructure and the responsiveness of the staff. If intrusion detection and incident response plans have been created, this type of test will identify any weaknesses in their execution. Unannounced testing offers a test of the organization's security procedures in addition to the security of the infrastructure.

In both cases, the IT representative in the organization who would normally report security breaches to legal authorities should be aware of the test to prevent escalation to **law enforcement organizations**.

Automated Testing

CEH
Certified Ethical Hacker

Time and Cost Savings

Automated testing can result in **time and cost savings** over a long term; however, it cannot replace an experienced security professional



Tools

Tools can have a **high learning curve** and may need frequent updating to be effective



Scope

With automated testing, there exists **no scope for any of the architectural elements** to be tested



Scanners

As with vulnerability scanners, there can be **false negatives or worse, false positives**



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Automated Testing

Instead of relying on security experts, some organizations and security-testing firms prefer to automate their security assessments. Here, a security tool is run against the target and the security posture is assessed. The tools attempt to replicate the attacks that intruders have been known to use. This is similar to vulnerability scanning. Based on the success or failure of these attacks, the tool attempts to assess and report security vulnerabilities.

However, it must be noted that a thorough security assessment also includes elements of **architectural review, security policy, firewall rule-base analysis, application testing, and general benchmarking**. Automated testing is generally limited to external penetration testing using the black-box approach and does not allow an organization to profit completely from the exercise. As an automated process, there is no scope for any of the policy or architectural elements in the testing, and it may need to be supplemented by a security professional's expertise.

One advantage attributed to automated testing is that it reduces the volume of traffic required for each test. This gives an impression that the organization can service its customers concurrently for the same overhead structure. Organizations need to evaluate if this indeed serves the purpose of the test. A **non-automated security assessment** will always be more flexible to an organization's requirements and more cost effective, as it will take into account other areas such as security architecture and policy, and will most likely be more thorough and therefore secure. In addition, testing at frequent intervals allows the consultants to explain to

the management of the organization and the technical audiences what they have discovered, the processes they used, and the **ramifications** of all the **recommendations**. Additionally, they can inform in person, as an individual entity helping to support the IT security department augmenting the budgets required.

Manual Testing

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

- Manual testing is the best option an organization can choose to benefit from the experience of a **security professional**.
- The objective of the professional is to assess the **security posture of the organization** from an attacker's perspective.
- A manual approach requires **planning, test designing, scheduling, and diligent documentation** to capture the results of the testing process.



Manual Testing

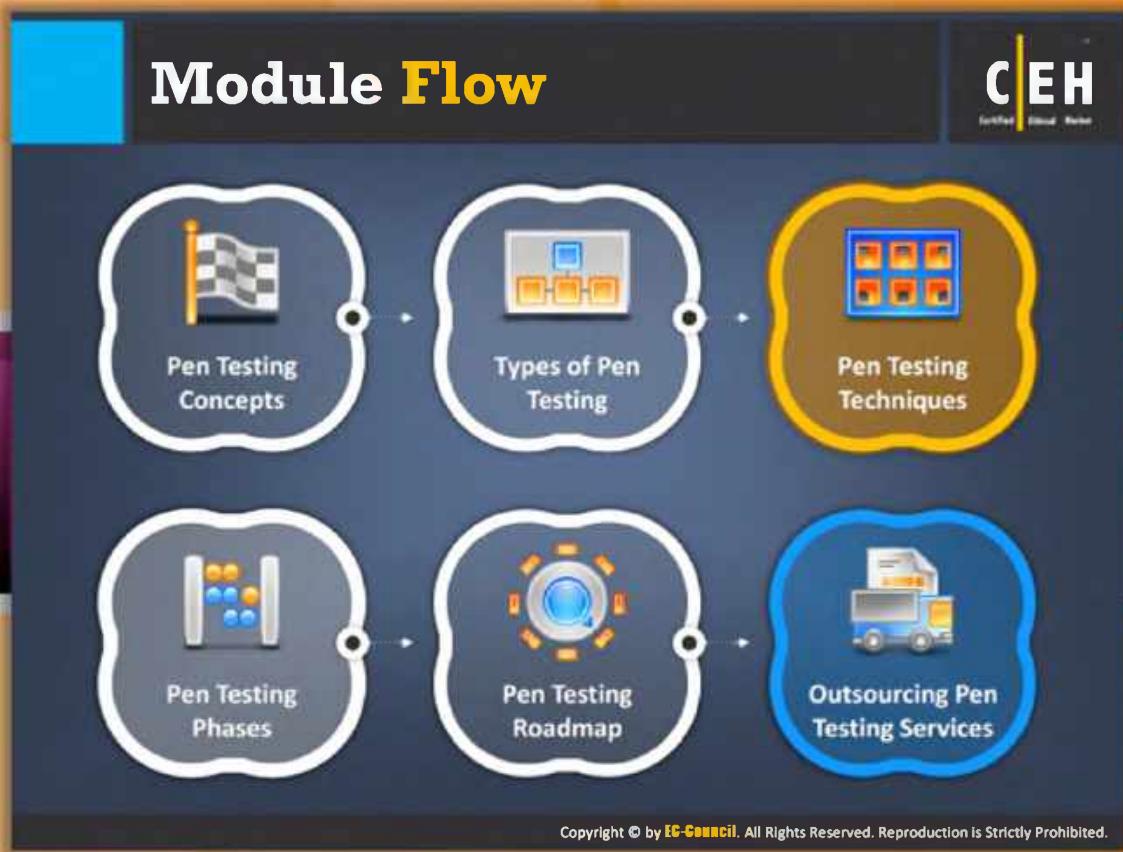
Several organizations choose to have a manual assessment of their security and benefit from the experience of a seasoned security professional. The objective of the professional is to assess the security posture of the organization from an attacker's perspective.

Under the manual approach, the security professional attempts to unearth holes in the security model of the organization by approaching it in a methodical manner. The phases of testing can involve **basic information gathering, social engineering, scanning, vulnerability assessment, exploiting vulnerabilities**, etc.

A manual approach requires planning, test designing and scheduling, and diligent documentation to capture the results of the testing process in its entirety. Documentation plays a significant role in deciding how well the team has been able to assess the security posture of the organization.

Some organizations may choose to have their own internal team to do the manual assessment and an external agency audit at the same time. Some others may choose to get a second external team to audit the findings of the **first external team**.

The rules of engagement and the expected deliverables should be clearly defined. In the long term, the management will benefit more from a manual approach as the team would be able to explain the gravity of the situation from an unbiased viewpoint and make recommendations on improving the security posture.



Module Flow

Considering that you became familiar with pen testing concepts and the types of penetration testing, we will move forward to penetration testing techniques.

This section covers various penetration testing techniques.

Pen Testing Concepts	Types of Pen Testing
Pen Testing Techniques	Pen Testing Phases
Pen Testing Roadmap	Outsourcing Pen Testing Services

Common Penetration Testing Techniques

C|EH
Certified Ethical Hacker

Passive Research	Is used to gather all the information about an organization's system configurations
Open Source Monitoring	Facilitates an organization to take necessary steps to ensure its confidentiality and integrity
Network Mapping and OS Fingerprinting	Is used to get an idea of the network's configuration being tested
Spoofing	Is the act of using one machine to pretend to be another Is used here for both internal and external penetration tests
Network Sniffing	Is used to capture the data as it travels across a network
Trojan Attacks	Are malicious code or programs usually sent into a network as email attachments or transferred via "Instant Message" into chat rooms
A Brute-force Attack	Is the most commonly known password cracking method. Can overload a system and possibly stop it from responding to the legal requests
Vulnerability Scanning	Is a comprehensive examination of the targeted areas of an organization's network infrastructure
A Scenario Analysis	Is the final phase of testing, making a risk assessment of vulnerabilities much more accurate

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Common Penetration Testing Techniques

The following are a few common techniques that can be used for penetration testing:



Passive research

Passive research is used to gather information about an organization related to the configuration from public domain sources such as **DNS records, name registries, ISP looking-glass servers, Usenet newsgroups**, etc.



Open source monitoring

Open source monitoring facilitates an organization to take necessary steps to ensure its confidentiality and integrity. Monitoring includes alerting in the following situations:

- ⌚ When the database is not available
- ⌚ When a database error occurs
- ⌚ The file system is running out of space etc.

Graphing and seeing trends for:

- ⌚ Database

- ⌚ Table locks
- ⌚ Replication lag
- ⌚ Table cache efficiency etc.



Network mapping and OS fingerprinting

Network mapping and OS fingerprinting gives an idea about the configuration of the entire network being tested. This technique is designed to specify different types of services present on the target system.



Spoofing

Spoofing is an attempt by someone or something to masquerade as someone else. For example: one machine pretends to be another. Spoofing is used here for both internal and external penetration tests.



Network sniffing

Network spoofing occurs when the attacker forges the source or destination IP address in the IP header. It is used to capture data as it travels across a network.



Trojan attacks

A Trojan attack is installing a Trojan (malicious software) onto the victim's system. It gets installed through **email**, **CD-ROM**, **Internet Explorer**, etc.



Brute force attacks

Session IDs can be guessed by using the brute force technique. It tries multiple possibilities of patterns until a session ID works. An attacker using a DSL line can make up to 1000 session IDs per second. This technique is used when the algorithm that produces session IDs is not random.



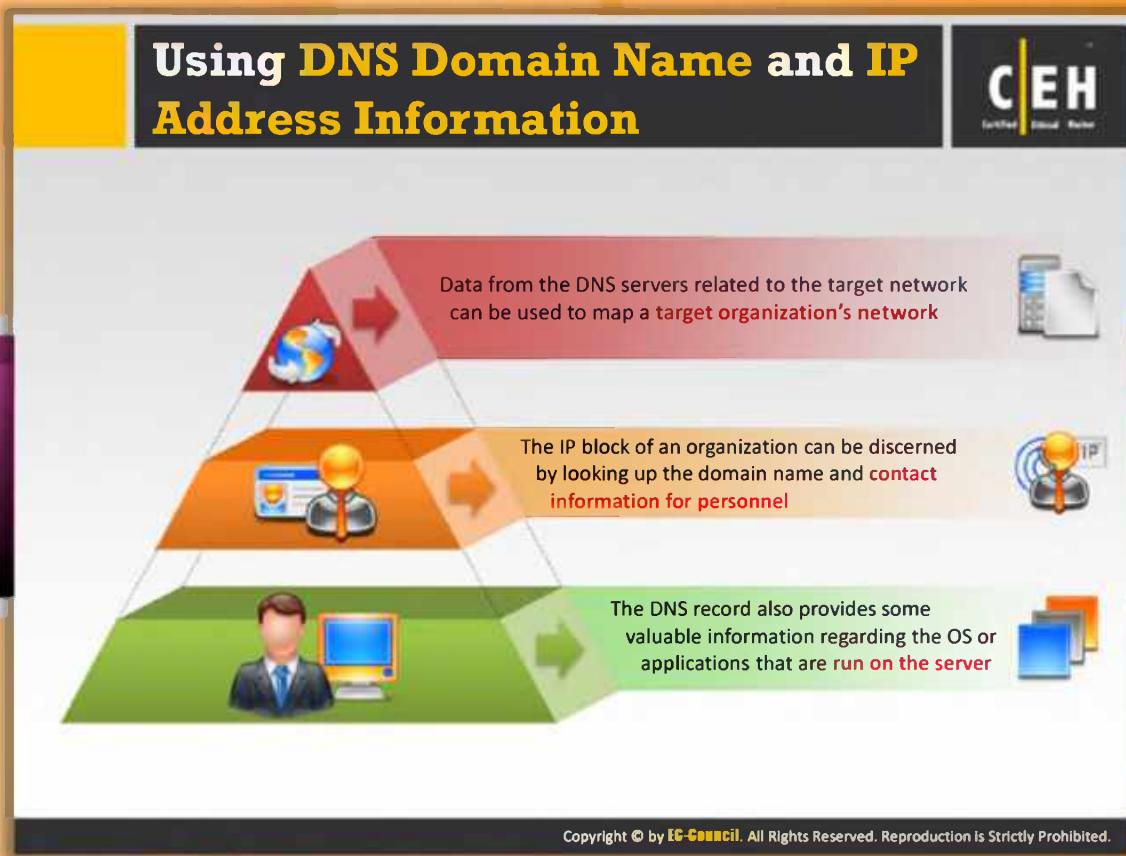
Vulnerability scanning

Vulnerability scanning is used to discover weaknesses in a security system in order to improve or repair before a breach occurs. It is a comprehensive examination of the targeted areas of an organization's network infrastructure



Scenario analysis

Scenario analysis helps in dealing with uncertainties. It is the final phase of testing, making a risk assessment of vulnerabilities much more accurate.

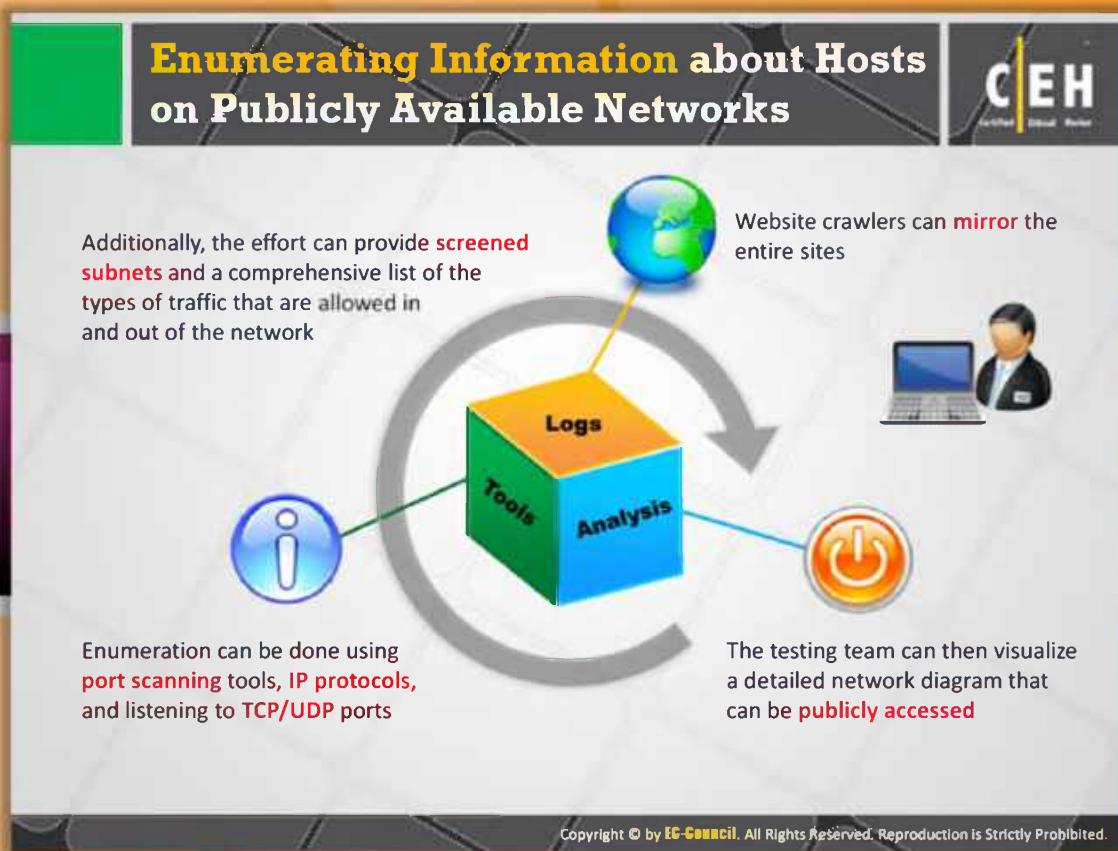


Using DNS Domain Name and IP Address Information

Data from the DNS servers related to the target network can be used to map a target organization's network. DNS zones can be analyzed for information about the target organization's network. This can result in obtaining further data, including the **server host's names, services offered by particular servers, IP addresses**, and contact data for the members of the IT staff.

Many attackers have been known to use software, which is easily available to the general public, to create well-organized network diagrams of the target network. IP address data regarding a particular system can be gained from the DNS zone or the American Registry of Internet Numbers (ARIN). Another way of obtaining an IP address is by using port-scanning software to deduce a **target organization's network diagram**.

By examining the DNS records, you can get a good understanding about where the servers of the target network are located. The DNS record also provides some valuable information regarding the OS or applications that are being run on the server. The IP block of an organization can be discerned by looking up the domain name and contact information for personnel can be obtained.



Enumerating Information about Hosts on Publicly Available Networks

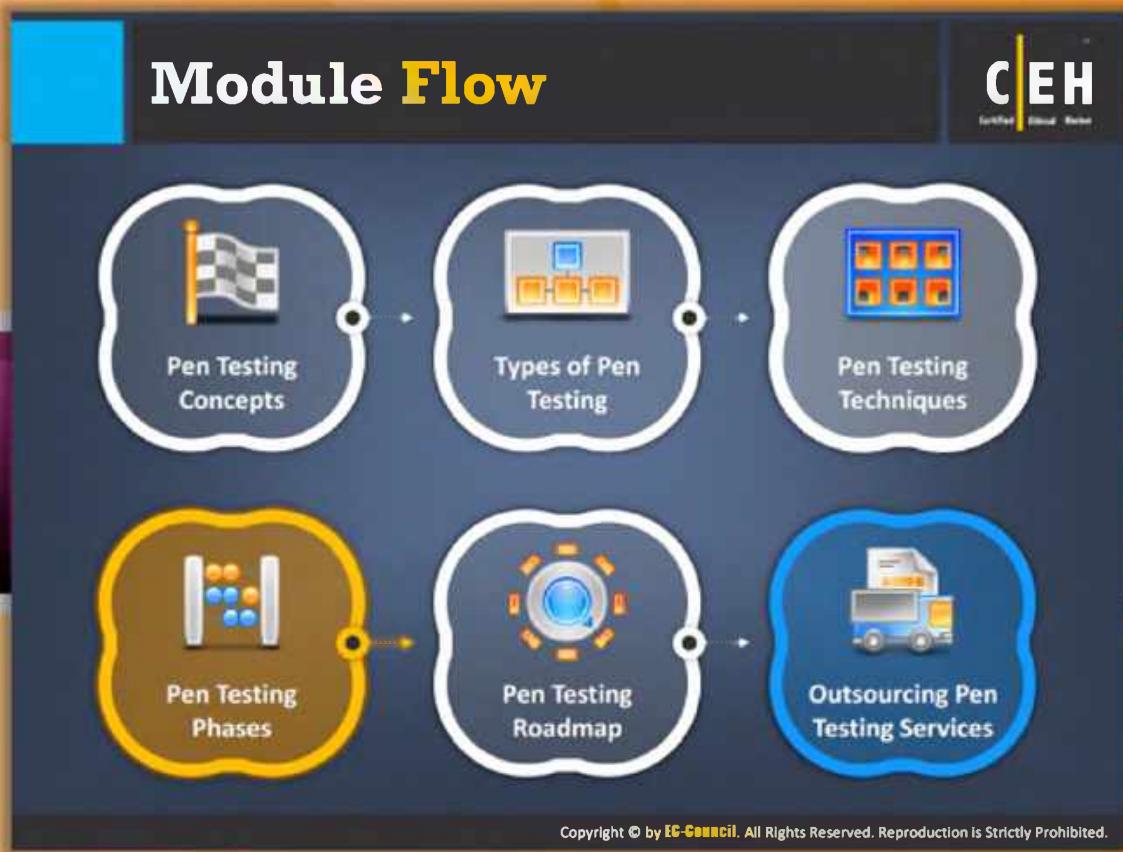
With the IP addresses obtained in the preceding step, the pen-test team can outline the network to explore possible points of entry from the perspective of an attacker. **Testers** achieve this by analyzing all data about the hosts that are uncovered to the Internet by the target organization. They can use port-scanning tools and IP protocols, and they can listen to **TCP/UDP ports**.

Port scans will also reveal information about hosts such as the current operating system that is running on the system and also other applications. An effective port-scanning tool can also help to deduce how the router and firewall IP filters are configured. The testing team can then visualize a detailed network diagram that can be publicly accessed.

Additionally, the effort can provide screened subnets and a comprehensive list of the types of traffic that is allowed in and out of the network. **Website crawlers** can mirror entire sites and allow the testing group to check for faulty source code or inadvertent inclusions of sensitive information. Many times, organizations have given information that is not intended for use by the public, but is posted on the website.

- If the rules of engagement permit, the **pen-test** team may purchase research reports on the organization available for sale and use the information available therein for

comprising the security of the target organization. These can include covert means, such as social engineering, as well. It is necessary to point out that prior approval from management is a critical aspect to be considered before indulging in such activities.



Module Flow

Pen testing is the test conducted in three phases for discovering the vulnerabilities or weakness in an organization's systems. The three phases are the **pre-attack phase, attack phase, and post-attack phase**.

Pen Testing Concepts	Types of Pen Testing
Pen Testing Techniques	Pen Testing Phases
Pen Testing Roadmap	Outsourcing Pen Testing Services

This section highlights the three phases of pen testing.



Phases of Penetration Testing

These are three phases of penetration testing.



Pre-attack Phase

This phase is focused on gathering as much information as possible about the target organization or network to be attacked. This can be **non-invasive** or **invasive**.



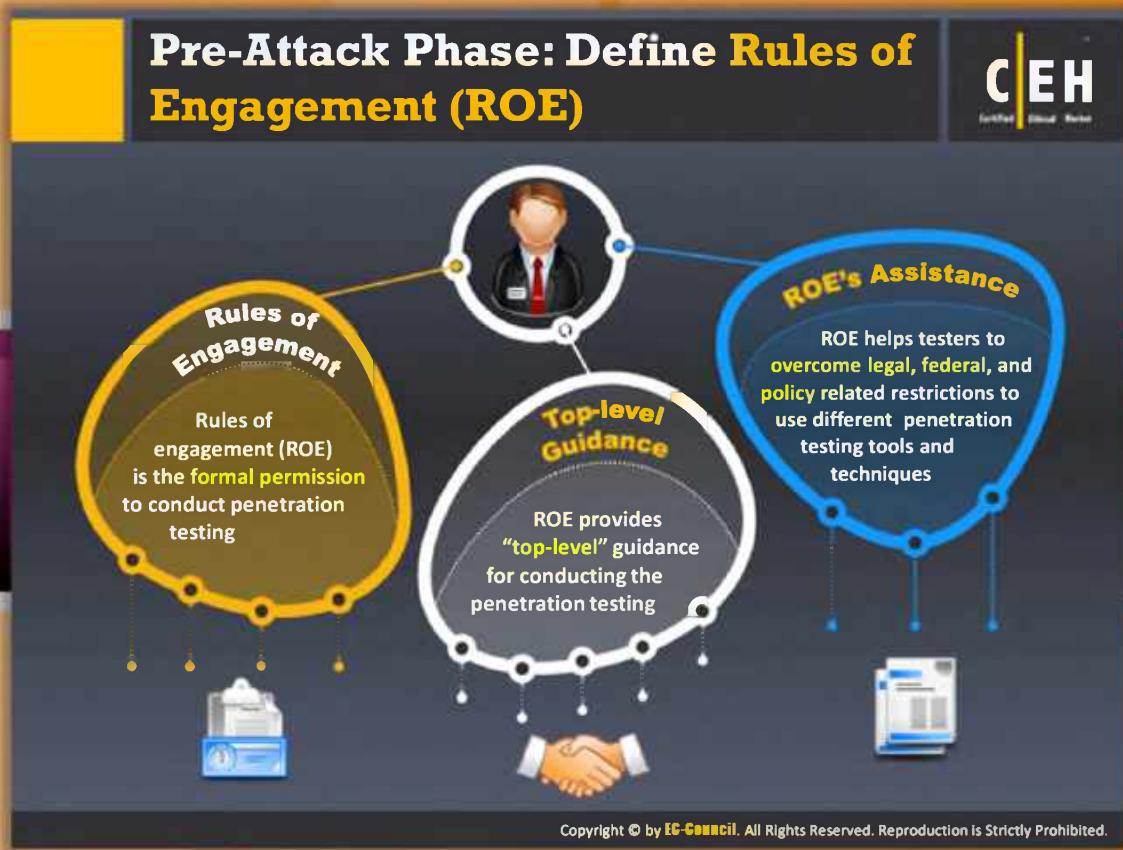
Attack Phase

The information gathered in the pre-attack phase forms the basis of the attack strategy. Before deciding the attack strategy, the tester may choose to carry out an invasive information gathering process such as scanning.



Post-attack Phase

This is a crucial part of the testing process, as the tester needs to restore the network to its original state. This involves cleanup of testing processes and removal of vulnerabilities created (not those that existed originally), **exploits crafted**, etc.



Pre-attack Phase: Define Rules of Engagement (ROE)

Rules of engagement (ROE) are the guidelines and constraints about the execution of penetration testing. It should be developed and presented before conducting the penetration test. It gives authority to the pen tester to conduct defined activities without the need for additional permissions. ROE helps pen testers to overcome **legal-, federal-, and policy-related restrictions** to use different penetration testing tools and techniques

Pre-Attack Phase: Understand Customer Requirements

Before proceeding with the penetration testing, a pen tester should identify what needs to be tested

Procedure

- ☛ Create a checklist of testing requirements
- ☛ Identify the time frame and testing hours
- ☛ Identify who will be involved in the reporting and document delivery



Items to be Tested

	Yes <input type="checkbox"/>	No <input type="checkbox"/>
Servers	<input type="checkbox"/>	<input type="checkbox"/>
Workstations	<input type="checkbox"/>	<input type="checkbox"/>
Routers	<input type="checkbox"/>	<input type="checkbox"/>
Firewalls	<input type="checkbox"/>	<input type="checkbox"/>
Networking devices	<input type="checkbox"/>	<input type="checkbox"/>
Cabling	<input type="checkbox"/>	<input type="checkbox"/>
Databases	<input type="checkbox"/>	<input type="checkbox"/>
Applications	<input type="checkbox"/>	<input type="checkbox"/>
Physical security	<input type="checkbox"/>	<input type="checkbox"/>
Telecommunications	<input type="checkbox"/>	<input type="checkbox"/>

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Pre-attack Phase: Understand Customer Requirements

Once ROE is defined to conduct penetration test, the second step in the pre-attack phase, you should clearly understand the customer requirements, i.e., what the customer expects from the penetration test. Before proceeding with the **penetration testing**, a pen tester should identify what needs to be tested in the target organization.

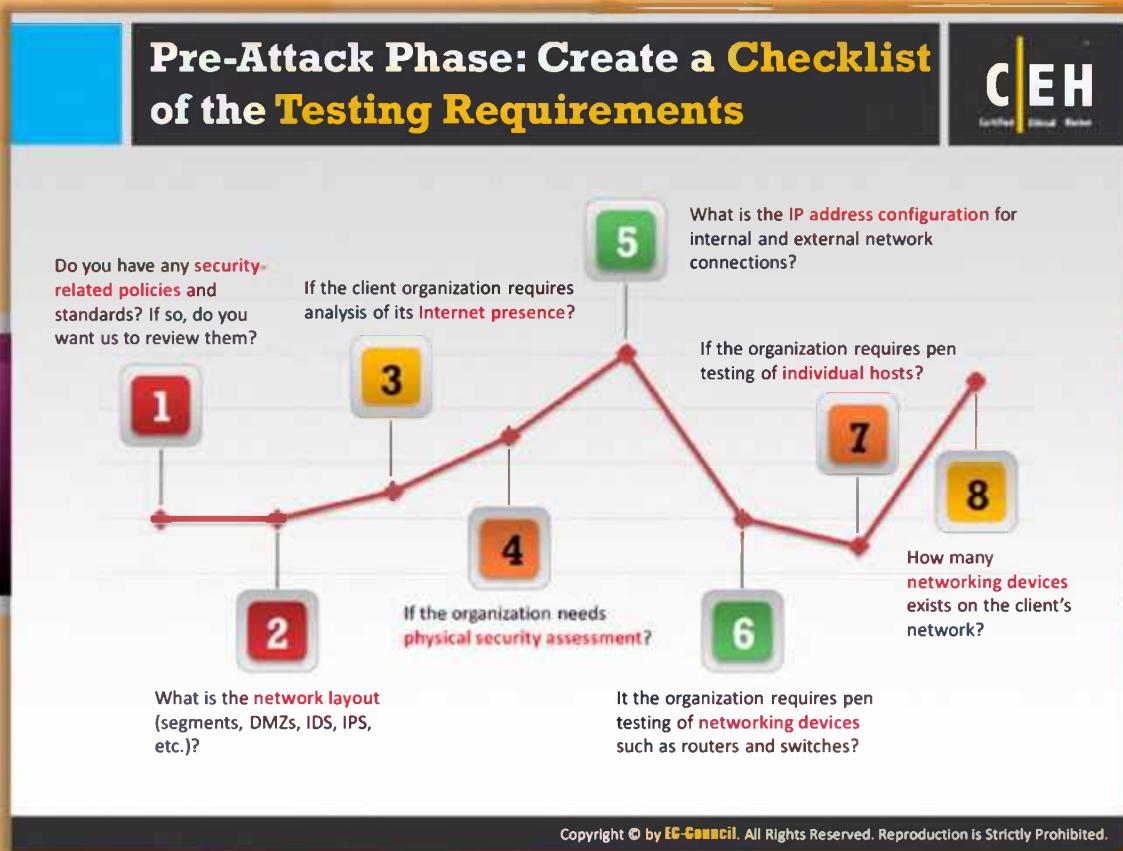
To clearly identify the customer requirements, do the following things:

- ☛ Create a checklist of testing requirements
- ☛ Identify the time frame and testing hours
- ☛ Identify who will be involved in the reporting and document delivery

Prepare the check list for the items that need to be tested in **target organization** as shown in following figure:

Items to be Tested			
	Yes <input type="checkbox"/>	No <input type="checkbox"/>	
Servers	Yes <input type="checkbox"/>	No <input type="checkbox"/>	
Workstations	Yes <input type="checkbox"/>	No <input type="checkbox"/>	
Routers	Yes <input type="checkbox"/>	No <input type="checkbox"/>	
Firewalls	Yes <input type="checkbox"/>	No <input type="checkbox"/>	
Networking devices	Yes <input type="checkbox"/>	No <input type="checkbox"/>	
Cabling	Yes <input type="checkbox"/>	No <input type="checkbox"/>	
Databases	Yes <input type="checkbox"/>	No <input type="checkbox"/>	
Applications	Yes <input type="checkbox"/>	No <input type="checkbox"/>	
Physical security	Yes <input type="checkbox"/>	No <input type="checkbox"/>	
Telecommunications	Yes <input type="checkbox"/>	No <input type="checkbox"/>	

FIGURE 20.1: Check list of the items that need to be tested



Pre-attack Phase: Create a Checklist of the Testing Requirements

To collect the **penetration test** requirements from the customer, ask the customer the following questions. The answers of these questions will help you to define the scope of the test.

- ❑ Do you have any security-related policies and standards? If so, do you want us to review them?
- ❑ What is the network layout (segments, DMZs, IDS, IPS, etc.)?
- ❑ If the client organization requires analysis of its Internet presence?
- ❑ If the organization needs **physical security assessment**?
- ❑ What is the IP address configuration for internal and external network connections?
- ❑ If the organization requires pen testing of networking devices such as routers and switches?
- ❑ If the organization requires pen testing of individual hosts?
- ❑ How many networking devices exists on the client's network?

Pre-Attack Phase: Create a Checklist of the Testing Requirements (Cont'd)



What security controls are deployed across the organization?

If the organization requires assessment of wireless networks?

If the organization requires assessment of analog devices in the network?

9 

10 

11 

12 

13 

14 

15 

If the organization deploy a mobile workforce? If so, if the mobile security assessment is required?

What workstation and server operating systems are deployed across the organization?

If the organization requires the assessment of web infrastructure?

What are the web application and services offered by the client?

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Pre-attack Phase: Create a Checklist of the Testing Requirements (Cont'd)

The following are a few more questions that you should ask the customer to complete the checklist of penetration testing requirements:

- ❑ What security controls are deployed across the organization?
- ❑ If the organization requires assessment of wireless networks?
- ❑ If the organization requires assessment of analog devices in the network?
- ❑ If the organization deploy a mobile workforce? If so, if the mobile security assessment is required?
- ❑ What are the web application and services offered by the client?
- ❑ If the organization requires the assessment of web infrastructure?
- ❑ What workstation and server operating systems are deployed across the organization?

Pre-Attack Phase: Define the Pen-Testing Scope

CEH
Certified Ethical Hacker

- Pen testing scope defines **what to test** and **how to test**
- Pen testing test components depends on the **client's operating environment, threat perception, security and compliance requirement, ROE and budget**

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Pre-attack Phase: Define the Pen-testing Scope

You should define the scope of your penetration test explicitly and in writing. This will help you to identify what needs to be tested in the target organization, and help to develop the procedure to test particular component once identified. This also help you to identify limitations, i.e., what should not be tested. Pen testing test components depend on **the client's operating environment, threat perception, security and compliance requirements, ROE, and budget**. The following are the possible areas of the scope of the penetration test:

- Network Security
- System Software Security
- Client-side Application Security
- Server-side Application Security
- Social Engineering
- Application Communication Security
- Physical Security
- Dumpster Diving
- Inside Accomplices
- Sabotage Intruder Confusion
- Intrusion Detection
- Intrusion Response

Pre-Attack Phase: Sign Penetration Testing Contract

The penetration testing contract must be **drafted by a lawyer** and **signed** by the penetration tester and the company.

The contract must clearly state the following:

```
graph TD; A[Objective of the penetration test] --- B[2]; B --- C[4]; C --- D[6]; D --- E[7]; E --- F[Non-disclosure clause]; F --- G[Fees and project schedule]; G --- H[Confidential information]; H --- I[Reporting and responsibilities];
```

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Pre-attack Phase: Sign Penetration Testing Contract

Once the requirements and scope of the penetration test is confirmed from the client, you need to sign the contract with the company to conduct the penetration test. This contract must be drafted by a **lawyer** and duly signed by the **penetration tester** and the company. The contract should include the following terms and conditions:

- ❑ Non-disclosure clause
- ❑ Objective of the penetration test
- ❑ Fees and project schedule
- ❑ Sensitive information
- ❑ Confidential information
- ❑ Indemnification clause
- ❑ Reporting and responsibilities

Pre-Attack Phase: Sign Confidentiality and Non-Disclosure (NDA) Agreements

Pen testers should sign **Confidentiality and Non-Disclosure (NDA) Agreements** that guarantees that the company's information will be treated confidentially

It also protects testers from legal liabilities in the event of some **untoward happening** during pen testing

Many documents and other information regarding pen-test contain critical information that could **damage one or both parties** if improperly disclosed

Agreements are designed to be used by both the parties to **protect sensitive information from disclosure**

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Pre-attack Phase: Sign Confidentiality and Non-Disclosure (NDA) Agreements

As a pen tester, you will also need to sign Confidentiality and Non-Disclosure (NDA) Agreements to maintain the confidentiality of the company's sensitive information. Many documents and other information regarding the pen test contain critical information that could damage one or both parties if disclosed to other parties. Both (pen tester and company) parties should agree and duly signed on the terms and conditions included in the Confidentiality and Non-Disclosure (NDA) Agreements before conducting penetration test.

The following are the advantages of signing **Confidentiality** and **Non-Disclosure (NDA)** Agreements:

- They ensure that the company's information will be treated confidentially.
- They will also help to provide cover for a number of other key areas, such as negligence and liability in the event of something untoward happening.

Pre-Attack Phase: Sign Confidentiality and Non-Disclosure (NDA) Agreements (Cont'd)

Agreement

- Both parties bear responsibility to **protect tools, techniques, vulnerabilities, and information** from disclosure beyond the terms specified by a written agreement

Protect

- Non-disclosure agreements should be **narrowly drawn** to protect sensitive information

Areas

- Specific areas to consider include:
 - Ownership
 - Use of the evaluation reports
 - Results; use of the **testing methodology** in customer documentation

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Pre-attack Phase: Sign Confidentiality and Non-Disclosure (NDA) Agreements (Cont'd)

The Confidentiality and Non-Disclosure agreements document is a powerful tool. Once you sign the NDA agreement, the company has the right to file a lawsuit against you even if you disclose the information to third party either intentionally or unintentionally. The following points should be considered while crafting Confidentiality and Non-Disclosure (NDA) Agreements:

- Both parties should bear responsibility to **protect tools, techniques, vulnerabilities, and information** from disclosure beyond the terms specified by a written agreement
- Non-disclosure agreements should be narrowly drawn to protect sensitive information.
- Specific areas to consider include:
 - Ownership
 - Use of the evaluation reports

Results; use of the testing methodology in customer documentation

Pre-Attack Phase: Information Gathering

CEH
Certified Ethical Hacker

- Pre-attack phase addresses the **mode of the attack** and the goals to be achieved
- Reconnaissance is considered as the first in the pre-attack phase, which attempts to **collect information** about the target
- Hackers try to find out as much **information** as possible about a **target**
- Hackers gather information in different ways that allows them to **formulate a plan of attack**

Types of Reconnaissance

Passive Reconnaissance

Involves collecting information about a target from the publicly accessible sources



Active Reconnaissance

Involves information gathering through social engineering, on-site visits, interviews, and questionnaires



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Pre-attack Phase: Information Gathering

The pre-attack phase addresses the mode of the attack and the goals to be achieved. Reconnaissance is considered as the first in the pre-attack phase and is an attempt to locate, gather, identify, and record information about the target. An attacker seeks to find out as much information as possible about the victim. Attackers gather information in different ways that allows them to formulate a **plan of attack**. There are two types of reconnaissance:



Passive reconnaissance

It comprises the attacker's attempts to scout for or survey potential targets and investigations or explorations of the target. It also includes information gathering and may involve competitive intelligence gathering, social engineering, breaching physical security, etc. Attackers typically spend more time on the pre-attack or reconnaissance activity than the actual attack.

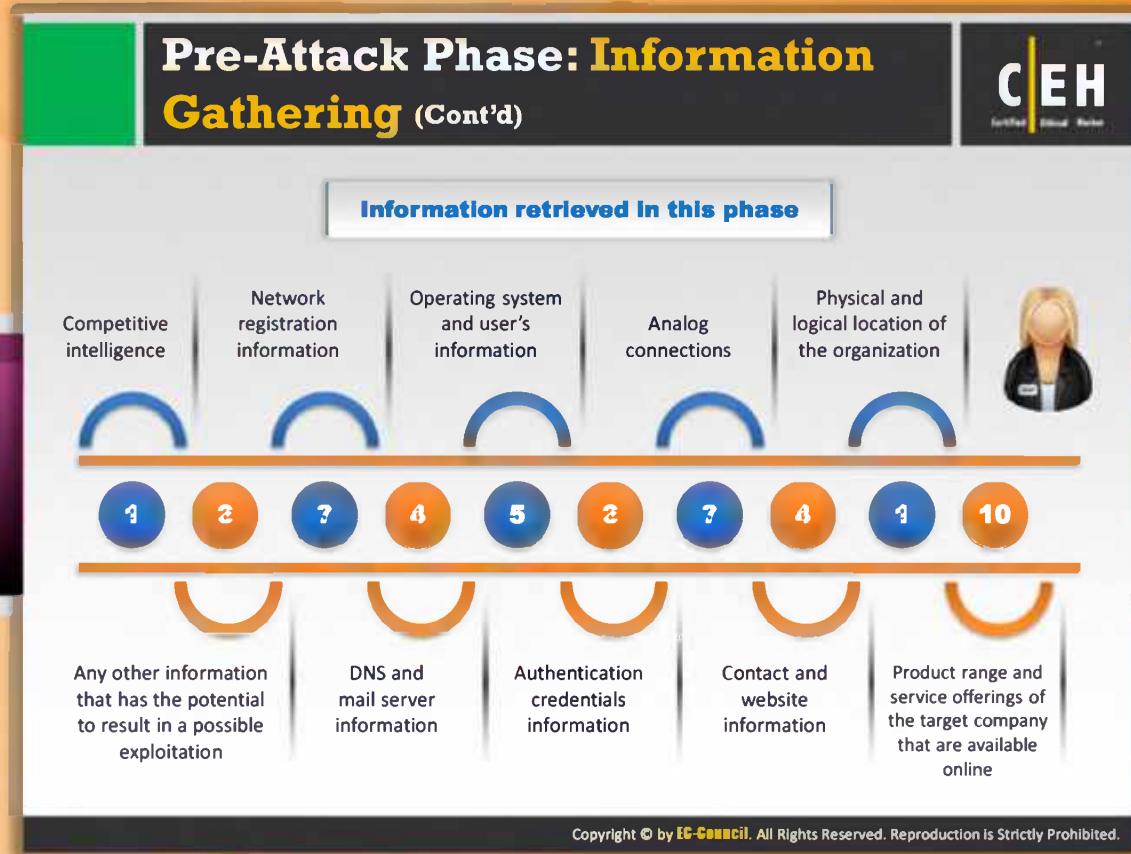
Beginning with passive reconnaissance, the tester gathers as much information as possible about the target company. Much of the leaked information caters to the network topology and the types of services running within. The tester can use this sensitive information to provisionally map out the network for planning a more **coordinated attack strategy** later.

With regard to publicly available information, access to this information is independent of the organization's resources, and can therefore be effectively accessed by anyone. Information is often contained on systems unrelated to the organization.



Active reconnaissance

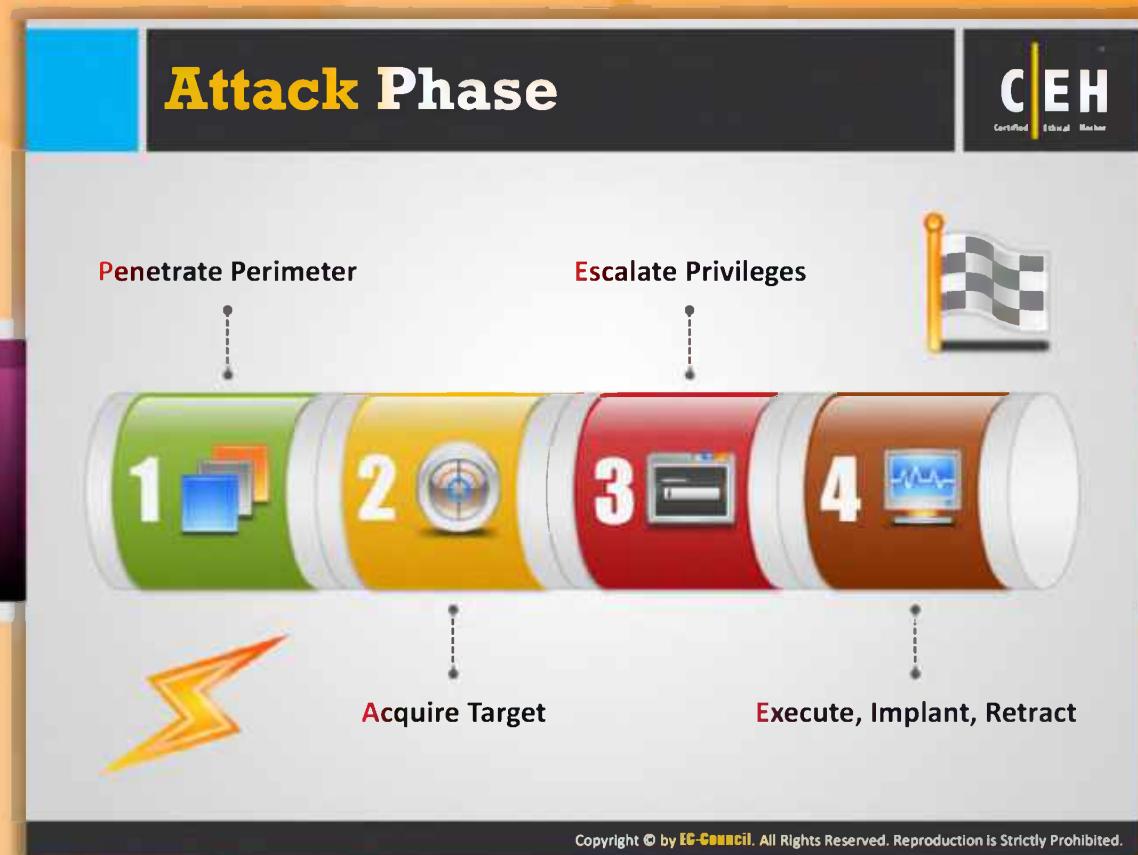
The information gathering process encroaches on the target territory. Here, the perpetrator may send probes to the target in the form of port scans, network sweeps, enumeration of shares and user accounts, etc. The attacker may adopt techniques such as **social engineering**, employing tools such as **scanners** and **sniffers** that automate these tasks. The footprints that the attacker leaves are larger, and novices can be easily identified.



Pre-attack Phase: Information Gathering (Cont'd)

The following information is retrieved during the pre-attack phase:

- Competitive intelligence
- Network registration information
- DNS and mail server information
- Operating system information
- User's information
- Authentication credentials information
- Analog connections
- Contact information
- Website information
- Physical and logical location of the organization
- Product range and service offerings of the target company that are available online
- Any other information that has the potential to result in a possible exploitation

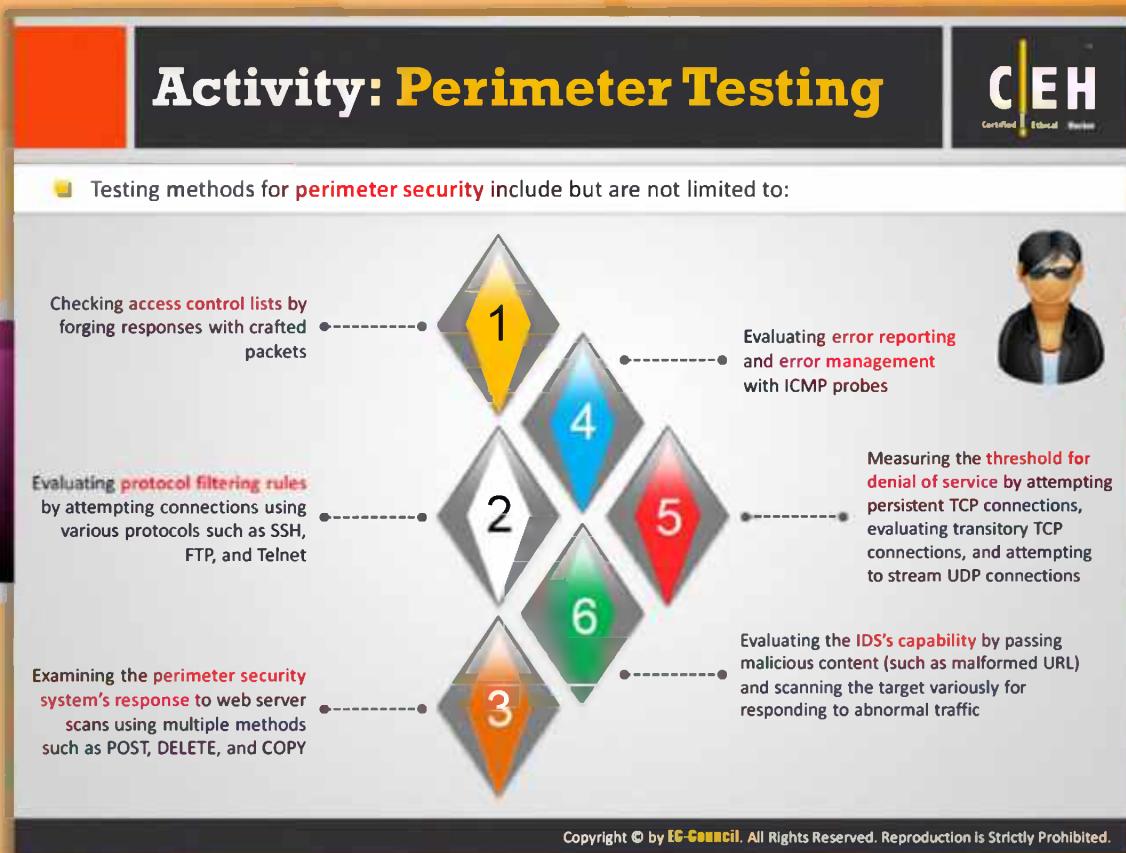


Attack Phase

This stage involves the actual compromise of the target. The attacker may exploit a vulnerability discovered during the pre-attack phase or use **security loopholes** such as a weak security policy to gain rights to the system. The important point here is that the attacker needs only one port of entry, whereas the organizations are left to defend several. Once inside, the attacker may escalate his privileges and install a backdoor so that he or she sustains access to the system and exploits it in order to achieve his/her **malicious** intent.

During the attack phase, the attacker or pen tester needs to:

- ☛ Penetrate perimeter
- ☛ Execute, implant, retract
- ☛ Acquire target
- ☛ Escalate privileges



Activity: Perimeter Testing

Social engineering is an ongoing activity through the testing phase as sensitive information can be acquired at any stage of testing. The tests that can be carried out in this context include (but are not limited to) impersonating or mocking phone calls to capture sensitive information, verifying information gathered through activities such as dumpster diving. Other means include email testing, trusted person acquisition, and attempts to retrieve legitimate authentication details such as passwords and access privileges. Information gathered here can be used later in web application testing also.

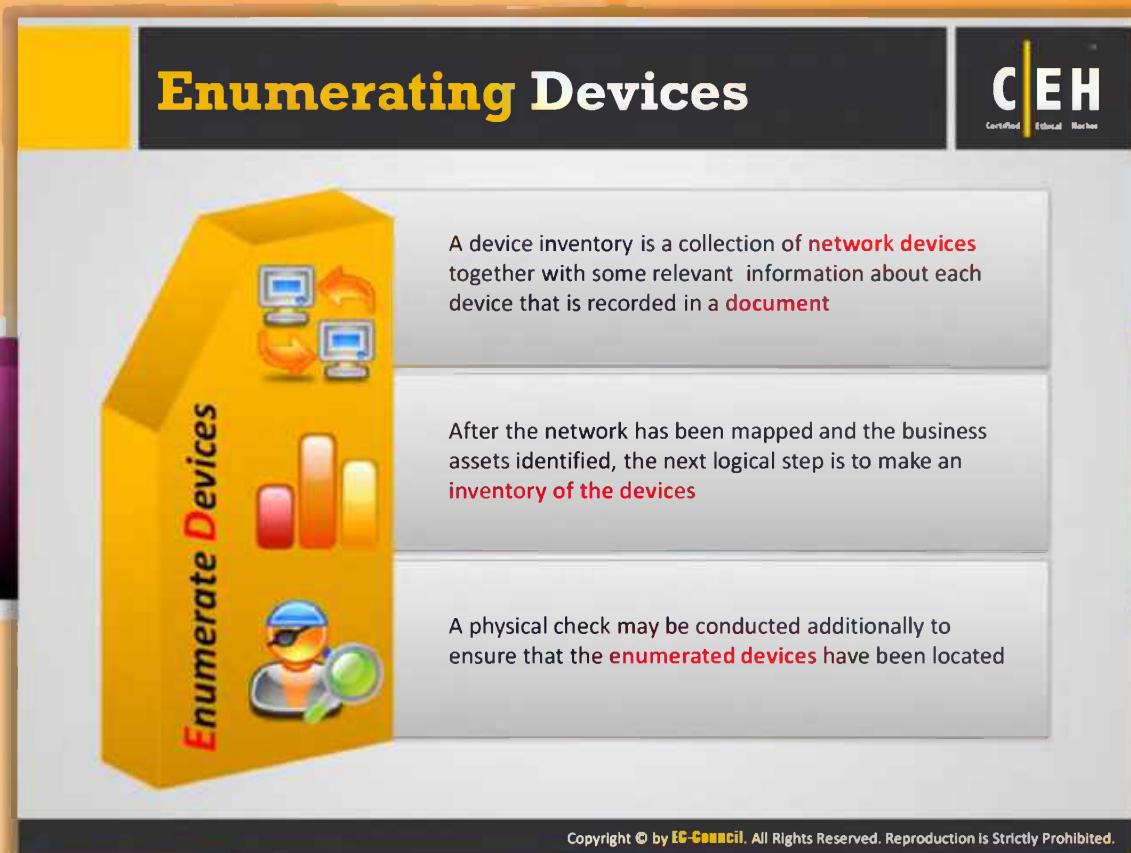
Firewall Testing: The information gained during the **pre-attack phase** using techniques such as firewalking is further exploited here. Attempts are made to evade the IDS and bypass the firewall.

The processes include but are not limited to:

Crafting and sending packets to check firewall rules. For example, sending SYN packets to test stealth detection. This determines the nature of various packet responses through the firewall. A SYN packet can be used to enumerate the target network. Similarly, other port scans with different flags set can be used to attempt enumeration of the network. This also gives an indication of the source port control on the target.

Usually, perimeter testing measures the firewall's ability to handle fragmentation: big packet fragments, overlapping fragments, flood of packets, etc. Testing methods for **perimeter security** include but are not limited to:

- ⌚ Evaluating error reporting and error management with ICMP probes
- ⌚ Checking access control lists with crafted packets
- ⌚ Measuring the threshold for denial-of-service by attempting persistent TCP connections, evaluating transitory TCP connections, and attempting streaming UDP connection
- ⌚ Evaluating protocol-filtering rules by attempting connections using various protocols such as SSH, FTP, and Telnet
- ⌚ Evaluating IDS capability by passing malicious content (such as malformed URLs) and scanning the target for response to abnormal traffic



The slide has a yellow header bar with the title "Enumerating Devices" in white. In the top right corner is the "CEH" logo with the text "Certified Ethical Hacker". The main content area features a large yellow 3D-style box on the left labeled "Enumerate Devices" vertically. Inside the box are three icons: a computer monitor with a circular arrow, a bar chart, and a person wearing a mask with a magnifying glass. The slide contains three text boxes:

- A device inventory is a collection of **network devices** together with some relevant information about each device that is recorded in a **document**.
- After the network has been mapped and the business assets identified, the next logical step is to make an **inventory of the devices**.
- A physical check may be conducted additionally to ensure that the **enumerated devices** have been located.

At the bottom right of the slide is the copyright notice: "Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited."



Enumerating Devices

A device inventory is a collection of network devices, together with some relevant information about each device, which is recorded in a document. After the network has been mapped and the business assets identified, the next **logical** step is to make an inventory of the devices.

During the initial stages of the pen test, the devices may be referred to by their identification on the network such as IP address, MAC address, etc. This can be done by pinging all devices on the network or by using device enumeration tools.

Later, when there is a physical security check, devices may be cross checked regarding their location and identity. This step can help to identify **unauthorized** devices on the network. The other method is to do ping sweeps to detect responses from devices and later correlate the results with the actual inventory.

The likely parameters to be captured in an inventory sheet would be:

- ⌚ Device ID
- ⌚ Description
- ⌚ Hostname
- ⌚ Physical location
- ⌚ IP address
- ⌚ MAC address

⌚ Network accessibility

Activity: Acquiring Target

CEH

- Acquiring a target refers to the set of activities undertaken where the **tester subjects the suspect machine** to more intrusive challenges such as vulnerability scans and security assessment
- Testing methods **for acquiring target** include but are not limited to:

Active probing assaults:
Use results of the network scans to gather further information that can lead to a compromise



Running vulnerability scans:
In this phase vulnerability scans are completed



Trusted systems and trusted process assessment:
Attempting to access the machine's resources using legitimate information obtained through social engineering or other means



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Activity: Acquiring Target

Usually, target acquisition refers to all the activities that are undertaken to unearth as much information as possible about a particular machine or systems so that it can be used later in the actual process of exploitation. Here, acquiring a target is referred to as the set of activities undertaken where the tester subjects the targeted machine to more **intrusive challenges** such as vulnerability scans and security assessment. This is done to obtain more information about the target and can be used in the exploit phase.

Examples of such activities include subjecting the machine to:

- **Active probing assaults:** Use the results of network scans to gather further information that can lead to a compromise.
- **Running vulnerability scans:** Vulnerability scans are completed in this phase.
- **Trusted systems and trusted process assessment:** Attempting to access the machine's resources using legitimate information obtained through social engineering or other means.

Activity: Escalating Privileges

The tester may take advantage of poor security policies and take advantage of email or unsafe web code to gather information that can lead to escalation of privileges

Once the target has been acquired, the tester attempts to **exploit the system** and gain greater access to the protected resources

Use of techniques such as brute force to achieve privileged status. Examples of tools include get admin and password crackers

Use of Trojans and protocol analyzers

Use of information gleaned through techniques such as social engineering to gain unauthorized access to the privileged resources

Activities include (but are not limited to)

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Activity: Escalating Privileges

When an attacker succeeds in gaining unauthorized access into a system or network, the degree of escalation depends on the various authorizations possessed by an attacker. The ultimate aim of an attacker would be to gain the highest possible administration privilege that gives access to the **entire network, sensitive information, online banking** etc.

Once the target has been acquired, the tester attempts to exploit the system and gain greater access to the protected resources

Activities include (but are not limited to):

- ➊ The tester may take advantage of poor security policies and take advantage of email or unsafe web code to gather information that can lead to the escalation of privileges
- ➋ Use of techniques such as brute force to achieve privileged status. Examples of tools include get admin and password crackers
- ➌ Use of Trojans and protocol analyzers
- ➍ Use of information gleaned through techniques such as social engineering to gain **unauthorized access** to the privileged resources



Activity: Execute, Implant, and Retract

In this phase, the tester effectively compromises the acquired system by executing the arbitrary code. The objective here is to explore the extent to which security fails. The tester attempts to execute the **arbitrary code**, hides files in the compromised system, and leaves the system without raising alarms. He or she then attempts to re-enter the system **stealthily**. Activities include:

- Executing exploits to take advantage of the vulnerabilities identified on the target system.
- Exploiting buffer overflows in order to trick the system into running arbitrary code.
- Executing activities that are usually subjected to containment measures such as the use of **Trojans** and **rootkits**.

Activities in the retract phase include manipulation of audit log files to remove traces of the activities:

- Examples include use of tools such as audit poll. The tester may also change settings within the system to remain inconspicuous during a re-entry and change log settings.
- The tester may re-enter the system using the backdoor implanted by the tester.

Post-Attack Phase and Activities

This phase is critical to any penetration test as it is the responsibility of the tester to restore the systems to their **pre-test states**.

Post-attack phase activities include some of the following:

- Removing all **files uploaded** on the system
- Cleaning all **registry entries** and removing vulnerabilities created
- Removing all **tools and exploits** from the tested systems
- Restoring the **network** to the pre-test state by removing shares and connections
- Analyzing all **results** and presenting the same to the organization

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Post-attack Phase and Activities

This phase is critical to any penetration test as it is the responsibility of the tester to restore the systems to a pre-test state. The objective of the test is to show where **security fails**, and unless there is a scaling of the penetration test agreement, whereby the tester is assigned the responsibility to correct the security posture of the systems, this phase must be completed.

Activities in this phase include (but are not restricted to):

- Removing all files uploaded on the system
- Cleaning all registry entries and removing vulnerabilities created
- Reversing all file and setting manipulations done during the test
- Reversing all changes in privileges and user settings
- Removing all tools and exploits from the tested systems
- Restoring the network to the pre-test stage by removing shares and connections
- Mapping of the network state
- Documenting and capturing all logs registered during the test
- Analyzing all results and presenting them to the organization

The penetration tester should document all his or her activities and record all observations and results so that the test can be repeatable and verifiable for the given **security posture** of the organization. For the organization to quantify the security risk in business terms, it is essential that the tester should identify critical systems and critical resources and map the threat to these.

Penetration Testing Deliverable Templates



A pentest report will carry details of the incidents that have occurred during the **testing process** and the range of activities carried out by the testing team

Broad areas covered include objectives, observations, activities undertaken, and incidents reported

The team may also recommend **corrective actions** based on the rules of the engagement

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Penetration Testing Deliverable Templates

A pen test report carries details of the incidents that have occurred during the testing process and the range of activities that the testing team carries out.

It captures the objectives as agreed upon in the rules of engagement and provides a brief description of the observations from the **testing engagement**.

Under the activities carried out will be all the tests, the devices against which the tests were conducted, and the preliminary observations. These are usually **cross-referenced** to the appropriate **test log entry**.

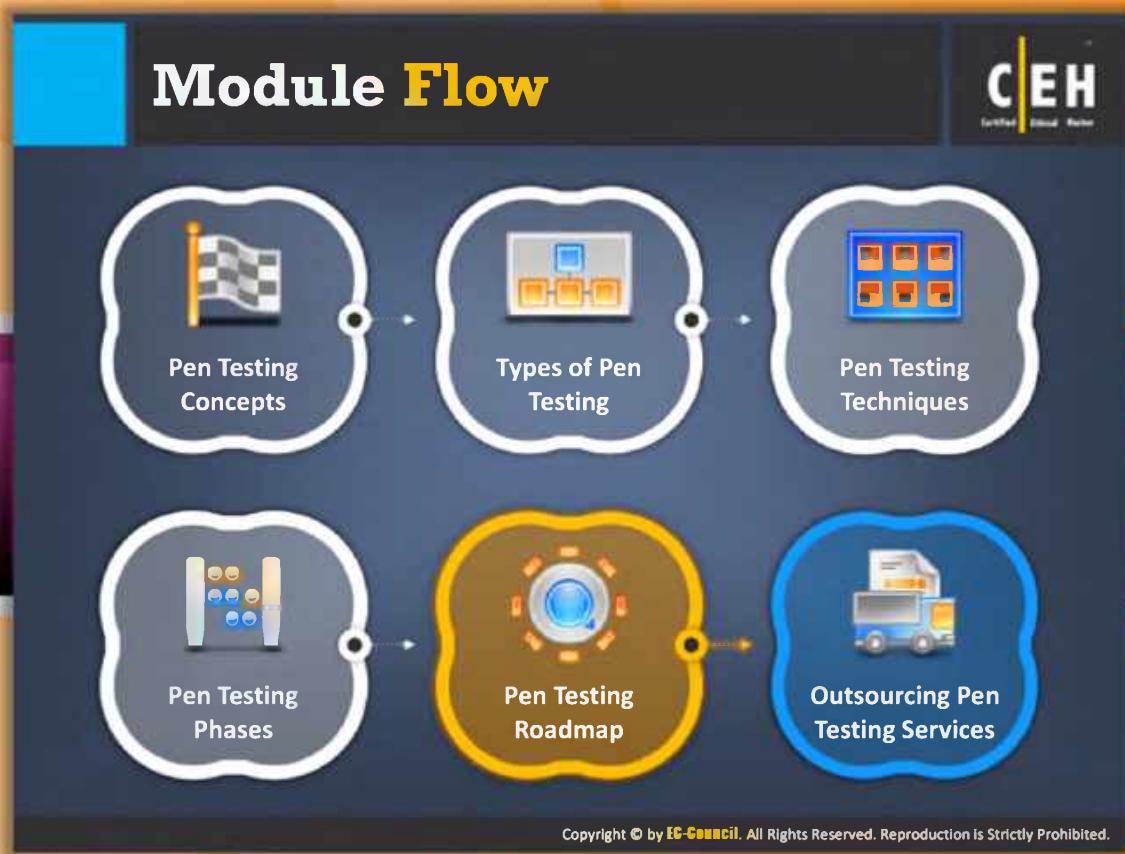
Other information that can be captured under incident description can include:

- ⌚ A detailed description of the incident
- ⌚ The date and time when the incident occurred
- ⌚ Contact information for the person who observed the incident
- ⌚ The stage of testing during which the incident occurred
- ⌚ A description of the steps taken to create the incident. This can be supplemented by screen captures
- ⌚ Observations on whether the incident can be repeated or not

- ④ Details on the tool (if detected), the name and version of the tool, and if relevant, any custom configuration settings

Under risk analysis, the impact of the test is captured from a business perspective. The information included is:

- ④ The initial estimate of the relative severity of the incident to the business
- ④ The initial estimate of the relative likelihood (or frequency) of the incident reoccurring in production
- ④ The initial estimate of the cause of the incident

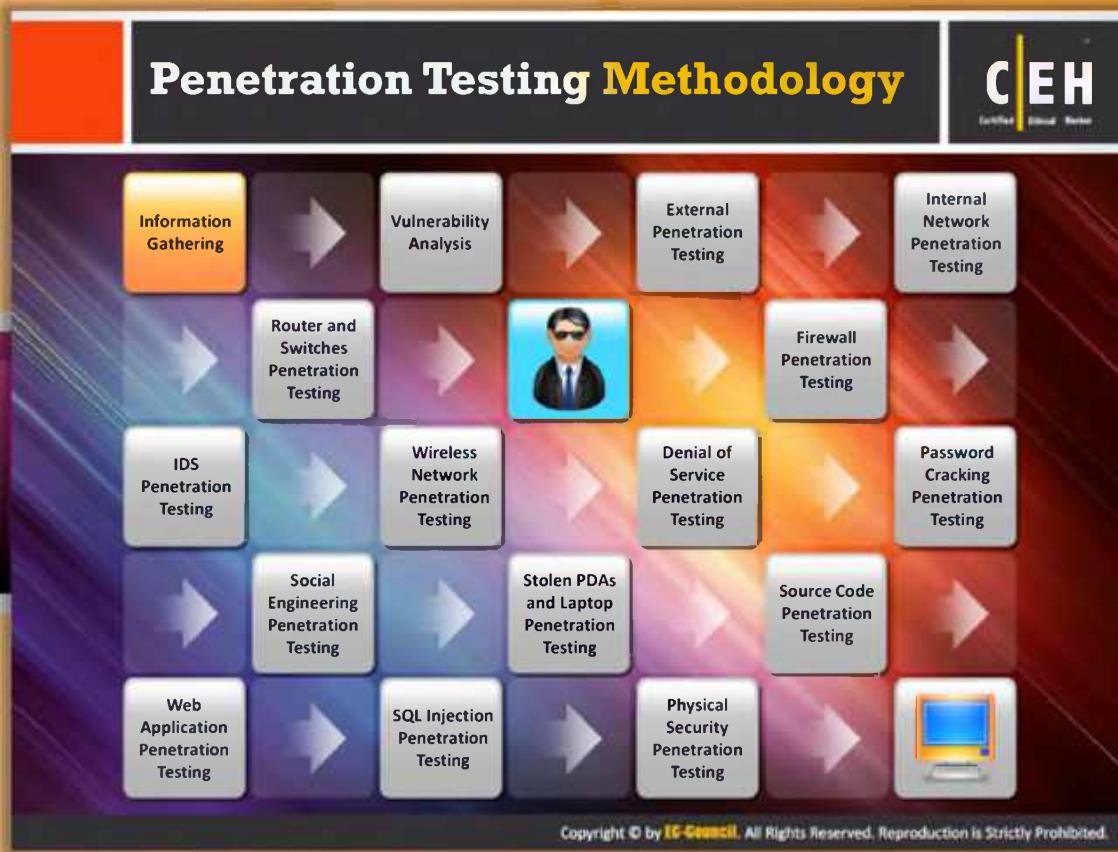


Module Flow

Pen Testing Roadmap

A penetration test is a technique that evaluates or audits the security of a computer system or other facility by launching an attack from a malicious source. It also proves how vulnerable that a computer system would be in the event of the real attack. The rules, practices, methods as well as procedures implemented, followed during the course of any information security audit program are defined by pen testing methodology. This methodology defines you a roadmap with proven practices as well as practical ideas that are to be handled with care for assessing the system security correctly. A detailed explanation about the pen testing roadmap is given in the next slides.

	Pen Testing Concepts		Types of Pen Testing
	Pen Testing Techniques		Pen Testing Phases
	Pen Testing Roadmap		Outsourcing Pen Testing Services



Penetration Testing Methodology

The following are the various phases in the penetration testing methodology:



Information Gathering

Information gathering is one of the major steps of the penetration testing. It is the first phase in the penetration testing process. It is done using various tools, scanners, online sources, sending simple http requests, specially crafted requests, etc.



Vulnerability Analysis

Vulnerability analysis is a method of identifying vulnerabilities on a network. It provides an overview of the flaws that exist in a system or network.



External Penetration Testing

An external penetration test is conducted to know whether the external network is secure or not. In external penetration testing, hacking is done in the same way the **actual attacker does but without causing any harm to the network**. This helps in making the network more secure. Various methods used in external penetration testing are:

- ⌚ Footprinting
- ⌚ Public Information & Information Leakage

- ⌚ DNS Analysis & DNS Brute forcing
- ⌚ Port Scanning
- ⌚ System Fingerprinting
- ⌚ Services Probing
- ⌚ Exploit Research
- ⌚ Manual Vulnerability Testing and Verification of Identified Vulnerabilities
- ⌚ Intrusion Detection/Prevention System Testing
- ⌚ Password Service Strength Testing
- ⌚ Remediation Retest (optional)



Internal Network Penetration Testing

In internal network penetration testing, all the possible **internal network flaws** are identified and simulated as if a real attack has taken place. Various methods used for the internal network penetration testing are:

- ⌚ Internal Network Scanning
- ⌚ Port Scanning
- ⌚ System Fingerprinting
- ⌚ Services Probing
- ⌚ Exploit Research
- ⌚ Manual Vulnerability Testing and Verification
- ⌚ Manual Configuration Weakness Testing and Verification
- ⌚ Limited Application Layer Testing
- ⌚ Firewall and ACL Testing
- ⌚ Administrator Privileges Escalation Testing
- ⌚ Password Strength Testing
- ⌚ Network Equipment Security Controls Testing
- ⌚ Database Security Controls Testing
- ⌚ Internal Network Scan for Known Trojans
- ⌚ Third-Party/Vendor Security Configuration Testing



Router and Switches Penetration Testing

Router switches penetration is carried out to determine:

- ⌚ End to end router security
- ⌚ Bandwidth and speed of the internet connection

- ⌚ Data transfer speed
- ⌚ Router performance
- ⌚ Router Security assessment



Firewall Penetration Testing

Firewall penetration testing is one of the most useful methods in analyzing security effectiveness. Through this method, you can identify how secure your firewall network is against the attacks performed by network intruders.



IDS Penetration Testing

An intrusion detection system (IDS) can be software or hardware. IDS penetration testing helps you to test the strength of the IDS. It can be performed with the help of tools such as IDS informer, an evasion gateway, etc.



Wireless Network Penetration Testing

Wireless networks are more economical than wired networks. Though wireless networks are cheaper, there are various risks associated with them. A wireless network is less protected than a wired one. Therefore, wireless networks must be tested strictly and the respective security enhancements must be applied.



Denial-of-Service Penetration Testing

The main purpose of a denial-of-service (DoS) attack is to slow down the website or even to crash it by sending too many requests, more than a particular server can handle. If the attacker knows the details of the server and its technical specifications, it becomes more vulnerable. Sometimes DoS is done on a **trial and error basis**. So the penetration tester must check how much the website or server can withstand. It is also necessary to provide an alternative way to react to the situation when the limit exceeds.



Password Cracking Penetration Testing

Passwords are used to protect computer resources from unauthorized access. Password cracking penetration testing identifies the vulnerabilities associated with **password management**. This helps in avoiding various kinds of malicious attacks such as brute force attacks, hybrid attacks, and dictionary attacks, etc.



Social Engineering Penetration Testing

Social engineering is a method used by attackers to get crucial information of a company. Attackers especially target individuals within the organization to gather as much information as possible about the company. This is completely documented and then the employees are educated about possible social engineering attacks and cautioned about various threats.

Stolen Laptops, PDAs, and Cell Phones Penetration Testing



The penetration tester should find out the possible **loopholes in physical locality** and identify the various ways that an intruder can enter into the company. Once the important electronic devices that contain sensitive information of the company are stolen, you can extract information from these stolen devices. Therefore, such penetration testing proves very beneficial. Penetration tests are done especially on senior members of the company as their PDAs, laptops and mobile phones often contain sensitive information.

Source Code Penetration Testing



The penetration tester should perform source code analysis by using some source code analysis tools. These tools will help the pen tester to detect the vulnerabilities in the source code.



Application Penetration Testing

Programmers may make some mistakes at the time of software creation. Those mistakes can become potential vulnerabilities. Application penetration testing helps in determining the design error of the software.



SQL Injection Penetration Testing

The penetration tester should perform SQL injection penetration testing on the application in order to find out vulnerabilities in the application. The pen tester should try to simulate different types of SQL injection attacks to find the possible vulnerabilities.



Physical Security Penetration Testing

Here the penetration tester tries to gain physical access to the organizational resources before, during, and after business hours. All the **physical security controls** must be properly tested.



Penetration Testing Methodology



Surveillance Camera Penetration Testing

A surveillance camera can be used to monitor the live target. The surveillance camera can be prone to security flaws due to non-robust design of the **web interface** created for the surveillance camera activities. As a pen tester, you should try to find out vulnerabilities in the web interface of the surveillance camera. You should do the following things to test the surveillance camera:

- ➊ The web interface should be completely debugged
- ➋ Try to look for the injection points from where the motion images are included remotely
- ➌ Validate the image path
- ➍ Create the different motion picture recorder and editor in order to validate motion or picture recorded by the surveillance camera whether they are same or not



Database Penetration Testing

In this process, a penetration tester tries to directly access data contained in the database or indirectly accessing the data through triggers or stored procedures executed by a database engine. This method helps in avoiding **unauthorized access of data**.

VoIP Penetration Testing



In VoIP penetration testing, access to the VOIP network is attempted to record the conversations and even a DoS attack may also be used to find out the company's security policies.



VPN Penetration Testing

Sometimes, employees are allowed to work from home or remotely and in such situations, there are lot of security issues associated with VPN. So the penetration team attempts to gain access to the VPN through a remote endpoint or a **VPN tunnel** and check the vulnerabilities.



Cloud Penetration Testing

Cloud computing systems are widespread today. There are risks associated with cloud computing. The organizations must figure out these risks and apply proper security mechanisms to protect against potential risks. To find out the vulnerabilities in a cloud-based application, conduct a penetration test on the cloud.



Virtual Machine Penetration Testing

An attacker can exploit the virtual machine security flaw by running malicious code on the virtual machine. The pen tester needs to find out the vulnerabilities in the VM by simulating the actions of an attacker before a real attack occurs.



War Dialing

Dial-up modems used by the companies have various vulnerabilities. These allow attackers to hack a system or network easily. Wardialing penetration testing will be useful:

- ④ To identify the vulnerabilities of the modems.
- ④ To know the passwords related vulnerabilities.
- ④ To know whether there is any open access to organizations systems or not.



Virus and Trojan Detection

Viruses and Trojans are the most widespread malicious software today. Once on the system and networks, these are very dangerous. Early detection of viruses and Trojans is very important.



Log Management Penetration Testing

A management log contains a record of all the events that use a data grid network. It contains the complete track of events such as status of node, agent transmission, job request, etc. Therefore, proper log management helps in tracking any malicious activity such as unauthorized access from outside attackers at an early stage.

File Integrity Checking



Checking the integrity of a file is the best way to tell whether it is corrupted or not. It involves checking the following things:

- ☛ File size
- ☛ Version
- ☛ When it was created
- ☛ When it was modified
- ☛ The login name of any user who modifies the file
- ☛ Its attributes (e.g., Read-Only, Hidden, System, etc.)

Mobile Devices Penetration Testing



In mobile penetration testing, the pen tester tries to access and manipulate the data on the particular mobile device simulating all possible attacks such as using social engineering, uploading malicious code, etc. Mobile device penetration pinpoints and addresses gaps in end-user awareness and security exposures in these devices before attackers actually misuse and compromise them.

Telecom and Broadband Penetration Testing



The pen tester tries to determine the vulnerabilities in the broadband connection of the particular corporate network. The pen tester simulates different types of attacks such as unauthorized access, installation of malicious software, DoS attacks on broadband connections to check whether the network withstands these types of attacks.

Email Security Penetration Testing



Email security penetration testing helps to check all the vulnerabilities associated with an email mechanism.

Security Patches Penetration Testing



Unless the system or software is updated with the latest security patches, it is vulnerable to attacks. Poorly designed security patches have more vulnerability so testing them helps in resolving such issues.

Data Leakage Penetration Testing



Penetration testing of data leakage helps in the following ways:

- ☛ Preventing confidential information from going out to the market or to competitors
- ☛ Allows increasing internal compliance level for data protection
- ☛ Improves awareness amongst employees on Safe Practices
- ☛ Will be useful to easily demonstrate compliance to regulations
- ☛ Controls exposure with workflows for mitigation



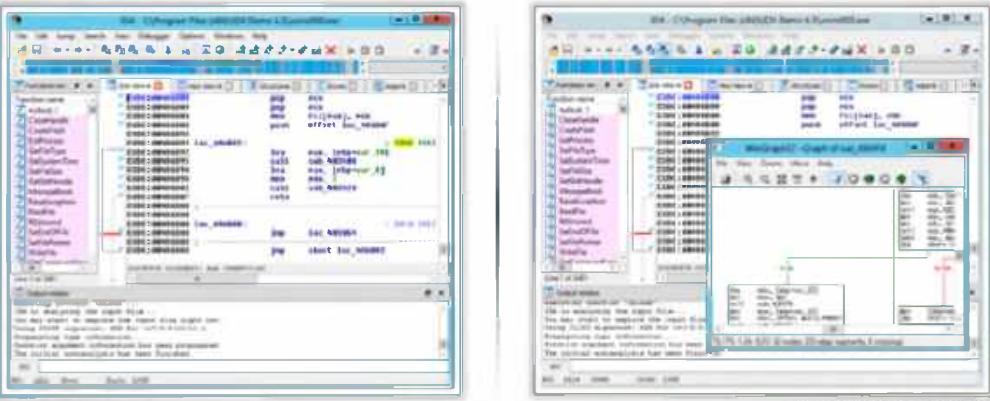
SAP Penetration Testing

Attackers may be able to break into SAP platform and can perform espionage, sabotage, and **fraud attacks on business-critical information**. The SAP penetration testing service simulates the process performed by an attacker. In SAP penetration testing, the pen tester tries to find the vulnerabilities in the SAP platform by conducting different types of attacks, and then checks whether he or she is able to break into the SAP platform.

Application Security Assessment

CEH
Certified Ethical Hacker

- Application security assessment is an **in-depth analysis of applications** to identify and assess **security vulnerabilities** that can expose the organization's sensitive information
- This test checks on application so that a malicious user cannot **access, modify, or destroy data or services** within the system



http://www.hex-rays.com

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Application Security Assessment

Application security assessment is done by a security professional to identify security vulnerabilities and significant issues.

Application security assessment involves:

- ❑ Inspection of application validation and bounds checking for both accidental and mischievous input.
- ❑ Manipulation of client-side code and locally stored information such as session information and configuration files.
- ❑ Examination of application-to-application interaction between system components such as the web service and back-end data sources.
- ❑ Discovery of opportunities that could be utilized by an attacker to escalate their permissions.
- ❑ Examination of event logging functionality.
- ❑ Examination of authentication methods in use for their robustness and resilience to various subversion techniques.

Even in a well-deployed and secured infrastructure, a weak application can expose the organization's crown jewels to unacceptable risk.

Application security assessment is designed to identify and assess threats to the organization through **bespoke** or proprietary applications or systems. This test checks the application so that a malicious user cannot access, modify, or destroy data or services within the system.

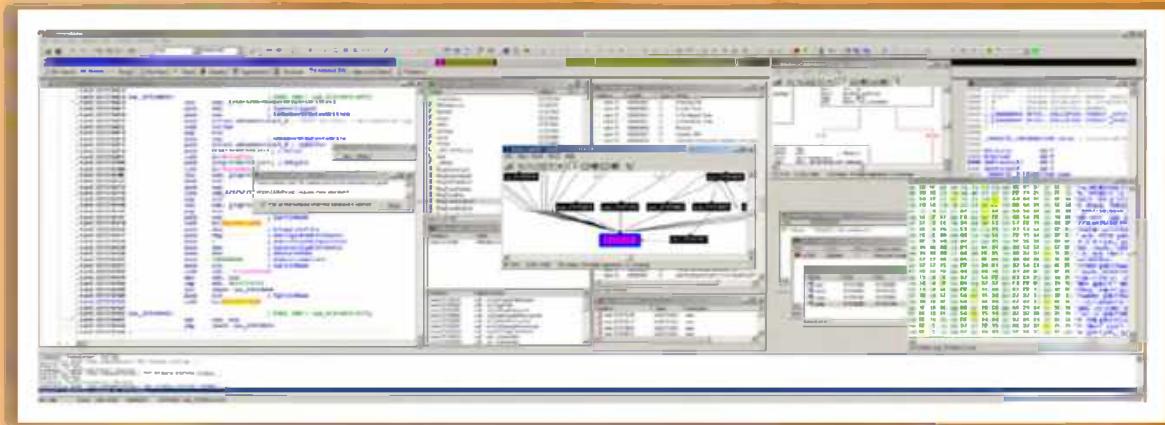
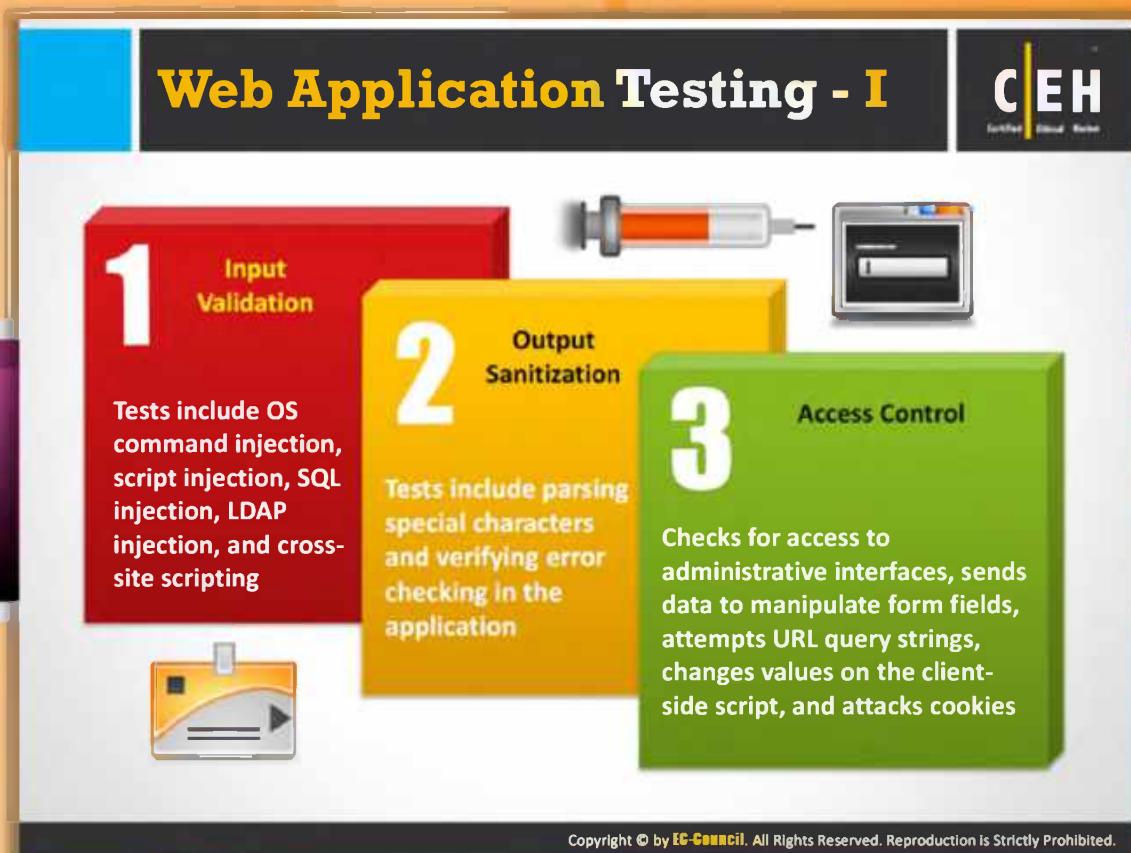


FIGURE 20.2: Application security assessment screenshot



Web Application Testing – I

This test phase can be carried out as the tester proceeds to acquire the target.



Input validation

Tests include OS command injection, script injection, SQL injection, LDAP injection, and cross-site scripting. Other tests include checking for dependency on the external data and the source verification.



Output sanitization

Tests include parsing special characters and verifying error checking in the application.



Access control

The tester checks access to administrative interfaces, transfers data for manipulating form fields, checks URL query strings, changes the values of **client-side script**, and attacks cookies. Other tests include checking for authorization breaches, enumerating assets accessible through the application, lapses in event handling sequences, proxy handling, and compliance with **least privilege access rule**.

Web Application Testing - II



Checking for buffer overflows include attacks against stack overflows, heap overflows, and format string overflows

Component checking checks for security controls on web server/application components that might expose the web application to vulnerabilities

DoS checking tests for DoS induced by malformed user input, user lockout, and application lockout due to traffic overload, transaction requests, or excessive requests on the application

Data and error checking checks for data-related security lapses such as storage of sensitive data in the cache or throughput of sensitive data using HTML

1. Checking for Buffer Overflows
2. Component Checking
3. Denial of Service
4. Data and Error Checking

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Web Application Testing - II



Checking for Buffer Overflows

Tests include attacks against stack overflows, heap overflows, and format string overflows.



Denial-of-service

Test for DoS is induced due to malformed user input, user lockout, and application lockout due to traffic overload, transaction requests, or excessive requests on the application.



Component checking

Check for security controls on web server/application components might expose the web application to vulnerabilities, such as basic authentication.



Data and error checking

Check for data-related security lapses such as storage of sensitive data in the cache or input of sensitive data using HTML. Check for verbose error messages that give away more details of the application than necessary and error type.

SQL injection techniques

SQL injection may be attempted against web applications to gain access to the **target system**.

Web Application Testing - III **CEH**
Certified Ethical Hacker

Confidentiality Check	Session Management	Configuration Verification
For applications using secure protocols and encryption , check for lapses in key exchange mechanism, adequate key length, and weak algorithms	It checks time validity of session tokens, length of tokens, expiration of session tokens while transiting from SSL to non-SSL resources , presence of any session tokens in the browser history or cache, and randomness of session ID (check for use of user data in generating ID)	It attempts to manipulate resources using HTTP methods such as DELETE and PUT , check for version content availability and any visible restricted source code in public domains, attempt directory and file listing, and test for known vulnerabilities and accessibility of administrative interfaces in servers and server components

Copyright © by EC-Council All Rights Reserved. Reproduction is Strictly Prohibited.

Web Application Testing - III



Confidentiality check

For applications using secure protocols and encryption, check for lapses in key exchange mechanism, inadequate key length, and weak algorithms. Validate authentication schemes by attempting user enumeration through login or a recovery process. Check digital certificates and use a signature verification process.



Session management

Check time validity of session tokens, length of tokens, and expiration of session tokens while transiting from **SSL to non-SSL** resources, presence of any session tokens in the **browser** history or **cache**, and randomness of session ID (check for use of user data in generating an ID).



Configuration verification

Attempt manipulation of resources using HTTP methods such as **DELETE** and **PUT**, check for version content availability, and any visible restricted source code in public domains, attempt directory, and file listing, test for known vulnerabilities, and accessibility of administrative interfaces in the server and server components.

Network Security Assessment

C|EH
Certified Ethical Hacker

- 1 It scans the network environment for identifying vulnerabilities and helps to improve an enterprise's security policy
- 2 It uncovers network security faults that can lead to data or equipment being exploited or destroyed by Trojans, denial of service attacks, and other intrusions
- 3 It ensures that the security implementation actually provides the protection that the enterprise requires when any attack takes place on a network, generally by "exploiting" a vulnerability of the system
- 4 It is performed by a team attempting to break into the network or servers

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Network Security Assessment

Network security assessment is an effective method to protect the systems from external attacks. **Vulnerabilities present in routers, firewalls, DNS, web and database servers**, and other systems become a doorway to attackers to perform attacks. Network assessment helps in reducing the risks related to networks. It gives a more clear idea about the risks posed by external and internal attackers.

- It scans the network environment for identifying vulnerabilities and helps to improve an enterprise's security policy
- It uncovers network security faults that can lead to data or equipment being exploited or destroyed by Trojans, denial-of-service attacks, and other intrusions
- It ensures that the security implementation actually provides the protection that the enterprise requires when any attack takes place on a network, generally by "**exploiting**" a vulnerability of the system
- It is performed by a team attempting to break into the network or servers

Wireless/Remote Access Assessment

Wireless/Remote Access assessment involves assessing risks associated with wireless/cellular networks, VPN systems, and mobile devices

The diagram features a central figure of a person wearing a mask and sunglasses, surrounded by six colored circles representing different wireless technologies: Bluetooth (orange), 802.11a,b and g (blue), Wireless networks (green), Radio communication channels (yellow), Wireless radio transmissions (teal), and GHz signals (light green). The background is a gradient from red to blue, with the text "Wireless Testing" curved above the central figure.

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Wireless/Remote Access Assessment

Wireless/remote access assessment addresses the security risks associated with an increasing mobile workforce. Wireless networking has various benefits as well as security risks. Assessment includes testing the following things:

- ⌚ Bluetooth
- ⌚ 802.11a,b and g
- ⌚ Wireless networks
- ⌚ Radio communication channels
- ⌚ Wireless radio transmissions
- ⌚ GHz signals

Wireless Testing

CEH
Certified Ethical Hacker

Methods for wireless testing include but are not limited to:

- Check if the access point's default **Service Set Identifier (SSID)** is easily available. Test for "broadcast SSID" and accessibility to the LAN through this. Tests can include **brute forcing the SSID character string** using tools like Kismet
- Check for **vulnerabilities in accessing the WLAN** through the wireless router, access point, or gateway. This can include verifying if the default Wired Equivalent Privacy (WEP) encryption key can be captured and decrypted
- Audit for broadcast beacon** of any access point and check all protocols available on the access points. Check if **Layer 2 switched networks** are being used instead of hubs for access point connectivity
- Subject authentication to playback of previous authentications in order to check for **privilege escalation and unauthorized access**
- Verify that **access is granted only to client machines** with registered MAC addresses



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Wireless Testing

A wireless network can be attacked in **multiple ways** and conducting a penetration test is difficult process here, compared to a wired network. To launch the attack against wireless networks, attackers use various methods such as:

- Denial-of-service attacks
- Man-in-the-middle attacks
- ARP poisoning attacks

Methods for wireless testing include but are not limited to:

- Check if the access point's default **Service Set Identifier (SSID)** is easily available. Test for "broadcast SSID" and accessibility to the LAN through this. Tests can include brute forcing the SSID character string using tools like Kismet
- Check for vulnerabilities in accessing the WLAN through the wireless router, access point, or gateway. This can include verifying if the default Wired Equivalent Privacy (WEP) encryption key can be captured and decrypted

- ⌚ Audit for a **broadcast beacon** of any access point and check all protocols available on the access points. Check if Layer 2 switched networks are being used instead of hubs for access point connectivity
- ⌚ Subject authentication to playback of previous authentications in order to check for privilege escalation and unauthorized access
- ⌚ Verify that access is granted only to client machines with registered MAC addresses

Telephony Security Assessment

The Certified Ethical Hacker (CEH) logo is in the top right corner.

A telephony security assessment is performed to identify vulnerabilities in **corporate voice technologies** that might result in toll fraud, eavesdropping on calls, unauthorized access to voice mail systems, DoS attack, etc.

Telephone security assessment includes **security assessment** of PBXs, Voice over IP (VoIP) systems, modems, mailboxes, etc.

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Telephony Security Assessment

The main objective of a telephony assessment is to conduct:

- ⌚ Toll fraud
- ⌚ Eavesdropping on telephone calls
- ⌚ Unauthorized access to voicemail system

A telephony security assessment addresses security concerns relating to **corporate voice technologies**. This includes the abuse of PBXs by outsiders to route calls at the **target's expense**, mailbox deployment and security, voice over IP (VoIP) integration, unauthorized modem use, and associated risks. Telephony security assessment consists of:

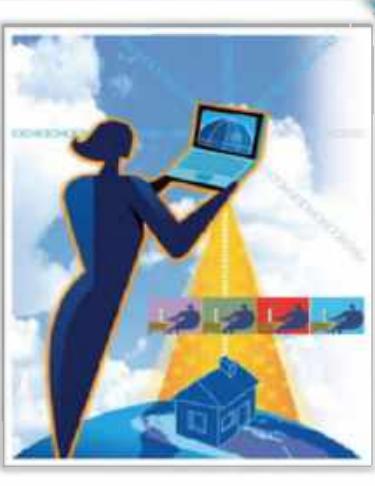
- ⌚ PBX testing
- ⌚ Voicemail testing
- ⌚ FAX review
- ⌚ Modem testing

Social Engineering

CEH
Certified Ethical Hacker

- Social engineering refers to the **non-technical** information system attacks that rely on tricking people to divulge sensitive information
- It exploits **trust, fear, and helping nature** of humans to extract the sensitive data such as security policies, sensitive documents, office network infrastructure, passwords, etc.





Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

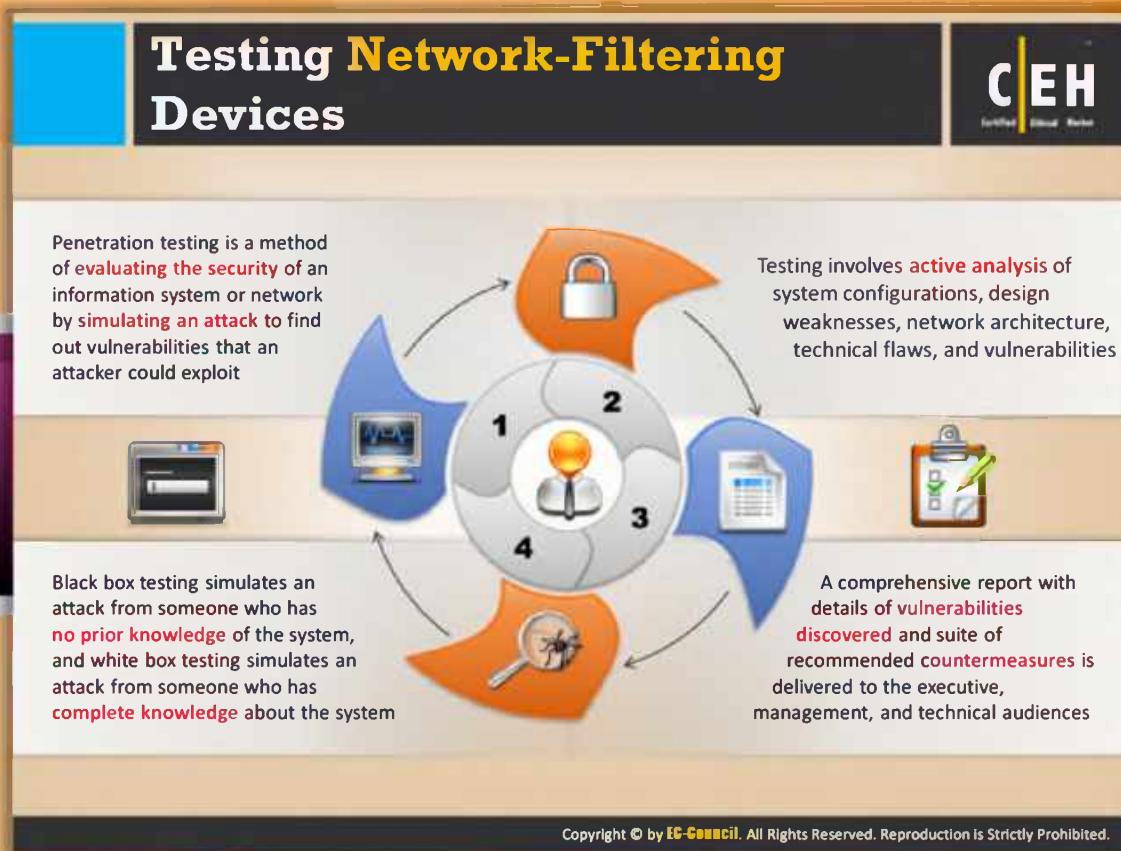


Social Engineering

Social engineering refers to the method of influencing and persuading people to reveal sensitive information in order to perform some malicious action. You can use this to gather confidential information, authorization details, and access details by deceiving and manipulating people.

All security measures adopted by the organization are in vain when employees get “**socially engineered**” by strangers. Some examples of social engineering include unwittingly answering the questions of strangers, replying to spam emails, and bragging in front of **co-workers**.

Most often, people are not even aware of a security lapse on their part. Possibilities are that they divulge information to a potential attacker inadvertently. Attackers take special interest in developing social engineering skills, and are so proficient that their victims don’t even realize that they have been scammed. Despite having security policies in the organization they can be compromised because social engineering attacks target the weakness of people to be helpful for launching their attack. Attackers always look for new ways to gather information; they ensure that they know the people on the perimeter—**security guards, receptionists**, and help desk workers—in order to exploit the human’s oversight. People have been conditioned not to be overly suspicious; they associate certain behavior and appearances with known entities.



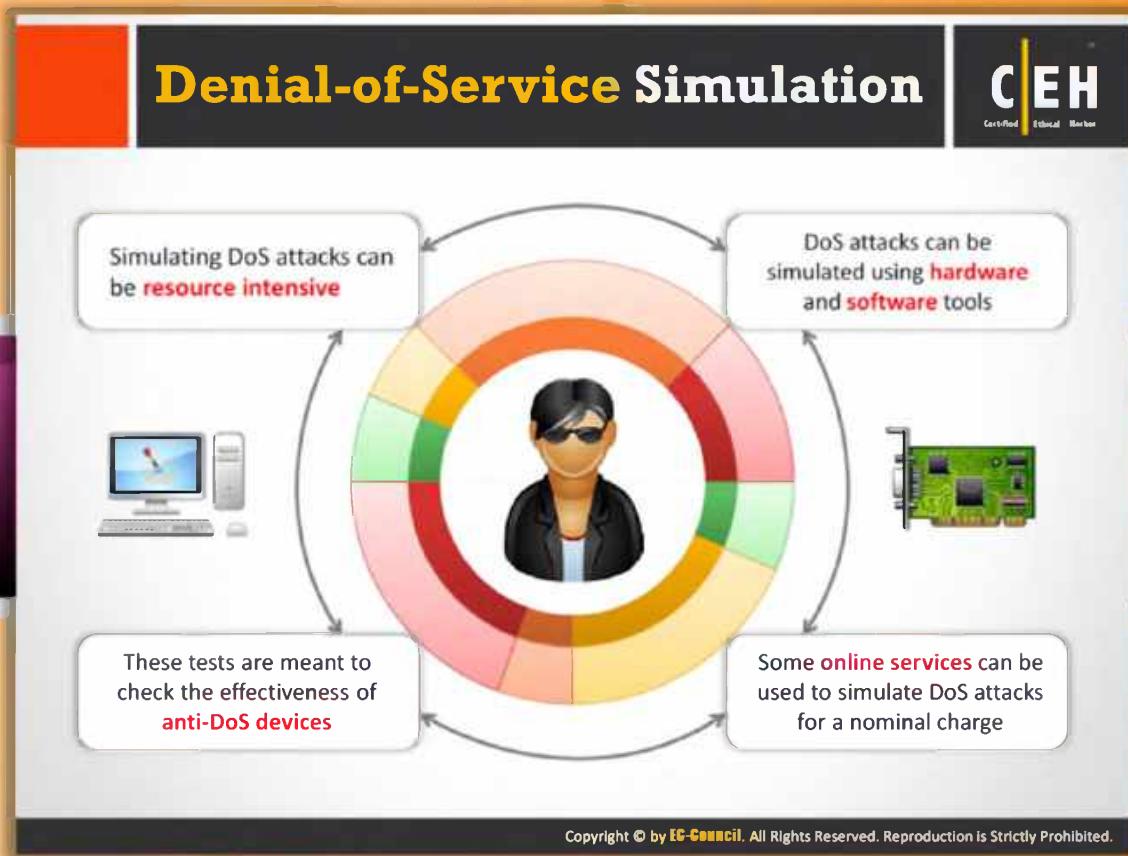
Testing Network-filtering Devices

There are various ways to configure network-filtering devices. In some instances, they may be careless to check malicious traffic, while in others; they may be strict to allow legitimate traffic. The objective of the pen test team would be to ascertain that only **legitimate traffic flows** through the filtering device. However, if multiple filters are used, like a DMZ configuration that uses two firewalls, each filter has to be tested to make sure that it has been configured in the correct way.

It is a fact, however, that even the most preventive firewall cannot restrict network intrusion when the intrusion is initiated within the organization. Most firewalls have the ability to log all activities. But, if the logs are unmonitored over a period of time, they may hinder the functionality of the firewall. Pen testers may test the firewall for endurance by checking the logs and ensuring that the logging activity does not interfere with the firewall's primary activity.

Proxy servers may be subjected to tests to determine their ability to filter out unwanted packets. The pen testers may recommend the use of a load balancer if the traffic load seems to be affecting the filtering capabilities of the devices.

Testing for default installations of the firewall can be done to ensure that default user IDs and passwords have been disabled or changed. Testers can also check for any **remote login capability** that are enabled and allow an intruder to disable the firewall.



Denial-of-Service Emulation

There are two classes of DoS: magic packet attacks and resource-exhaustion attacks.

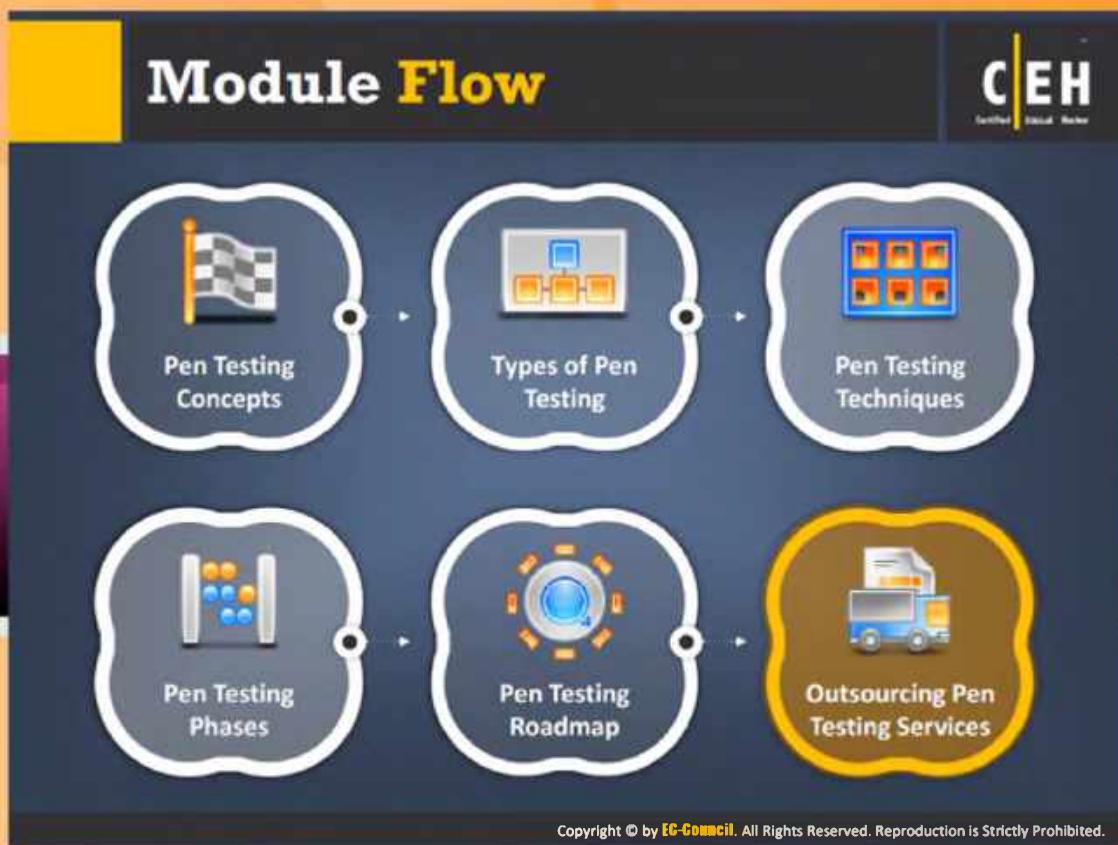
Magic packet attacks usually take advantage of the existing vulnerability in the OS or application for vast abnormal response and excessive CPU utilization or a full system crash by sending one or a few particular packets, for example, **WinNuke** and **Ping of Death**.

Resource-exhaustion attacks do not completely rely on the vulnerabilities; instead they make use of the available computer resources. A resource-exhaustion DoS attack is implemented by intentional utilization of the maximum resources and then stealing them.

While small DoS attacks can be duplicated by running DoS from one machine connected to the target network, large tests that seek to duplicate DoS attacks may need to utilize many machines and large amounts of network bandwidth. These may prove to be time consuming and resource intensive, as well. Instead of **deploying several generic servers**, hardware devices may be used to create large volumes of network traffic. They can also come with attack/testing modules that are designed to emulate the most common DoS attacks.

Simulating hacker attacks can include spoofing the DoS source address to that of a router or device on the network itself so that if the IDS are triggered, the network cuts itself off and the objective is achieved. Another option is to emulate the DoS from an online site over the Internet. Some firms offer this service for a charge and **route traffic** over the Internet to emulate the attack.

There are several tools available to simulate a denial-of-service attack and assess the effectiveness of anti-DoS devices. For example, Web Avalanche can be configured to increase the **connection-per-second** rate and bandwidth usage. This formulates connections which is less latent and usually faster than the average user's HTTP connection. However, this may not essentially affect the capabilities of the devices that are tested to study traffic.



Module Flow

Pen testing results can be effective when the test is performed by a skilled pen tester. Hiring a highly skilled professional on permanent basis may be a huge investment; therefore, most companies prefer outsourcing their pen testing services. Outsourcing the pen testing can increase the frequency, scope, and consistency of its security evaluations.

Pen Testing Concepts	Types of Pen Testing
Pen Testing Techniques	Pen Testing Phases
Pen Testing Roadmap	Outsourcing Pen Testing Services

A detailed explanation about outsourcing penetration testing services is explained on the next slides.

Outsourcing Penetration Testing Services

The diagram illustrates the cycle of outsourcing penetration testing services. It features two main concepts represented by arrows: an orange arrow labeled "Driving outsourcer services" and a blue arrow labeled "Underwriting penetration testing". Both arrows converge on a central circular icon depicting a person wearing a suit and sunglasses, representing a pen tester. The background of the slide is white, and there is a green bar at the top left.

- To get the network audited by an external agency to acquire an **intruder's point** of view
- The organization may require a specific **security assessment** and **suggestive corrective measures**

- Professional liability insurance pays for settlements or judgments for which pen testers become liable as a result of their actions, or failure to perform **professional services**
- It is also known as **E&O insurance** or **professional indemnity insurance**

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Outsourcing Penetration Testing Services

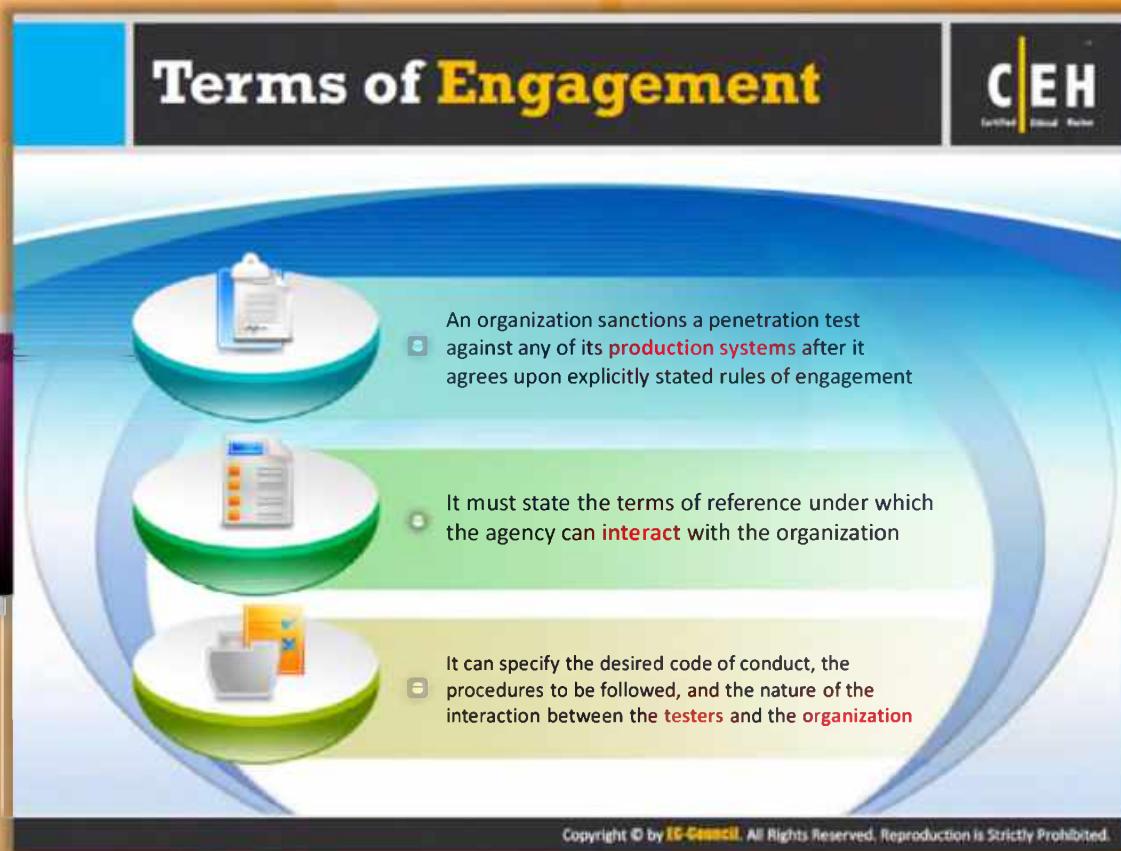
An organization may choose to outsource **penetration-testing** services if there is a lack of specific technical knowledge and expertise within the organization. The organization may require a specific security assessment and suggested corrective measures. Alternatively, the organization may choose to get its network audited by an **external** agency to acquire an intruder's point of view. The need to outsource may also be due to insufficient staff time and resources. The baseline audit may require an ongoing external assessment or the organization may want to build customer and **partner confidence**.

From an organization's perspective, it would be prudent to appoint a cutout. A cutout is a company's in-house monitor over the course of the test. This person will be fully aware of how the test will be conducted, the time frame involved, and the comprehensive nature of the test. The cutout will also be able to intervene during the test to save both pen testers and crucial production systems from **unacceptable damage**.

Underwriting Penetration Testing

- There is an inherent risk involved in undertaking a penetration test. Most organizations would like to know if the penetration testing organization has professional liability insurance. Professional liability insurance pays for settlements or judgments for which pen testers become liable as a result of their actions or failure to perform **professional services**. They take care of the costs involved in defending against the claim, which

includes the attorney's fees, court costs, and other related expenditures involved in investigation, and this also includes the expenditure of the settlement process. From a pen tester's perspective, professional liability insurance is malpractice insurance for professional service providers. It is also known as E&O insurance or professional indemnity insurance.



Terms of Engagement

Source: <http://seclists.org>

Terms of engagement are essential to protect both the **organization's interests** and the pen tester's liabilities. The terms lay down clearly defined guidelines within which the testers can test the systems. They can specify the desired code of conduct, the procedures to be followed, and the nature of interaction between the testers and the organization.

It is prudent for an organization to sanction a penetration test against any of its production systems only after it agrees upon explicitly stated rules of engagement. This contract agreed upon with the pen test agency must state the terms of reference under which the agency can interact with the organization.

For instance, if the pen test agency is undertaking network mapping, the rules of engagement may read as follows: "Pen test agency can obtain much of the required information regarding the site's network profile, such as IP address ranges, telephone number ranges, and other general network topology through public information sources, such as Internet registration services, web pages, and telephone directories. More detailed information about the site's network architecture can be obtained through the use of **domain name server (DNS)** queries, ping sweeps, port scans, and connection route tracing. Informal inquiries, not related to

organization, may also be attempted to gather information from users and administrators that could assist in gaining access to network resources."

Project Scope



Determining the scope of the pentest is essential to decide if the test is a **targeted test** or a comprehensive test

Comprehensive assessments are coordinated efforts by the pentest agency to uncover as much **vulnerability** as possible throughout the organization

A targeted test will seek to **identify vulnerabilities** in specific systems and practices

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Project Scope

Determining the scope of the **pen test** is essential to decide if the test is a targeted test or a comprehensive test. One of the factors that have a significant effect on the effort estimation and cost component of the penetration test is whether or not the pen test agency can undertake a **zero knowledge test** or a partial knowledge test.

Providing even partial knowledge to the pen testers results in time and cost savings. The burden is on the client to make sure that the information provided is complete to the extent intended to be. This is important because if sensitive system data about critical systems is given beforehand, it might defeat the purpose of the penetration test.

If the agency is going to undertake a targeted test, it can seek to identify vulnerabilities in specific systems and practices such as:

- ⊕ Remote access technologies such as **dial-in modems, wireless, and VPN**
- ⊕ Perimeter defenses of Internet-connected systems
- ⊕ Security of web applications and database applications
- ⊕ Vulnerability to denial-of-service attacks

On the other hand, comprehensive assessments are coordinated efforts by the pen test agency to uncover as much vulnerability as possible throughout an organization's IT practices and networked infrastructure.

Pen Test Service Level Agreements

The CEH logo is in the top right corner.

A service level agreement is a contract that details the terms of service that an **outsourcer** will provide

The bottom line is that SLAs define the minimum levels of availability from the testers and determine what actions will be taken in the event of **serious disruption**

SLAs done by experts or professionals can include both **remedies** and **penalties**

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Pen Test Service Level Agreements

The contract agreement that describes the terms of service that an outsourcer provides is known as a Service Level Agreement (SLA). SLAs should match the testing requirements as closely as possible. Proficiently done SLAs can include **remedies** and **penalties** for missing particular service levels.

These penalties encourage the pen test team to achieve the objectives, and make sure that they get back on track quickly. Many organizations also ask for referrals and examples of SLAs they have used with other customers who had similar testing needs. The organization may want to verify the metrics used and the quality of the results achieved to assess the ability of the pen-test team to meet its requirements.

From a pen tester's perspective, it may be difficult to provide examples of real-world SLAs because they are considered confidential business information, similar to other contract terms. The bottom line is that **SLAs** define the minimum levels of availability from the testers and determine what actions can be taken in the event of serious disruption.

Normally, the contract covers those issues as compensation, warranties and remedies, resolution of disputes, and legal compliance. It basically frames the **relationship**, and

determines the **major responsibilities**, both during normal testing and in an emergency situation.

Penetration Testing Consultants

C|EH
Certified Ethical Hacker

Hiring **qualified** penetration tester results in the **quality** of the penetration testing

Main role of penetration testing consultants include **validation of security controls** implemented across an organization's external or internal resources such as firewalls, servers, routers, etc., and develop security policies and procedures

Each area of the network must be examined **in-depth**

A proficient pen tester should possess experience in **different IT fields** such as software development, systems administration, and consultancy

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Penetration Testing Consultants

When companies outsource penetration testing, though it is a bit costly to **hire qualified professionals** who are exclusively trained, it usually yields good results. More qualitative work can be done and desired goals can be achieved.

- ➊ Hiring a qualified penetration tester results in the quality of the penetration testing.
- ➋ A penetration test of a corporate network can examine numerous different hosts (with a number of different operating systems), network architecture, policies, and procedures.
- ➌ Each area of the network must be **examined in-depth**.
- ➍ Penetration testing skills cannot be obtained without years of experience in IT fields, such as development, systems administration, or consultancy.

Module Summary



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

- ❑ A pen test simulates methods that intruders use to gain unauthorized access to an organization's networked systems and then compromise them
- ❑ Penetration testing assesses the security model of the organization as a whole and reveals potential consequences of a real attacker breaking into the network
- ❑ Internal testing involves testing computers and devices within the company
- ❑ Pen testing test components depends on the client's operating environment, threat perception, security and compliance requirement, ROE and budget
- ❑ The penetration testing contract must be drafted by a lawyer and signed by the penetration tester and the company
- ❑ Security assessment categories are security audits, vulnerability assessments, and penetration testing



Module Summary

- ❑ A pen test simulates methods that intruders use to gain unauthorized access to an organization's networked systems and then compromise them.
- ❑ Penetration testing assesses the security model of the organization as a whole and reveals potential consequences of a real attacker breaking into the network.
- ❑ Internal testing will be performed from a number of network access points, representing each logical and physical segment.
- ❑ Pen testing test components depend on the client's operating environment, threat perception, security and compliance requirement, ROE, and budget.
- ❑ The penetration testing contract must be drafted by a lawyer and signed by the penetration tester and the company.
- ❑ Security assessment categories are security audits, vulnerability assessments, and penetration testing.

References

Module 01: Introduction to Ethical Hacking

1. Zero-day attacks are meaner, more rampant than we ever thought, from <http://arstechnica.com/security/2012/10/zero-day-attacks-are-meaner-and-more-plentiful-than-thought/>.
2. SECURITY POLICY: TARGET, CONTENT, & LINKS, from <http://csrc.nist.gov/nissc/1998/proceedings/paperG4.pdf>.
3. Anatomy of the Hack - Hands-on Security, from <http://www.slideshare.net/NewBU/anatomy-of-the-hack-hands-on-security-information-assurance-club>.
4. Hacker methodology, from <http://www.hackersecuritymeasures.com/>.
5. Ethical Hacking, from www.securedeath.com.
6. C. C. Palmer, Ethical hacking from <http://researchweb.watson.ibm.com/journal/sj/403/palmer.html>.
7. An Overview of Computer Security, from www.cc.gatech.edu/classes/AY2005/cs4803cns_fall/security_overview.ppt.
8. Dr. Death, (2006), Ethical Hacking, from <http://www.securedeath.com>.
9. Ethical Hacking, from <http://neworder.box.sk/news/921>.
10. How are Penetrating Testing conducted?, from www.corsaire.com.
11. Ethical Hacking: The Security Justification Redux, from <http://www.sosresearch.org/publications/ISTAS02ethicalhack.PDF>.
12. Ethical Hacking, from www.sosresearch.org/publications.
13. Ethical Hacking, from www.research.ibm.com.
14. Covering Tracks, from <http://rootprompt.org>.
15. Attack, from <http://www.linuxsecurity.com/content/view/17/70/>.
16. Security Issues in Wireless MAGNET at Network Layer, from <http://csce.unl.edu/~jaljaroo/publications/TR02-10-07.pdf>.
17. Glossary of Security and Internet terms, from <http://wssg.berkeley.edu/SecurityInfrastructure/glossary.html>.
18. Glossary of Vulnerability Testing Terminology, from <http://www.ee.oulu.fi/research/ouspg/sage/glossary/>.
19. Information about hackers, from <http://www.antionline.com>.
20. Information about hackers, from http://w2.eff.org/Net_culture/Hackers/.
21. LEX LUTHOR, information about hackers, from <http://bak.spc.org/dms/archive/britphrk.txt>.
22. Information about hackers, from <http://directory.google.com/Top/Computers/Hacking/>.
23. Information about hackers, from <http://directory.google.com/Top/Computers/Security/Hackers/>.
24. Information about hackers, from <http://bak.spc.org/dms/archive/profile.html>.

25. Information about hackers, from
http://dir.yahoo.com/Computers_and_Internet/Security_and_Encryption/Hacking/.

Module 02: Footprinting and Reconnaissance

26. Search Operators, from http://www.googleguide.com/advanced_operators.html.
27. The Complete Windows Trojans Paper, from
http://www.windowsecurity.com/whitepapers/trojans/The_Complete_Windows_Trojans_Paper.html.
28. Naples, (2008), Information Gathering Tools, Available from
http://it.toolbox.com/wiki/index.php/Information_Gathering_Tools.
29. Extract Website Information from archive.org, Available from www.archive.org.
30. Footprinting, from
http://www.ethicalhacker.net/component/option,com_smf/Itemid,49/topic,228.msg672.
31. Simson Garfinkel and David Cox, (2009), Finding and Archiving the Internet Footprint,
<http://simson.net/clips/academic/2009.BL.InternetFootprint.pdf>.
32. CHAPTER 2 [FOOTPRINTING], from <http://www.ecqurity.com/wp/footprinting-encored.pdf>.
33. Donna F. Cavallini and Sabrina I. PACIFICI, Got COMPETITIVE INTELLIGENCE,
<http://www.llrx.com/features/gotci.ppt>.
34. Spammers & hackers: using the APNIC Whois Database to find in their network, from
http://www.apnic.net/info/faq/abuse/using_whois.html.
35. P. Mockapetris, (1987), DOMAIN NAMES - CONCEPTS AND FACILITIES, from
<http://www.ietf.org/rfc/rfc1034.txt>.
36. Manic Velocity, Footprinting And The Basics Of Hacking, from
<http://web.textfiles.com/hacking/footprinting.txt>.
37. Dean, (2001), Windows 2000 Command Prompt Troubleshooting Tools, from
<http://www.pcmech.com/show/troubleshoot/192/>.
38. nslookup Command, from
<http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.aix.doc/cmds/aixcmd/s4/nslookup.htm>.
39. The nslookup Manual Page, from <http://www.stopspam.org/usenet/mmf/man/nslookup.html>.
40. Bob Hillery, (2001), Neohapsis Archives - Incidents list - Re: Finding out who owns ..., from
<http://archives.neohapsis.com/archives/incidents/2001-01/0032.html>.
41. Ryan Spangler, (2003), Analysis of Remote Active Operating System Fingerprinting Tools, from
<http://www.packetwatch.net/documents/papers/osdetection.pdf>.
42. Ryan Spangler, (2003), Analysis of Remote Active Operating System Fingerprinting Tools, from
<http://www.securiteam.com/securityreviews/5ZP010UAAI.html>.
43. Fingerprint methodology: IPID sampling, from <http://www.insecure.org/nmap/nmap-fingerprinting-old.html>.
44. Fyodor, (1998), Remote OS detection via TCP/IP Stack FingerPrinting, from
<http://www.donkboy.com/html/fingerprt.htm>.
45. Remote OS Detection, from <http://nmap.org/book/osdetect.html>.
46. Regional Internet Registry, from http://en.wikipedia.org/wiki/Regional_Internet_Registry.
47. Boy Scouts, Fingerprinting from <http://onin.com/fp/fpmeritbdg.html#top>.
48. The Hacker's Choice, from <http://freeworld.thc.org/welcome/>.
49. THC Fuzzy Fingerprint, from <http://freeworld.thc.org/thc-ffp/>.

50. Katherine Knickerbocker, CJ625 Student Paper, from <http://all.net/CID/Attack/papers/Spoofing.html>.
51. Arik R. Johnson, What is competitive intelligence? How does competitive ..., from <http://aurorawdc.com/whatsci.htm>.
52. Guangliang (Benny), (2006), Spamming and hacking, from http://www.apnic.net/info/faq/abuse/using_whois.html.
53. Dhillon, (2006), Footprinting: The Basics of Hacking: Hack In The Box, from <http://www.hackinthebox.org/modules.php?op=modload&name=News&file=article&sid=5359&mode=tread&order=0&thold=0>.
54. Roshen, (2006), Paladin - Customers - Success Stories - Penetration Testing, from http://paladion.net/penetration_testing.php.
55. Paul Thompson, (2006), Cognitive Hacking and Digital Government: Digital Identity, from <http://www.ists.dartmouth.edu/library/chd0803.pdf>.
56. Greg Sandoval, (2006), MPAA accused of hiring a hacker, from http://news.com.com/MPAA+accused+of+hiring+a+hacker/2100-1030_3-6076665.html.
57. Kurt Seifried, (2005), Closet20001213 Backdoors, Back Channels and HTTP(S), from [http://www.seifried.org/security/index.php/Closet20001213_Backdoors,_Back_Channels_and_HTTP\(S\)](http://www.seifried.org/security/index.php/Closet20001213_Backdoors,_Back_Channels_and_HTTP(S).).
58. Happy Browser, (2005), from <http://www.hotscripts.com/Detailed/39030.html>.
59. Client-server architecture, from http://www.networkinvasion.co.uk/N_scan.htm.
60. Elegant architecture: NASI, from <http://www.nessus.org/features/>.
61. The Signatures, from <http://www.honeynet.org/papers/finger/>.
62. Ryan Spangler, (2003), Analysis of Remote Active Operating System Fingerprinting Tools, Nmap tool : technique, from <http://www.securiteam.com/securityreviews/5ZP010UAAI.html>.
63. Beware!: War dialing, Sandstorm Sandtrap 1.5 Wardialer Detector Plus 16 and Basic Detectors, from http://www.data-connect.com/Santstorm_PhoneSweep.htm.
64. Appendix A - Glossary of Terms: IPSEC, from http://www.imsglobal.org/gws/gwsv1p0/imsgws_securityProfv1p0.html.
65. Def. and info. Vulnerability scanning, from http://www.webencanto.com/computer_glossary/Communications/Ethics/vulnerability_scanning.html.
66. Footprinting, from http://books.mcgraw-hill.com/downloads/products//0072193816/0072193816_ch01.pdf.
67. P. Mockapetris, Zvon - RFC 1034 [DOMAIN NAMES - CONCEPTS AND FACILITIES] - DOMAIN..., from <http://www.zvon.org/tmRFC/RFC1034/Output/chapter3.html>.
68. Gaurav, (2006), The Domain Name System (DNS), from <http://people.csa.iisc.ernet.in/gaurav/np/rfcs/dns.html>.
69. Using the Internet for Competitive Intelligence, from http://www.cio.com/CIO/arch_0695_cicolumn.html.
70. Reporting network abuse: Spamming and hacking, from http://www.apnic.net/info/faq/abuse/using_whois.html.
71. Bastian Ballmann, (2011), Information gathering tools, from [http://www2.packetstormsecurity.org/cgi-bin/search/search.cgi?searchvalue=information+gathering&type=archives&\[search\].x=0&\[search\].y=0](http://www2.packetstormsecurity.org/cgi-bin/search/search.cgi?searchvalue=information+gathering&type=archives&[search].x=0&[search].y=0).
72. Google Earth, from <http://www.google.com/earth/index.html>.
73. pipl, from <https://pipl.com/>.
74. spokeo, from <http://www.spokeo.com>.
75. Zaba Search, from <http://www.zabasearch.com>.

76. 123 People Search, from <http://www.123people.com>.
77. ZoomInfo, from <http://www.zoominfo.com>.
78. PeekYou, from <http://www.peekyou.com>.
79. Wink People Search, from <http://wink.com>.
80. Intelius, from <http://www.intelius.com>.
81. AnyWho, from <http://www.anywho.com>.
82. PeopleSmart, from <http://www.peoplesmart.com>.
83. People Lookup, from <https://www.peoplelookup.com>.
84. WhitePages, from <http://www.whitepages.com>.
85. Facebook, from <https://www.facebook.com/>.
86. LinkedIn, from <http://www.linkedin.com>.
87. Google+, from <https://plus.google.com>.
88. Twitter, from <http://twitter.com>.
89. Google Finance, from <http://finance.google.com/finance>.
90. Yahoo Finance, from <http://finance.yahoo.com>.
91. Zaproxy, from <https://code.google.com/p/zaproxy/downloads/list>.
92. Burp Suite, from <http://portswigger.net/burp/download.html>.
93. Firebug, from <https://getfirebug.com/downloads/>.
94. HTTrack Website Copier, from <http://www.httrack.com/page/2/>.
95. BlackWidow, from <http://softbytelabs.com/us/downloads.html>.
96. Webripper, from <http://www.calluna-software.com/Webripper>.
97. SurfOffline, from <http://www.surfoffline.com/>.
98. Website Ripper Copier, from <http://www.tensons.com/products/websiterippercopier/>.
99. PageNest, from <http://www.pagenest.com>.
100. Teleport Pro, from <http://www.tenmax.com/teleport/pro/download.htm>.
101. Backstreet Browser, from <http://www.spadixbd.com/backstreet/>.
102. Portable Offline Browser, from http://www.metaproducts.com/Portable_Offline_Browser.htm.
103. Offline Explorer Enterprise, from http://www.metaproducts.com/offline_explorer_enterprise.htm.
104. Proxy Offline Browser, from <http://www.proxy-offline-browser.com/>.
105. GNU Wget, from <ftp://ftp.gnu.org/gnu/wget/>.
106. iMiser, from <http://internetresearchtool.com>.
107. Hooey Webprint, from <http://www.hooeywebprint.com.s3-website-us-east-1.amazonaws.com/download.html>.
108. Wayback Machine, from <http://archive.org/web/web.php>.
109. WebSite-Watcher, from <http://aignes.com/download.htm>.
110. eMailTrackerPro, from <http://www.emailtrackerpro.com>.
111. PoliteMail, from <http://www.politemail.com>.
112. Email Lookup – Free Email Tracker, from <http://www.ipaddresslocation.org>.
113. Read Notify, from <http://www.readnotify.com>.
114. Pointofmail, from <http://www.pointofmail.com>.

115. DidTheyReadIt, from <http://www.didtheyreadit.com>.
116. Super Email Marketing Software, from <http://www.bulk-email-marketing-software.net>.
117. Trace Email, from <http://whatismyipaddress.com/trace-email>.
118. WhoReadMe, from <http://whoreadme.com>.
119. MSGTAG, from <http://www.msgtag.com/download/free/>.
120. GetNotify, from <http://www.getnotify.com>.
121. Zendio, from <http://www.zendio.com/download>.
122. G-Lock Analytics, from <http://glockanalytics.com>.
123. EDGAR Database, from <http://www.sec.gov/edgar.shtml>.
124. Hoovers, from <http://www.hoovers.com>.
125. LexisNexis, from <http://www.lexisnexis.com>.
126. Business Wire, from <http://www.businesswire.com>.
127. Market Watch, from <http://www.marketwatch.com>.
128. The Wall Street Transcript , from <http://www.twst.com>.
129. Lipper Marketplace, from <http://www.lippermarketplace.com>.
130. Euromonitor, from <http://www.euromonitor.com>.
131. Fagan Finder, from <http://www.faganfinder.com>.
132. SEC Info, from <http://www.secinfo.com>.
133. The Search Monitor, from <http://www.thesearchmonitor.com>.
134. Compete PRO™, from <http://www.compete.com>.
135. Copernic Tracker, from <http://www.copernic.com>.
136. ABI/INFORM Global, from <http://www.proquest.com>.
137. SEMRush, from <http://www.semrush.com>.
138. AttentionMeter, from <http://www.attentionmeter.com>.
139. Jobitorial, from <http://www.jobitorial.com>.
140. Google Hacking Database, from <http://www.hackersforcharity.org>.
141. MetaGoofil, from <http://www.edge-security.com>.
142. Google Hack Honeypot, from <http://ghh.sourceforge.net>.
143. Goolink Scanner, from <http://www.ghacks.net>.
144. GMapCatcher, from <http://code.google.com>.
145. SiteDigger, from <http://www.mcafee.com>.
146. SearchDiggity, from <http://www.stachliu.com>.
147. Google Hacks, from <http://code.google.com>.
148. Google HACK DB, from <http://www.secpoint.com>.
149. BiLE Suite, from <http://www.sensepost.com>.
150. Gooscan, from <http://www.darknet.org.uk>.
151. WHOIS Lookup at DomainTools.com, from <http://whois.domaintools.com/>.
152. Domain Dossier, from <http://centralops.net/co>.
153. SmartWhois, from <http://www.tamos.com/download/main/index.php>.
154. CountryWhois, from <http://www.tamos.com/products/countrywhois/>.

155. Whois Analyzer Pro, from <http://www.whoisanalyzer.com/download.opp>.
156. LanWhois, from <http://lantricks.com/download/>.
157. HotWhois, from <http://www.tialsoft.com/download/?url=http://www.tialsoft.com/hwhois.exe>.
158. Batch IP Converter, from <http://www.networkmost.com/download.htm>.
159. Whois 2010 Pro, from <http://lapshins.com/>.
160. CallerIP, from <http://www.callerippro.com/download.html>.
161. ActiveWhois, from <http://www.johnru.com/>.
162. Whois Lookup Multiple Addresses, from <http://www.sobelsoft.com/>.
163. WhoisThisDomain, from http://www.nirsoft.net/utils/whois_this_domain.html.
164. SmartWhois, from <http://smartwhois.com>.
165. Whois, from <http://tools.whois.net>.
166. Better Whois, from <http://www.betterwhois.com>.
167. DNSstuff, from <http://www.dnsstuff.com>.
168. Whois Source, from <http://www.whois.sc>.
169. Network Solutions Whois, from <http://www.networksolutions.com>.
170. Web Wiz, from <http://www.webwiz.co.uk/domain-tools/whois-lookup.htm>.
171. WebToolHub, from <http://www.webtoolhub.com/tn561381-whois-lookup.aspx>.
172. Network-Tools.com, from <http://network-tools.com>.
173. Ultra Tools, from <https://www.ultratools.com/whois/home>.
174. dnsstuff, from <http://www.dnsstuff.com/>.
175. network-tools, from <http://network-tools.com/>.
176. DNS Queries, from <http://www.dnsqueries.com/en/>.
177. DIG, from <http://www.kloth.net/services/dig.php>.
178. myDNSTools, from <http://www.mydnstools.info/nslookup>.
179. DNSWatch, from <http://www.dnswatch.info>.
180. DomainTools, from <http://www.domaintools.com>.
181. Professional Toolset, from <http://www.dnsstuff.com/tools>.
182. DNS, from <http://e-dns.org>.
183. DNS Records, from <http://network-tools.com>.
184. DNS Lookup Tool, from <http://www.webwiz.co.uk/domain-tools/dns-records.htm>.
185. DNSData View, from <http://www.nirsoft.net>.
186. DNS Query Utility, from <http://www.webmaster-toolkit.com>.
187. WHOIS-RWS, from <http://whois.arin.net/ui>.
188. Netcraft, from <http://searchdns.netcraft.com/?host>.
189. Shodan, from <http://www.shodanhq.com/>.
190. Path Analyzer Pro, from <http://www.pathanalyzer.com/download.opp>.
191. VisualRoute 2010, from <http://www.visualroute.com/download.html>.
192. Network Pinger, from <http://www.networkpinger.com/en/downloads/#download>.
193. Magic NetTrace, from <http://www.tialsoft.com/download/?url=http://www.tialsoft.com/mNTr.exe>.
194. GEO Spider, from <http://oreware.com/viewprogram.php?prog=22>.

195. 3D Traceroute, from <http://www.d3tr.de/download.html>.
196. vTrace, from <http://vtrace.pl/download.html>.
197. AnalogX HyperTrace, from <http://www.analogx.com/contents/download/Network/htrace/Freeware.htm>.
198. Trout, from <http://www.mcafee.com/apps/free-tools/termsofuse.aspx?url=/us/downloads/free-tools/trout.aspx>.
199. Network Systems Traceroute, from <http://www.net.princeton.edu/traceroute.html>.
200. Roadkil's Trace Route, from <http://www.roadkil.net/program.php/P27/Trace%20Route>
201. Ping Plotter, from <http://www.pingplotter.com>.
202. myiptest, from <http://www.myiptest.com/staticpages/index.php/how-about-you>.
203. Maltego, from <http://www.paterva.com/web6/products/download4.php>.
204. Domain Name Analyzer Pro, from <http://www.domainpunch.com/domain-name-analyzer-pro/download.php>.
205. Web Data Extractor, from <http://www.webextractor.com>.
206. Prefix Whois, from <http://pwhois.org>.
207. Netmask (IRPAS), from <http://www.phenoelit.org/irpas/download.html>.
208. Binging, from <http://www.blueinfy.com/tools.html>.
209. Tctrace (IRPAS), from <http://www.phenoelit.org/irpas/download.html>.
210. Spiderzilla, from <http://spiderzilla.mozdev.org/installation.html>.
211. Autonomous System Scanner (ASS) (IRPAS), from <http://www.phenoelit.org/irpas/download.html>.
212. Sam Spade, from http://www.majorgeeks.com/Sam_Spade_d594.html.
213. DNS DIGGER, from <http://www.dnsdigger.com>.
214. Robtex, from <http://www.robtex.com>.
215. Dig Web Interface, from <http://www.digwebinterface.com>
216. SpiderFoot, from <http://sourceforge.net/projects/spiderfoot/?so.urce=dlp>.
217. Domain Research Tool, from <http://www.domainresearchtool.com>.
218. CallerIP, from <http://www.calleripro.com/download.html>.
219. ActiveWhois, from <http://www.johnru.com>.
220. Zaba Search, from <http://www.zabasearch.com/>.
221. yoName, from <http://yoname.com>.
222. GeoTrace, from <http://www.nabber.org/projects/geotrace/>.
223. Ping-Probe, from <http://www.ping-probe.com/Ping-Probe/index.html>.
224. DomainHostingView, from <http://www.nirsoft.net>.

Module 03: Scanning Networks

225. Explanation of the Three-Way Handshake via TCP/IP, from <http://support.microsoft.com/kb/172983>.
226. Appendix G. Lists of reserved ports, ICMP types and codes, and Internet protocols, from <http://www.ingate.com/files/422/fwmanual-en/xa10285.html>.
227. The Art of Port Scanning - by Fyodor, from http://nmap.org/nmap_doc.html.
228. Methods of IP Network Scanning - Stealth TCP Scanning Methods, from <http://www.codewalkers.com/c/a/Server-Administration/Methods-of-IP-Network-Scanning/3/>.

229. What is Port Scanning and Types of Port Scanning, from <http://www.hackillusion.com/what-is-port-scanning-and-types-of-port-scanning/>.
230. UDP Scan, from <http://www.networkuptime.com/nmap/page3-10.shtml>.
231. Hacking Exposed, from <http://www.scribd.com/doc/62708034/Hacking-Exposed-Book>.
232. Network Security Assessment, from https://www.trustmatta.com/downloads/pdf/Matta_IP_Network_Scanning.pdf.
233. Quick-Tip: SSH Tunneling Made Easy, from <http://www.revsys.com/writings/quicktips/ssh-tunnel.html>.
234. Detecting Spoofed Packets, from <http://seclab.cs.ucdavis.edu/papers/DetectingSpoofed-DISCEX.pdf>.
235. Scanning modes: FIN, Xmas, Null, from <http://www.openxtra.co.uk/support/howto/nmap-scan-modes.php>.
236. Port scanning techniques (Window scan), from <http://www.paulisageek.com/nmap/index.html>.
237. Prabhaker Mateti, UDP Scanning, from <http://www.cs.wright.edu/~pmateti/Courses/499/Probing/>.
238. FTP server bounce attack, TCP Fragmenting, Intrusion detection systems use signature-based mechanisms, from <http://www.in-f-or.it/informatica/docs/portscan.pdf>.
239. Laura Chappell, (2003), OS Fingerprinting With ICMP: ICMP echo, from <http://www.securitypronews.com/it/security/spn-23-20030929OSFingerprintingwithICMP.html>.
240. Scan Type-sF -sX -sN, from <http://content.ix2.net/arc/t-4370.html>.
241. Unixo3/introduction to Nmap, from http://www.samhart.com/cgi-bin/classnotes/wiki.pl?UNIX03/Introduction_To_Nmap.
242. Fyodor, (2006), Art of port scanning: Features, Ideal scanning and related IPID games, Nmap: description, Fingerprint methodology: IPID samplingBounce attacks worked, Technique: TCP reverse ident scanning, from http://www.insecure.org/nmap/nmap_doc.html.
243. Antirez, hping2(8) - Linux man page: Description, Hping2 Commands, from <http://www.hping.org/manpage.html>.
244. Chris McNab, (2008), Third Party IP Network Scanning Methods, Available from <http://www.codewalkers.com/c/a/Server-Administration/Third-Party-IP-Network-Scanning-Methods/>.
245. Thierry Lagarde , AutoScan Network, Available from http://autoscan-network.com/index.php?option=com_content&task=view&id=48&Itemid=32.
246. Onion Routing, Available from <http://dictionary.zdnet.com/definition/onion+routing.html>.
247. Van Geelkerken F.W.J, (2006), Digital Mixing (MIX nets), Available from <http://www.iusmentis.com/society/privacy/remailers/onionrouting/>.
248. Keith J. Jones, Mike Shema, & Bradley C. Johnson, Vulnerability Scanners, from www.foundstone.com/pdf/books/AntiHackerSample.pdf.
249. Examining Port Scan Methods- Analysing Audible Techniques, from <http://www.in-f-or.it/informatica/docs/portscan.pdf>.
250. IMS General Web Services Security Profile, http://www.imsglobal.org/gws/gwsv1p0/imsgws_securityProfv1p0.html.
251. Beware!: War dialing, from http://www.castlecops.com/a1361-War_dialing.html.
252. Simson L. Garfinkel, Automatic Parity Detection, from <http://archive.cert.uni-stuttgart.de/archive/bugtraq/1998/12/msg00215.html>.
253. Lance Mueller, CREATE A REVERSE SSH TUNNEL, <http://www.lancemueler.com/blog/Create%20Reverse%20SSH%20to%20reach%20servlet%20inside%20firewall.pdf>.

254. Avi Kak, (2010), Port Scanning, Vulnerability Scanning, Packet Sniffing, and Intrusion Detection, <http://cobweb.ecn.purdue.edu/~kak/compsec/NewLectures/Lecture23.pdf>.
255. Renaud Deraison, Ron Gula, and Todd Hayton, (2009), Passive Vulnerability Scanning Introduction, http://nessus.org/whitepapers/passive_scanning_tenable.pdf.
256. Cheng Guang, TCP Analysis Based on Flags, <http://www.nordu.net/development/2nd-cnnw/tcp-analysis-based-on-flags.pdf>.
257. Cheng Tang & Jonathan Gossels, (1999), Wardialing: Practical Advice to Understand Your Exposure, <http://www.systemexperts.com/assets/tutors/wardial0299.pdf>.
258. Network Security Library, from http://www.windowsecurity.com/whitepapers/misc/Examining_port_scan_methods__Analyzing_Audible_Te_.pdf.
259. Lance Cottrell, Anonymizer Limitations: Logs, from http://www.livinginternet.com/i/is_anon.htm.
260. Michel Leconte, (2006), Network security consulting, from <http://www.activsupport.com/Small-Business-Network-Security-Soluti>.
261. Angry IP Scanner, from <http://angryip.org/w/Download>.
262. SolarWinds Engineer's Toolset, from <http://downloads.solarwinds.com/solarwinds/Release/Toolset/ZP-Toolset/ZP-Toolset-01.html>.
263. Colasoft Ping Tool, from http://www.colasoft.com/download/products/download_ping_tool.php.
264. PacketTrap MSP, from <http://www.packettrap.com/download?hsCtaTracking=e95ec5b5-069f-4cd5-962c-9c0e6e32a6da%7C072dfe23-353f-46c2-9ab0-1a27d39c01f1>.
265. Visual Ping Tester - Standard, from <http://www.pingtester.net>.
266. Ping Sweep (Integrated into WhatsupGold), from http://www.whatsupgold.com/products/download/network_management.aspx?k_id=ping-sweep-tool.
267. Ping Scanner Pro, from <http://www.digilextechnologies.com>.
268. Network Ping, from http://www.greenline-soft.com/product_network_ping/index.aspx.
269. Ultra Ping Pro, from <http://ultraping.webs.com/downloads.htm>.
270. Ping Monitor, from <http://www.niliand.com>.
271. PingInfoView, from http://www.nirsoft.net/utils/multiple_ping_tool.html.
272. Pinkie, from <http://www.ipuptime.net/category/download/>.
273. Colasoft Packet Builder, from http://www.colasoft.com/download/products/download_packet_builder.php.
274. NetScanTools Pro, from <http://www.netscantools.com/nstprodmorerequestform.html>.
275. PRTG Network Monitor, from <http://www.paessler.com/download/prtg>.
276. Global Network Inventory Scanner, from http://www.magnetosoft.com/products/global_network_inventory/gni_features.htm.
277. Net Tools, from <http://mabsoft.com/nettools.htm>.
278. SoftPerfect Network Scanner, from <http://www.softperfect.com/products/networkscanner/>.
279. IP Tools, from <http://www.ks-soft.net/ip-tools.eng/downpage.htm>.
280. Advanced Port Scanner, from <http://www.radmin.com/download/previousversions/portscanner.php>.
281. MegaPing, from http://www.magnetosoft.com/products/megaping/megaping_features.htm.
282. Netifera, from <http://netifera.com>.
283. Network Inventory Explorer, from <http://www.10-strike.com/networkinventoryexplorer/download.shtml>.

284. Free Port Scanner, from http://www.nsauditor.com/network_tools/free_port_scanner.html#.UWJRvqLzvrw.
285. ID Serve, from <http://www.grc.com>.
286. Netcraft, from <http://toolbar.netcraft.com>.
287. Netcat, from <http://sourceforge.net/projects/netcat/files/latest/download?source=files>.
288. GFI LanGuard, from <http://www.gfi.com/downloads/mirrors.aspx?pid=lanss>.
289. SAINT, from <http://www.saintcorporation.com/products/software/saintScanner.html>.
290. Retina CS, from <http://www.beyondtrust.com/Landers/TY-Page-RetinaCSCommunity/index.html>.
291. OpenVAS, from <http://www.openvas.org>.
292. Core Impact Professional, from <http://www.coresecurity.com>.
293. Security Manager Plus, from <http://www.manageengine.com/products/security-manager/download.html>.
294. Nmap, from <http://www.rapid7.com/products/nmap/compare-downloads.jsp>.
295. Shadow Security Scanner, from <http://www.safety-lab.com/en/download.htm>.
296. QualysGuard, from <http://www.qualys.com>.
297. Nsauditor Network Security Auditor, from http://www.nsauditor.com/network_security/network_security_auditor.html#.UWKEEx6Lzvrw.
298. Security Auditor's Research Assistant (SARA), from <http://www-arc.com/sara/>.
299. LANsurveyor, from <http://www.solarwinds.com/register/MoreSoftware.aspx?External=false&Program=17592&c=70150000000PjNE>.
300. OpManager, from <http://www.manageengine.com/network-monitoring/download.html>.
301. NetworkView, from <http://www.networkview.com/html/download.html>.
302. The Dude, from <http://www.mikrotik.com/thedude>.
303. LANState, from <http://www.10-strike.com/lanstate/download.shtml>.
304. HP Network Node Manager i software, from <http://www8.hp.com/us/en/software-solutions/software.html?compURI=1170657#>.
305. FriendlyPinger, from <http://www.kilievich.com/fpinger/download.htm>.
306. NetMapper, from <http://www.opnet.com>.
307. Ipsonar, from <http://www.lumeta.com/product/product.html>.
308. NetBrain Enterprise Suite, from <http://www.netbraintech.com/instant-trial/>.
309. CartoReso, from <http://cartoreso.campus.ecp.fr>.
310. Spiceworks-Network Mapper, from <http://www.spiceworks.com/download/>.
311. Switch Center Enterprise, from <http://www.lan-secure.com/downloads.htm#network>.
312. NetCrunch, from <http://www.adremsoft.com/demo/download-product.php?product=nc7&file=NCServer7Premium.exe>.
313. Proxy Workbench, from <http://proxyworkbench.com/>.
314. Proxifier, from <http://www.proxifier.com/download.htm>.
315. Proxy Switcher, from <http://www.proxyswitcher.com/>.
316. SocksChain, from <http://ufasoft.com/socks/>.
317. TOR (The Onion Routing), from <https://www.torproject.org/download/download>.

318. Proxy, from <http://www.analogx.com/contents/download/Network/proxy/Freeware.htm>.
319. Proxy Commander, from <http://www.dlao.com/proxycmd/>.
320. Protoport Proxy Chain, from <http://www.protoport.com>.
321. Proxy Tool Windows App, from <http://webproxylist.com/proxy-tool-windows-app/>.
322. Proxy+, from <http://www.proxyplus.cz/>.
323. Gproxy, from <http://gpass1.com/gproxy.php>.
324. FastProxySwitch, from <http://www.affinity-tools.com/fps/>.
325. Fiddler, from <http://www.fiddler2.com/fiddler2/version.asp>.
326. ProxyFinder Enterprise, from <http://www.proxy-tool.com>.
327. Socks Proxy Scanner, from <http://www.mylanviewer.com>.
328. ezProxy, from <https://www.oclc.org/ezproxy/download.en.h.html>.
329. Charles, from <http://www.charlesproxy.com/>.
330. JAP Anonymity and Privacy, from http://anon.inf.tu-dresden.de/win/download_en.html.
331. UltraSurf, from <http://www.ultrasurf.us>.
332. CC Proxy Server, from <http://www.youngzsoft.net/ccproxy/proxy-server-download.htm>.
333. WideCap, from <http://widecap.ru>.
334. FoxyProxy Standard, from <https://addons.mozilla.org>.
335. ProxyCap, from <http://www.proxycap.com>.
336. Super Network Tunnel, from <http://www.networktunnel.net>.
337. HTTP-Tunnel, from <http://www.http-tunnel.com>.
338. Bitvise, from <http://www.bitvise.com>.
339. Psiphon, from <http://psiphon.ca>.
340. Your-Freedom, from <http://www.your-freedom.net>.
341. Just Ping, from <http://www.just-ping.com>.
342. WebSitePulse, from <http://www.websitepulse.com>.
343. G-Zapper, from <http://www.dummysoftware.com/gzapper.html>.
344. Mowser, from <http://www.mowser.com>.
345. Spotflux, from <http://www.spotflux.com>.
346. Anonymous Web Surfing Tool, from <http://www.anonymous-surfing.com>.
347. U-Surf, from <http://ultimate-anonymity.com>.
348. Hide Your IP Address, from <http://www.hideyouripaddress.net>.
349. WarpProxy, from <http://silent-surf.com>.
350. Anonymizer Universal, from <http://www.anonymizer.com>.
351. Hope Proxy, from <http://www.hopeproxy.com>.
352. Guardster, from <http://www.guardster.com>.
353. Hide My IP, from <http://www.privacy-pro.com/features.html>.

Module 04: Enumeration

354. rpcinfo, from <http://www.usoft.spb.ru/commands/rpcinfo/>.
355. RPCCLIENT, from <http://www.sarata.com/manpages/man1/rpcclient.html>.

356. Enumeration, from <http://www.edenofire.com/tutes/hack.php>.
357. smtp-user-enum User Documentation, from <http://pentestmonkey.net/tools/user-enumeration/smtp-user-enum>.
358. Chris Gates, (2006), Windows Enumeration: USER2SID & SID2USER, from <http://www.windowsecurity.com/whitepaper/Windows-Enumeration-USER2SID-SID2USER.html>.
359. What is SNMP?, from <http://www.wtcs.org/snmp4tpc/snmp.htm>.
360. SNMP, from http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/snmp.htm#xtocid5.
361. SNMPForDummies, from <http://wiki.outboundindex.net/SNMPForDummies>.
362. Jan van Oorschot, Jeroen Wortelboer and Dirk Wisse, (2001), SNMP - The Mission Statement, <http://www.securityfocus.com/infocus/1301>.
363. rpcinfo(1M), from <http://docs.hp.com/en/B2355-90692/rpcinfo.1M.html>.
364. GRAPE- INFO- DOT- COM, from <http://www.grape-info.com>.
365. Joris Evers, (2006), AT&T hack exposes 19,000 identities, from http://news.cnet.com/2100-1029_3-6110765.html.
366. SNMP from http://www.iss.net/security_center/advice/Reference/Networking/SNMP/default.htm.
367. Simple Network Management Protocol (SNMP), from <http://www.cisco.com/en/US/docs/internetworking/technology/handbook/SNMP.html>.
368. Linux / Unix finger command, from <http://www.computerhope.com/unix/ufinger.htm>.
369. Chris Gates, (2006), Windows Enumeration: USER2SID & SID2USER <http://www.windowsecurity.com/whitepapers/Windows-Enumeration-USER2SID-SID2USER.html>.
370. SuperScan, from <http://www.mcafee.com/us/downloads/free-tools/superscan.aspx>.
371. Hyena, from http://www.systemtools.com/hyena/trial_download.htm.
372. Winfingerprint, from <http://www.winfingerprint.com>.
373. NetBIOS Enumerator, from <http://nbtenum.sourceforge.net/>.
374. PsTools, from <http://technet.microsoft.com/en-us/sysinternals/bb897553.aspx>.
375. OpUtils, from <http://www.manageengine.com/products/oputils/download.html>.
376. SolarWind's IP Network Browser, from <http://www.solarwinds.com/engineers-toolset/ip-network-browser.aspx>.
377. Getif, from <http://www.wtcs.org/snmp4tpc/getif.htm>.
378. OiDVIEW SNMP MIB Browser, from <http://www.oidview.com/mibbrowser.html>.
379. iReasoning MIB Browser, from <http://ireasoning.com/mibbrowser.shtml>.
380. SNScan, from <http://www.mcafee.com/us/downloads/free-tools/snscan.aspx>.
381. SNMP Scanner, from <http://www.secure-bytes.com/SNMP+Scanner.php>.
382. SNMP Informant, from <http://www.snmp-informant.com/>.
383. Net-SNMP, from <http://net-snmp.sourceforge.net/download.html>.
384. Nsauditor Network Security Auditor, from http://www.nsauditor.com/network_security/network_security_auditor.html#.UV7LH5NHLZ4.
385. Spiceworks, from <http://www.spiceworks.com/free-snmp-network-management-software/>.
386. Enum4linux, from <http://labs.portcullis.co.uk/application/enum4linux/>.
387. Softerra LDAP Administrator, from <http://www.ldapadministrator.com/>.
388. JXplorer, from <http://www.jxplorer.org/>.
389. LDAP Admin Tool, from <http://www.ldapsoft.com/ldapbrowser/ldapadmintool.html>.

390. LDAP Account Manager, from <https://www.ldap-account-manager.org/lamcms/>.
391. LEX - The LDAP Explorer, from <http://www.ldapexplorer.com/>.
392. LDAP Admin, from <http://www.ldapadmin.org/>.
393. Active Directory Explorer, from <http://technet.microsoft.com/en-us/sysinternals/bb963907.aspx>.
394. LDAP Administration Tool, from <http://sourceforge.net/projects/ldap-at/>.
395. LDAP Search, from <http://securityxploded.com/ldapsearch.php>.
396. Active Directory Domain Services Management Pack, from <http://www.microsoft.com/en-us/download/details.aspx?id=21357>.
397. LDAP Browser/Editor, from <http://www.novell.com/coololutions/tools/13765.html>.
398. NSLookup, from <http://www.kloth.net/services/nslookup.php>.

Module 05: System Hacking

399. Why Keyloggers are extremely dangerous?, from <http://gamecreator.hubpages.com/hub/Why-Keyloggers-are-extremely-dangerous>.
400. Steganography in Depth, from <http://www.crcnetbase.com/doi/abs/10.1201/9780203504765.ch4>.
401. Detecting spoofed packets, from <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1194882>.
402. NTLM Authentication in Java, from <http://www.luigidragone.com/software/ntlm-authentication-in-java/>.
403. A Tutorial Review on Steganography, from http://www.jiit.ac.in/jiit/ic3/IC3_2008/IC3-2008/APP2_21.pdf.
404. network scanning, from <http://searchmidmarketsecurity.techtarget.com/definition/network-scanning>.
405. Ricky M. Magalhaes, (2004), Using passwords as a defense mechanism to improve Windows security, from www.windowsecurity.com/articles/Passwords_Improve_Windows_Security_Part2.html.
406. Piazza & Peter, (2002), Hybrid threats have rosy future: attacks that combine virus ..., http://findarticles.com/p/articles/mi_hb6380/is_200207/ai_n25618875?tag=content;col1.
407. Andreas Westfeld and Andreas Pfitzmann, Attacks on Steganographic Systems, citeseexr.ist.psu.edu/.
408. DaijiSanai and HidenobuSeki, (2004), Optimized Attack for NTLM2 Session Response <http://www.blackhat.com/presentations/bh-asia-04/bh-jp-04-pdfs/bh-jp-04-seki.pdf>.
409. Zhi Wang, Xuxian Jiang, Weidong Cui, and Xinyuan Wang, Countering Persistent Kernel Rootkits Through Systematic Hook Discovery, <http://research.microsoft.com/en-us/um/people/wdcui/papers/hookmap-raid08.pdf>.
410. Elia Florio, When Malware Meets Rootkits, <http://www.symantec.com/avcenter/reference/when.malware.meets.rootkits.pdf>.
411. Peter Piazza, (2002), SMO: Tech Talk, from <http://www.securitymanagement.com/library/001272.html>.
412. Brute force attack - Wikipedia, the free encyclopedia, from http://en.wikipedia.org/wiki/Brute_force_attack.
413. Talk:Brute force attack – Bvio, from http://bvio.ngic.re.kr/Bvio/index.php/Talk:Brute_force_attack.
414. Passwords, from <http://searchsecurity.techtarget.com/searchSecurity/downloads/HackingforDummiesCh07.pdf>.
415. Authernative, Inc. | Products | FAQs, from <http://www.authernative.com/faqs.shtml>.
416. CIAC Notes, from <http://www.ciac.org/ciac/notes/Notes03a.shtml>.
417. Path: newshost.uwo.caluwovax.uwo.ca!mnevilles From: mnevilles@uwovax ..., from <http://www.uwo.ca/its/doc/newsletters/InTouch/vol1-9495/win18.txt>.
418. The Hack FAQ: Password Basics, from <http://www.nmrc.org/pub/faq/hackfaq/hackfaq-04.html>.

419. Luigi Dragone, NTLM Authentication in Java, from <http://www.luigidragone.com/networking/ntlm.html>.
420. Hardening the Base Windows 2000 Server, from <http://www.microsoft.com/technet/security/prodtech/windows2000/secwin2k/swin2k06.mspx>.
421. Bill Wall, Sunbelt TECH BRIEFING, from <http://www.stratvantage.com/security/ntpass.htm>.
422. Security Options, from <http://www.microsoft.com/technet/security/topics/serversecurity/tcg/tcgch05n.mspx>.
423. Technical Explanation of Network SMB Capture, from http://ebook.coolersky.com/hack/lc5.04_doc/smb_capture.html.
424. Detecting Alternate Data Streams, from <http://www.windowsitpro.com/Article/ArticleID/16189/16189.html>.
425. Bojan Smoijver, Linux Today - ZDNet Australia: Threats Move Beyond Linux to Windows, <http://www.linuxtoday.com/security/2002121100426SCSVNT>.
426. Neohapsis Archives - NTBugtraq - Proposal for protection from ..., from <http://archives.neohapsis.com/archives/ntbugtraq/2000-q2/0245.html>.
427. Russell Kay, (2006), Sidebar: A Simple Rootkit Example, http://www.computerworld.com/securitytopics/security/hacking/story/0,10801,108116,00.html?from=story_package.
428. Russell Kay, (2006), Rootkits offer the lure of total control, from <http://www.techworld.com/security/features/index.cfm?featureid=2219>.
429. Paladin Networks, from <http://www.paladion.net/media/insights/ihfaq.htm>.
430. NTFS Streams - Everything you need to know (demos and tests included), from <http://www.diamondcs.com.au/streams/streams.htm>.
431. H. Carvey, (2002), The Dark Side of NTFS (Microsoft's Scarlet Letter), from <http://www.infosecwriters.com/texts.php?op=display&id=53>.
432. Stegonography (a secretly hidden coding that dates back to ancient ...), from http://www.wordinfo.info/words/index/info/view_unit/3403/?letter=S&spage=9.
433. Ravindranath AV, Steganography: Hiding Data in Images, from <http://www.asptoday.com/Content.aspx?id=2347>.
434. Paul Robertson, (2005), CS 450 Homework 4, from <http://www.cs.umb.edu/~paulr/CS450/assignments/ass4.html>.
435. NBTdeputy (v 1.0.1), from <http://www.securityfriday.com/Topics/winxp1.html>.
436. Sir Dystic, (2002), NBName, from <http://www.securityfocus.com/comments/tools/1670/12751/threaded>.
437. Fred B. Schneider, Authentication, from <http://www.cs.cornell.edu/Courses/cs513/2000sp/NL10.html>.
438. CS513: System Security - Topic Outline, from <http://www.cs.cornell.edu/courses/cs513/2005fa/02.outline.html>.
439. Security Options from <http://www.microsoft.com/technet/security/guidance/serversecurity/tcg/tcgch05n.mspx>.
440. Hardening the Base Windows 2000 Server, [http://technet.microsoft.com/hi-in/library/cc751216\(en-us\).aspx](http://technet.microsoft.com/hi-in/library/cc751216(en-us).aspx).
441. Brute force attack, from http://www.reference.com/browse/wiki/Brute_force_attack.
442. What is pwdump2, from http://www.bindview.com/Services/razor/Utilities/Windows/pwdump2_readme.cfm.
443. Derogee, Steganography and Steganalysis, from <http://www.liacs.nl/home/tmoerlan/privtech.pdf>.

444. Techworld.com - Rootkits offer the lure of total control, from <http://www.techworld.com/websecurity/features/index.cfm?featureid=2219&pagetype=samecatsamechan>.
445. Dasmith, Software Analysis, from <http://www.unc.edu/~smithdr/inls187/sr.html>.
446. Hacking Windows-Specific Services, from <http://awkwardalliteration.com/ebooks/Computers/Hacking%20Exposed-%20Windows%202003%20Chapter%205.pdf>.
447. Ricky M. Magalhaes, (2003), Using passwords as a defense mechanism to improve Windows security, from http://www.windowsecurity.com/articles/Passwords_Improve_Windows_Security_Part2.html.
448. Winrtgen, from <http://www.oxid.it/projects.html>.
449. RainbowCrack, from <http://project-rainbowcrack.com/index.htm#download>.
450. Elcomsoft Distributed Password Recovery, from <http://www.elcomsoft.com/edpr.html>.
451. Securityoverride, from <http://securityoverride.org/default-password-list>.
452. Cirt, from <http://cirt.net>.
453. Default-password, from <http://default-password.info>.
454. Defaultpassword, from <http://www.defaultpassword.us>.
455. Passwordsdatabase, from <http://www.passwordsdatabase.com>.
456. W3dt, from <https://w3dt.net/tools/defaultpasswords/>.
457. Virus, from http://www.virus.org/default_passwds.
458. Open-sez.me, from <http://open-sez.me/passwd.htm>.
459. Routerpasswords, from <http://www.routerpasswords.com/>.
460. Fortyboundhead, from http://www.fortyboundhead.com/tools_dpw.asp.
461. pwdump7, from http://www.tarasco.org/security/pwdump_7/.
462. fgdump, from <http://www.foofus.net/~fizzgig/fgdump/>.
463. L0phtCrack, from <http://www.l0phtcrack.com/download.html>.
464. Ophcrack, from <http://ophcrack.sourceforge.net/download.php>.
465. RainbowCrack, from <http://project-rainbowcrack.com/index.htm#download>.
466. Password Unlocker Bundle, from <http://www.passwordunlocker.com/password-recovery-bundle.html>.
467. Proactive System Password Recovery, from <http://www.elcomsoft.com/pspr.html>.
468. John the Ripper, from <http://www.openwall.com/john/>.
469. Windows Password Cracker, from <http://www.windows-password-cracker.com/download.html>.
470. WinPassword, from <http://lastbit.com/ntpsw/default.asp>.
471. Passware Kit Enterprise, from <http://www.lostpassword.com/kit-enterprise.htm>.
472. PasswordsPro, from <http://www.insidepro.com/eng/passwordspro.shtml>.
473. LSASecretsView, from http://www.nirsoft.net/utils/lsa_secrets_view.html.
474. LCP, from <http://www.lcpsoft.com/english/download.htm>.
475. Password Cracker, from <http://www.amlpages.com/pwdcrack.shtml>.
476. Kon-Boot, from <http://www.thelead82.com/kon-boot/konbootWIN.html>.
477. Windows Password Recovery Tool, from <http://www.windowspasswordsrecovery.com/>.
478. Hash Suite, from <http://hashsuite.openwall.net/download>.
479. SAMInside, from <http://www.insidepro.com/eng/saminside.shtml>.

480. Windows Password Recovery, from http://www.passcape.com/windows_password_recovery.
481. Password Recovery Bundle, from <http://www.top-password.com/password-recovery-bundle.html>.
482. krbpwguess, from <http://www.cquare.net/wp/tools/password-recovery/krbpwguess/>.
483. Windows Password Breaker Enterprise, from <http://www.recoverwindowspassword.com/windows-password-breaker.html>.
484. Rekeysoft Windows Password Recovery Enterprise, from <http://www.rekeysoft.com/reset-windows-password.html>.
485. Active@ Password Changer, from <http://www.password-changer.com/>.
486. Offline NT Password & Registry Editor, from <http://pogostick.net/~pnh/ntpasswd/>.
487. Windows Password Reset Kit, from <http://www.reset-windows-password.net/>.
488. Windows Password Recovery Tool, from <http://www.windowspasswordsrecovery.com/>.
489. ElcomSoft System Recovery, from <http://www.elcomsoft.com/esr.html#forgot administrator password>.
490. Trinity Rescue Kit, from http://trinityhome.org/Home/index.php?content=TRINITY_RESCUE_KIT_DOWNLOAD&front_id=12&lang=en&locale=en.
491. Windows Password Recovery Bootdisk, from <http://www.rixler.com/windows-password-recovery-bootdisk.htm>.
492. PasswordLastic, from <http://www.passwordlastic.com/windows-password-recovery-lastic>.
493. Stellar Phoenix Password Recovery, from <http://www.stellarinfo.com/password-recovery.htm>.
494. Windows Password Recovery Personal, from <http://www.windows-passwordrecovery.com/>.
495. Windows Administrator Password Reset, from <http://www.systoolsgroup.com/windows-administrator-password-reset.html>.
496. RemoteExec, from <http://www.isdecisions.com/products/remoteexec>.
497. PDQ Deploy, from <http://www.adminarsenal.com/download-pdq>.
498. DameWare NT Utilities, from http://www.dameware.ru/nt_utilities.html.
499. Spytech SpyAgent, from <http://www.spytech-web.com/spyagent.shtml>.
500. All In One Keylogger, from <http://www.relytec.com/>.
501. Ultimate Keylogger, from <http://www.ultimatekeylogger.com/download/>.
502. Advanced Keylogger, from <http://www.mykeylogger.com/perfect-keylogger>.
503. The Best Keylogger, from <http://www.thebestkeylogger.com/>.
504. SoftActivity Keylogger, from <http://www.softactivity.com/download-al.asp>.
505. Elite Keylogger, from <http://www.widestep.com/elite-keystroke-recorder-info>.
506. Powered Keylogger, from <http://www.mykeylogger.com/undetectable-keylogger/>.
507. StaffCop Standard, from <http://www.staffcop.com/download/>.
508. iMonitorPC, from <http://www.imonitorpc.com/>.
509. PC Activity Monitor Standard, from <http://www.pcacme.com/download.html>.
510. KeyProwler, from <http://keyprowler.com/download.aspx>.
511. Keylogger Spy Monitor, from <http://ematrixsoft.com/download.php?p=keylogger-spy-monitor-software>.
512. REFOG Personal Monitor, from <http://www.refog.com/personal-monitor.html>.
513. Actual Keylogger, from <http://www.actualkeylogger.com/download-free-key-logger.html>.
514. Spyector, from <http://www.spyector.com/download.html>.

515. KidLogger, from <http://kidlogger.net/download.html>.
516. PC Spy Keylogger, from <http://www.pc-spy-keylogger.com>.
517. Revealer Keylogger, from <http://www.logixoft.com/free-keylogger-download>.
518. Spy Keylogger, from <http://www.spy-key-logger.com/download.html>.
519. Actual Spy, from <http://www.actualspy.com/download.html>.
520. SpyBuddy® 2013, from <http://www.exploreanywhere.com/products/spybuddy/>.
521. Amac Keylogger, from <http://www.amackeylogger.com/>.
522. Aobo Mac OS X KeyLogger, from <http://www.keylogger-mac.com/>.
523. Perfect Keylogger for Mac, from <http://www.blazingtools.com>.
524. Award Keylogger for Mac, from <http://www.award-soft.com/content/view/275/136>.
525. Mac Keylogger, from http://www.award-soft.com/Mac_Keylogger/.
526. REFOG Keylogger for MAC, from <http://www.refog.com/mac-keylogger.html>.
527. KidLogger for MAC, from <http://kidlogger.net/download.html>.
528. MAC Log Manager, from <http://www.keylogger.in/keylogger/maclogmanager.html>.
529. logkext, from <https://code.google.com/p/logkext/>.
530. Keyboard Spy, from <http://alphaomega.software.free.fr/keyboardspy/Keyboard%20Spy.html>.
531. FreeMacKeylogger, from <http://www.hwsuite.com/free-mac-keylogger/>.
532. KeyGrabber, from <http://www.keydemon.com>.
533. KeyGhost, from <http://www.keyghost.com>.
534. Activity Monitor, from <http://www.softactivity.com/download.asp>.
535. Remote Desktop Spy, from <http://www.global-spy-software.com/download.php>.
536. SSPro, from <http://www.gpsoftdev.com/download-monitoring-software/>.
537. RecoveryFix Employee Activity Monitor, from <http://www.recoveryfix.com/download-employee-monitoring.html>.
538. Employee Desktop Live Viewer, from <http://www.nucleustechologies.com/download-employee-desktop-live-viewer.php>.
539. NetVizor, from <http://www.netvizor.net/download.htm>.
540. Net Spy Pro, from <http://www.net-monitoring-software.com/windows/trial.html>.
541. REFOG Employee Monitor, from <http://www.refog.com/employee-computer-monitoring-software.html>.
542. OsMonitor, from <http://www.os-monitor.com/download.htm>.
543. LANVisor, from <http://www.lanvisor.com/download.htm>.
544. Work Examiner Standard, from <http://www.workexaminer.com/download.html>.
545. Power Spy, from <http://ematrixsoft.com/index.php>.
546. eBLASTER, from http://www.spectorsoft.com/products/eBlaster_Windows/index.asp?source=nav-hs-eBwin.
547. Imonitor Employee Activity Monitor, from <http://www.employee-monitoring-software.cc/>.
548. Employee Monitoring, from <http://www.employeemonitoring.net/download.asp>.
549. OsMonitor, from <http://www.os-monitor.com/download.htm>.
550. Ascendant NFM, from <http://www.ascendant-security.com/download.shtml>.
551. Spylab WebSpy, from <http://www.spylab.org/download.htm>.

552. Personal Inspector, from <http://www.spyarsenal.com/personal-inspector/>.
553. CyberSpy, from <http://www.cyberspysoftware.com/download.html>.
554. AceSpy, from <http://www.acespy.com/features.html>.
555. EmailObserver, from http://www.softsecurity.com/prod_D7_more.html.
556. Net Nanny Home Suite, from http://www.netnanny.com/products/netnanny_home_suite/detail/technical.
557. Aobo Filter for PC, from <http://www.aobo-porn-filter.com/downloads>.
558. CyberSieve, from <http://www.softforyou.com/cs-download.php>.
559. Child Control, from <http://salfeld.com/download/child-control/index.html>.
560. SentryPC, from <http://www.sentrypc.com/trial.htm>.
561. iProtectYou Pro, from <http://www.softforyou.com/ip-index.html>.
562. K9 Web Protection, from <http://www1.k9webprotection.com/getk9/download-software>.
563. Verity Parental Control Software, from <http://www.nchsoftware.com/childmonitoring/index.html>.
564. Profil Parental Filter, from <http://www.profiltechnology.com/en/home/profil-parental-filter>.
565. PC Pandora, from <http://www.pcpandora.com/download/>.
566. KidsWatch, from <http://www.kidswatch.com/>.
567. SoftActivity TS Monitor, from <http://www.softactivity.com/downloadtsm.aspx>.
568. Desktop Spy, from <http://www.spyarsenal.com/download.html>.
569. IcyScreen, from <http://www.16software.com/icyscreen/screenshots.php>.
570. Spector Pro, from http://www.spectorsoft.com/products/SpectorPro_Windows/index.asp?source=nav-hs-ProWin.
571. PC Tattletale, from <http://www.pctattletale.com/>.
572. Computer Screen Spy Monitor, from <http://www.mysuperspy.com/download.htm>.
573. PC Screen Spy Monitor, from <http://ematrixsoft.com/download.php?p=pc-screen-spy-monitor-software>.
574. Kahlown Screen Spy Monitor, from <http://www.lesoftrejion.com/>.
575. Guardbay Remote Computer Monitoring Software, from <http://www.guardbay.com>.
576. HT Employee Monitor, from <http://www.hidetools.com/employee-monitor.html>.
577. Spy Employee Monitor, from <http://www.spysw.com/employee-monitor-software.htm>.
578. USB Spy, from <http://www.everstrike.com/usb-monitor/>.
579. USB Monitor, from <http://www.hhdsoftware.com/usb-monitor>.
580. USB Grabber, from <http://usbgrabber.sourceforge.net/>.
581. USBTrace, from http://www.sysnucleus.com/usbtrace_download.html.
582. USBDevview, from http://www.nirsoft.net/utils/usb_devices_view.html.
583. Advanced USB Port Monitor, from <http://www.aggsoft.com/usb-port-monitor.htm>.
584. USB Monitor Pro, from <http://www.usb-monitor.com/>.
585. USB Activity Monitoring Software, from <http://www.datadoctor.org/partition-recovery/downloads.html>.
586. Stealth iBot Computer Spy, from <http://www.brickhousesecurity.com/product/stealth+ibot+computer+spy.do>.
587. KeyCarbon USB Hardware Keylogger, from <http://www.spywaredirect.net/keycarbon-usb.html>.
588. USB 2GB Keylogger, from http://dijj.com/KL2-Keylogger-2GB-USB-Hardware-keelog/prod_24.html.

589. Spy Voice Recorder, from <http://www.mysuperspy.com/recorder.htm>.
590. Sound Snooper, from <http://www.sound-snooper.com/en/download.php>.
591. WebCam Recorder, from <http://webcamrecorder.com/>.
592. WebcamMagic, from <http://www.robomagic.com/webcammagic.htm>.
593. MyWebcam Broadcaster, from <http://www.eyespyfx.com/broadcast.php>.
594. I-Can-See-You, from <http://www.internetsafetysoftware.com>.
595. Digi-Watcher, from <http://www.digi-watcher.com/>.
596. NET Video Spy, from <http://www.sarbash.com/download.shtml>.
597. Eyeline Video Surveillance Software, from <http://www.nchsoftware.com/surveillance/index.html>.
598. Capturix VideoSpy, from <http://www.capturix.com/default.asp?target=consumer&product=cvs>.
599. WebCam Looker, from <http://felenasoft.com/webcamlooker/en/>.
600. SecuritySpy, from <http://www.bensoftware.com/securityspy/download.html>.
601. iSpy, from <http://www.ispyconnect.com/download.aspx>.
602. Printer Activity Monitor, from <http://www.redline-software.com/eng/products/pam/>.
603. Print Monitor Pro, from <http://www.spyarsenal.com/printer-monitoring-software/print-monitor-pro/>.
604. Accurate Printer Monitor, from <http://www.aggsoft.com/printer-monitor.htm>.
605. Print Censor Professional, from <http://useulsoft.com/print-censor/#.UWPW8JNHLZ4>.
606. All-Spy Print, from <http://www.all-spy.com/all-spy-print.html>.
607. O&K Print Watch, from <http://www.prnwatch.com/okpw.html>.
608. Print Job Monitor, from <http://www.imonitorsoft.com/product-print-job-monitor.htm>.
609. PrintTrak, from <http://www.lygil.com/printtrak/printtrak.htm>.
610. Printer Admin - Copier Tracking System, from <http://www.printeradmin.com/copy-management.htm>.
611. Print Inspector, from <http://www.softperfect.com/products/pinspector/>.
612. Print365, from <http://krawasoft.com/index.html>.
613. Mobile Spy, from <http://www.phonespysoftware.com/>.
614. VRS Recording System, from <http://www.nch.com.au/vrs/index.html>.
615. Modem Spy, from <http://www.modemspy.com/en/download.php>.
616. MobiStealth Cell Phone Spy, from <http://www.mobistealth.com/mobile-phone-spy-software>.
617. SPYPhone GOLD, from <http://spyera.com/products/spyphone-gold-internet>.
618. SpyPhoneTap, from <http://www.spyphetap.com/>.
619. FlexiSPY OMNI, from <http://www.flexispy.com/en/flexispy-omni-spy-app-cell-phone.htm>.
620. SpyBubble, from <http://www.spypbble.com/cell-phone-spy.php>.
621. MOBILE SPY, from <http://www.mobile-spy.com/>.
622. StealthGenie, from <http://www.stealthgenie.com/>.
623. CellSPYExpert, from <http://www.cellspyexpert.com/>.
624. SPYPhone, from <http://spyera.com/products/spy-phone-basic-internet>.
625. EasyGPS, from <http://www.easygps.com/>.
626. FlexiSPY PRO-X, from <http://www.flexispy.com/spyphone-call-interceptor-gps-tracker-symbian.htm>.
627. GPS TrackMaker Professional, from <http://www.trackmaker.com/dwlpage.php>.
628. MOBILE SPY, from <http://www.mobile-spy.com/>.

629. World-Tracker, from <http://www.world-tracker.com/v4/>.
630. ALL-in-ONE Spy, from <http://www.thespyphone.com/allinone.html>.
631. Trackstick, from <http://www.trackstick.com/download.html>.
632. MobiStealth Pro, from <http://www.mobistealth.com>.
633. mSpy, from <http://www.buymspy.com/>.
634. GPS Retriever, from http://www.mobilebugstore.com/Blackberry_gps_retriver.aspx.
635. Zemana AntiLogger, from <http://www.zemana.com/Download.aspx>.
636. Anti-Keylogger, from <http://www.anti-keyloggers.com/>.
637. PrivacyKeyboard, from <http://www.anti-keylogger.com/products/privacykeyboard/overview.html#download>.
638. DefenseWall HIPS, from <http://www.softsphere.com/programs/>.
639. KeyScrambler, from <http://www.qfxsoftware.com/download.htm>.
640. I Hate Keyloggers, from <http://dewasoft.com/privacy/i-hate-keyloggers.htm>.
641. SpyShelter STOP-LOGGER, from <http://www.spyshelter.com/download-spyshelter>.
642. DataGuard AntiKeylogger Ultimate, from <http://www.maxsecuritylab.com/dataguard-anti-keylogger/download-anti-keylogger.php>.
643. PrivacyKeyboard, from <http://www.privacykeyboard.com/privacy-keyboard.html>.
644. Elite Anti Keylogger, from <http://www.elite-antikeylogger.com/free-download.html>.
645. CoDefender, from <https://www.encassa.com/downloads/default.aspx>.
646. PC Tools Spyware Doctor, from <http://www.pctools.com/spyware-doctor/>.
647. SUPERAntiSpyware, from <http://superantispyware.com/index.html>.
648. Spyware Terminator 2012, from <http://www.pcrx.com/spywareterminator/>.
649. Ad-Aware Free Antivirus+, from http://www.lavasoft.com/products/ad_aware_free.php.
650. Norton Internet Security, from <http://in.norton.com/downloads-trial-norton-internet-security>.
651. SpyHunter, from <http://www.enigmasoftware.com/products/>.
652. Kaspersky Internet Security 2013, from <http://www.kaspersky.com/internet-security-free-trial>.
653. SecureAnywhere Complete 2012, from http://www.webroot.com/En_US/consumer-products-secureanywhere-complete.html.
654. MacScan, from <http://macscan.securemac.com/>.
655. Spybot – Search & Destroy, from <http://www.safer-networking.org/dl/>.
656. Malwarebytes Anti-Malware PRO, from http://www.malwarebytes.org/products/malwarebytes_pro/.
657. Fu, from <http://www.f-secure.com/v-descs/fu.shtml>.
658. KBeast, from <http://core.ipsecs.com/rootkit/kernel-rootkit/kbeast-v1/>.
659. Hacker Defender HxDef Rootkit, from <http://vishnuvalentino.com/hacking-tutorial/hacker-defender-hxdef-rootkit-tutorial-in-10-steps-nostalgia/>.
660. Stinger, from <http://www.mcafee.com/us/downloads/free-tools/how-to-use-stinger.aspx>.
661. UnHackMe, from <http://www.greatis.com/unhackme/download.htm>.
662. Virus Removal Tool, from <http://www.sophos.com/en-us/products/free-tools/virus-removal-tool.aspx>.
663. Hypersight Rootkit Detector, from <http://northsecuritylabs.com/>.
664. Avira Free Antivirus, from <http://www.avira.com/en/avira-free-antivirus>.
665. SanityCheck, from <http://www.resplendence.com/downloads>.

666. GMER, from <http://www.gmer.net/>.
667. Rootkit Buster, from http://downloadcenter.trendmicro.com/index.php?regs=NABU&clk=result_page&clkval=drop_list&catid=6&prodid=155.
668. Rootkit Razor, from <http://www.tizersecure.com/>.
669. RemoveAny, from <http://www.free-anti-spy.com/en/index.php>.
670. TDSSKiller, from <http://support.kaspersky.com/5350?el=88446>.
671. Prevx, from <http://www.prevx.com/freescan.asp>.
672. StreamArmor, from <http://securityxploded.com/streamarmor.php>.
673. ADS Spy, from <http://www.merijn.nu/programs.php#adsspy>.
674. ADS Manager, from <http://dmitrybrant.com/adsmanager>.
675. Streams, from <http://technet.microsoft.com/en-us/sysinternals/bb897440.aspx>.
676. AlternateStreamView, from http://www.nirsoft.net/utils/alternate_data_streams.html.
677. NTFS-Streams: ADS manipulation tool, from <http://sourceforge.net/projects/ntfs-ads/>.
678. Stream Explorer, from <http://www.rekenwonder.com/streamexplorer.htm#Streams>.
679. ADS Scanner, from <http://www.pointstone.com/products/ADS-Scanner/>.
680. RKDetector, from <http://www.rkdetector.com/>.
681. GMER, from <http://www.gmer.net/>.
682. HijackThis, from <http://www.trendmicro.com/us/security/products/index.html>.
683. SNOW, from <http://www.darkside.com.au/snow/index.html>.
684. QuickStego, from <http://quickcrypto.com/free-steganography-software.html>.
685. Hide In Picture, from <http://sourceforge.net/projects/hide-in-picture/>.
686. gifshuffle, from <http://www.darkside.com.au/gifshuffle/index.html>.
687. CryptoPix, from <http://www.briggssoft.com/cpix.htm>.
688. BMPSecrets, from <http://bmpsecrets.com/>.
689. OpenPuff, from http://embeddedsw.net/OpenPuff_Steganography_Home.html.
690. OpenStego, from <http://openstego.sourceforge.net/>.
691. PHP-Class StreamSteganography, from <http://www.phpclasses.org/package/6027-PHP-Store-and-hidden-information-in-PNG-images.html>.
692. Red JPEG, from <http://www.totalcmd.net/plugring/redjpeg.html>.
693. Steganography Studio , from <http://stegstudio.sourceforge.net/>.
694. Virtual Steganographic Laboratory (VSL), from <http://vsl.sourceforge.net/>.
695. wbStego, from <http://wbstego.wbailer.com/>.
696. Merge Streams, from <http://www.ntkernel.com/w&p.php?id=23>.
697. Office XML, from <http://www.irongeek.com/i.php?page=security/ms-office-stego-code>.
698. Data Stash, from http://www.skyjuicesoftware.com/software/ds_info.html.
699. FoxHole, from <http://foxhole.sourceforge.net>.
700. Xidie Security Suite, from <http://www.stegano.ro>.
701. StegParty, from <http://www.fasterlight.com>.
702. Hydan, from <http://www.crazyboy.com/hydan/>.

703. StegJ, from <http://sourceforge.net/projects/stegj/files/>.
704. StegoStick, from <http://stegostick.sourceforge.net/>.
705. SNOW, from <http://www.darkside.com.au/snow/index.html>.
706. OmniHide PRO, from <http://omnihide.com/>.
707. Our Secret, from <http://www.securekit.net/oursecret.htm>.
708. RT Steganography, from <http://rtstegvideo.sourceforge.net/>.
709. Masker, from <http://www.softpuls.com/masker/>.
710. Max File Encryption, from <http://www.softeza.com/fileencryption/>.
711. MSU StegoVideo, from http://www.compression.ru/video/stego_video/index_en.html.
712. BDV DataHider, from <http://www.bdvnotepad.com/products/bdv-datahider/>.
713. StegoStick, from <http://stegostick.sourceforge.net/>.
714. OpenPuff, from http://embeddedsw.net/OpenPuff_Steganography_Home.html.
715. Stegsecret, from <http://stegsecret.sourceforge.net/>.
716. PSM Encryptor, from <http://demo.powersoftmakers.com/psme.zip>.
717. DeepSound, from <http://jpinsoft.net/DeepSound/Download.aspx>.
718. Mp3stegz, from <http://mp3stegz.sourceforge.net/>.
719. MAXA Security Tools, from <http://www.maxa-tools.com/mst.php?lang=en>.
720. BitCrypt, from <http://bitcrypt.moshe-szweizer.com/>.
721. MP3Stego, from <http://www.petitcolas.net/fabien/steganography/mp3stego/>.
722. Hide4PGP, from <http://www.heinz-repp.onlinehome.de/>.
723. CHAOS Universal, from <http://safechaos.com/cu.htm>.
724. SilentEye, from <http://www.silenteeye.org/>.
725. QuickCrypto, from <http://www.quickcrypto.com/download.html>.
726. CryptArkan, from http://www.kuskov.com/component?option=com_remository&Itemid,30/func,fileinfo&id,1/.
727. StegoStick, from <http://stegostick.sourceforge.net/>.
728. Invisible Secrets 4, from <http://www.invisiblesecrets.com/>.
729. Folder Lock, from <http://www.newsoftwares.net/folderlock/>.
730. A+ Folder Locker, from <http://www.giantmatrix.com/products/aplus-folder-locker/>.
731. Toolwiz BSafe, from <http://www.toolwiz.com/products/toolwiz-bsafe/>.
732. Hide Folders 2012, from <http://fspro.net/hide-folders/>.
733. GiliSoft File Lock Pro, from <http://www.gilisoft.com/product-file-lock-pro.htm>.
734. Universal Shield, from <http://www.everstrike.com/shield.htm>.
735. WinMend Folder Hidden, from <http://www.winmend.com/folder-hidden/>.
736. Encrypted Magic Folders , from <http://www.pc-magic.com/des.htm#emf>.
737. QuickCrypto, from <http://www.quickcrypto.com/download.html>.
738. Max Folder Secure, from <http://www.maxfoldersecure.com/>.
739. Spam Mimic, from <http://www.spammimic.com/>.
740. Sams Big G Play Maker, from <http://www.scramdisk.clara.net/>.
741. Gargoyle Investigator™ Forensic Pro, from <http://wetstonetech.com/product/2/downloads>.

742. XStegsecret, from <http://stegsecret.sourceforge.net/>.
743. Stego Suite, from <http://www.wetstonetech.com/product/1/downloads>.
744. StegAlyzerAS, from <http://www.sarc-wv.com/products/stegalyzeras/>.
745. StegAlyzerRTS, from <http://www.sarc-wv.com/products/stegalyzerrts/>.
746. StegSpy, from <http://www.spy-hunter.com/stegspy>.
747. StegAlyzerSS , from <http://www.sarc-wv.com/products/stegalyzerss/>.
748. StegMark SDK, from <http://www.datamark.com.sg/downloads-sdk.htm>.
749. Steganography Studio, from <http://stegstudio.sourceforge.net/>.
750. Virtual Steganographic Laboratory (VSL), from <http://vsl.sourceforge.net/>.
751. Stegdetect, from <http://www.outguess.org/detection.php>.
752. Auditpol, from [http://technet.microsoft.com/en-us/library/cc755264\(v=ws.10\).aspx](http://technet.microsoft.com/en-us/library/cc755264(v=ws.10).aspx).
753. CCleaner, from <http://www.piriform.com/download>.
754. MRU-Blaster, from <http://www.brightfort.com/mrUBLASTER.html>.
755. Wipe, from <http://privacyroot.com/software/www/en/wipe.php>.
756. Tracks Eraser Pro, from <http://www.acesoft.net/features.htm>.
757. BleachBit, from <http://bleachbit.sourceforge.net/news/bleachbit-093>.
758. AbsoluteShield Internet Eraser Pro, from <http://www.internet-track-eraser.com/ineteraser.php>.
759. Clear My History, from <http://www.hide-my-ip.com/clearmyhistory.shtml>.
760. EvidenceEraser, from <http://www.evidenceeraser.com/>.
761. WinTools.net Professional, from <http://www.wintools.net/>.
762. RealTime Cookie & Cache Cleaner (RtC3) , from <http://www.kleinsoft.co.za/buy.html>.
763. AdvaHist Eraser, from <http://www.advacrypt.cjb.net/>.
764. Free Internet Window Washer, from http://www.eusing.com/Window_Washer/Window_Washer.htm.

Module 06: Trojans and Backdoors

765. Placing Backdoors through Firewalls, from <http://www.cyberwarzone.com/cyberwarfare/placing-backdoors-through-firewalls>.
766. A Deep Look into Netcat – The TCP/IP Swiss Army Knife, from <http://www.linux-support.com/cms/a-deep-look-into-netcat-the-tcpip-swiss-army-knife/>.
767. Trojans Revealed: Hackers Center: Internet Security Archive ..., <http://www.hackerscenter.com/archive/view.asp?id=24717>.
768. Dancho Danchev, The Complete Windows Trojans Paper, from http://www.frame4.com/content/pubs/comp_trojans.txt.
769. The corporate threat posed by email Trojans, from <http://www.gfisoftware.de/whitepapers/network-protection-against-trojans.pdf>.
770. Trojan Horses, from <http://www-i4.informatik.rwth-aachen.de/lufg/teaching/ss2004/dependability-seminar/paper/final8.pdf>.
771. Trojans - and how to protect your network against them, from http://www.windowsecurity.com/whitepapers/trojans_protect_your_network.html.
772. Fausi Qattan & Fredrik Thernelius, (2004), Master's Thesis, from <http://www.dsv.su.se/research/seclab/pages/pdf-files/04-34.pdf>.
773. Malicious Intrusion Techniques, <http://www.telecomworx.com/Adobe/Files39087.pdf>.

774. Increased use of Trojan Horse Programs, from <http://www.niscc.gov.uk/niscc/docs/tn-20040216-00080.html?lang=en>.
775. Anti Trojan source - How to protect your network against trojans ..., from <http://news.myinstall.com/news/45/>.
776. Dancho Danchev, Trojan White Paper, from <http://www.anti-trojan-software-reviews.com/trojan-white-paper-p2.htm>.
777. Trojans, from <http://www.emailprivacy.info/trojans>.
778. Remote Access Trojan FAQ and Port List Computer Security - Network ..., from <http://www.infosyssec.com/infosyssec/trojanportlist.html>.
779. WINSNORT.com: Intrusion Detection, from http://www.winsnort.com/modules.php?op=modload&name=FAQ&file=index&myfaq=yes&id_cat=13.
780. Trojan Horse Computer Infection Symptoms, from <http://hacker-eliminator.com/trojansymptoms.html>.
781. LockDown Millennium Advanced Online Help, from <http://lockdowncorp.com/manual/TrojanInfectionSymptoms.htm>.
782. Commodoon Communications - Threats to your Security on the Internet, from <http://www.commodoon.com/threat/threat-detect.htm>.
783. Van Hauser/THC, Placing Backdoors Through Firewalls, from http://www.cgisecurity.com/lib/placing_backdoors_through_firewalls.txt.
784. Mikejc, (2004), Tech-Recipes.com - Use System File Checker to Solve Problems, from http://www.tech-recipes.com/windows_tips602.html.
785. Exploring the Explodable, from <http://www.guninski.com/browsers.html>.
786. David Wells, (1996), Wrappers, from <http://www.objs.com/survey/wrap.htm>.
787. Milly, Steve A., Stan, Ojatex, Gordon, Darius and Buzz, (2000), WordPad, from www.pc-help.org/security/scrap.htm.
788. Trojans FAQ, <http://www.windowsecurity.com/faqs/Trojans/>.
789. Information on Computer Viruses, from <http://www-rohan.sdsu.edu/viruses.html>.
790. Advanced Network Configuration and Troubleshooting, from <http://snow.nl/dist/xhtmlc/ch05s02.html>.
791. Tom Armstrong, (2001), Netcat - The TCP/IP Swiss Army Knife, from <http://m.nu/program/util/netcat/netcat.html>.
792. Microsoft - Windows File, from Protection, from http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/system_file_protection.mspx?mf=true.
793. Scott W. Hotaling's, Placing Backdoors Through Firewalls, from <http://masc2279.no-ip.org/gadgets-toys/internet/placing-backdoors-through-firewalls>.
794. How to block ICMP tunneling?, from <https://listserv.icsalabs.com/pipermail/firewall-wizards/1999-July/006060.html>.
795. Newbie: Security, from <http://www.unixgeeks.org/security/newbie/security/firewall.html>.
796. Phrack Magazine Oo. Volume Seven, Issue Forty-Nine File 06 of ..., from <http://www.phrack.org/phrack/49/P49-06>.
797. Dancho Danchev dancho, The Complete Windows Trojans Paper, from <http://www.astalavista.com/index.php?section=directory&linkid=640>.
798. Declan McCullagh, (2006), Case 2:00-cr-00170-WHA-VPM, from <http://www.politechbot.com/docs/feds.trojan.hacking.brief.082406.pdf>.
799. TCPView, from <http://technet.microsoft.com/en-us/sysinternals/bb897437.aspx>.

800. CurrPorts, from <http://www.nirsoft.net/utils/cports.html>.
801. Process Monitor, from <http://technet.microsoft.com/en-in/sysinternals/bb896645.aspx>.
802. What's Running, from <http://www.whatsrunning.net/>.
803. PrcView, from <http://www.teamcti.com>.
804. Winsonar, from <http://www.fewbyte.com/winsonar.html>.
805. HiddenFinder, from <http://www.wenpoint.com/download/>.
806. Autoruns for Windows, from <http://technet.microsoft.com/en-us/sysinternals/bb963902.aspx>.
807. KillProcess, from http://orangelampsoftware.com/products_killprocess.php.
808. Security Task Manager, from <http://www.neuber.com/taskmanager/>.
809. Yet Another (remote) Process Monitor, from <http://yaprocmn.sourceforge.net/>.
810. MONIT, from <http://mmonit.com/monit/>.
811. OpManager, from <http://www.manageengine.com/network-monitoring/process-monitoring.html>.
812. jv16 Power Tools 2012, from <http://www.macecraft.com/jv16-powertools-2012/>.
813. PC Tools Registry Mechanic, from <http://www.pctools.com/registry-mechanic/>.
814. Reg Organizer, from <http://www.chemtable.com/organizer.htm>.
815. Registry Shower, from <http://www.registryshower.com/download.htm>.
816. Comodo Cloud Scanner, from <http://www.comodo.com/home/internet-security/cloud-scanner.php>.
817. Buster Sandbox Analyzer, from <http://bsa.isoftware.nl/>.
818. All-Seeing Eyes, from <http://www.fortego.com/en/ase.html>.
819. MJ Registry Watcher, from <http://www.jacobsm.com/mjsoft.htm#rgwtchr>.
820. Active Registry Monitor, from <http://www.devicelock.com/arm/>.
821. SpyMe Tools, from http://www.lcibrossolutions.com/spyme_tools.htm.
822. Regshot, from <http://regshot.sourceforge.net/>.
823. Registry Live Watch, from <http://leelusoft.blogspot.in/2009/11/registry-live-watch-10.html>.
824. DriverView, from <http://www.nirsoft.net/utils/driverview.html>.
825. Driver Detective, from <http://www.drivershq.com/>.
826. Unknown Device Identifier, from <http://www.zhangduo.com/udi.html>.
827. DriverGuide Toolkit, from <http://www.driverguidetoolkit.com/>.
828. DriverMax, from <http://www.innovative-sol.com/drivermax/index.htm>.
829. Driver Magician, from <http://www.drivermagician.com/>.
830. Driver Reviver, from <http://www.reviversoft.com/driver-reviver/>.
831. DriverScanner, from <http://www.uniblue.com/software/driverscanner/>.
832. Double Driver, from <http://www.boozet.org/dd.htm>.
833. My Drivers, from <http://www.zhangduo.com/driverbackup.html>.
834. DriverEasy, from <http://www.drivereeasy.com/>.
835. Windows Service Manager (SrvMan), from <http://tools.sysprogs.org/srvman/>.
836. SMART Utility, from <http://www.thewindowsclub.com/smart-a-utility-for-tweaking-windows-7-vista-xp-services>.
837. Netwrix Service Monitor, from http://www.netwrix.com/windows_services_monitoring_freeware.html.
838. Vista Services Optimizer, from <http://www.smartpcutilities.com/servicesoptimizer.html>.

839. ServiWin, from <http://www.nirsoft.net/utils/serviwin.html>.
840. Windows Service Manager Tray, from <http://winservicemanager.codeplex.com/>.
841. AnVir Task Manager, from <http://www.anvir.com/download.htm>.
842. Process Hacker, from <http://processhacker.sourceforge.net/downloads.php>.
843. Free Windows Service Monitor Tool, from [http://www.manageengine.com/free-windows-service-monitor-free-windows-service-monitor-index.html](http://www.manageengine.com/free-windows-service-monitor/free-windows-service-monitor-index.html).
844. Overseer Network Monitor, from <http://www.overseer-network-monitor.com/Download.aspx>.
845. Total Network Monitor, from <http://www.softinventive.com/products/total-network-monitor/>.
846. Starter, from http://codestuff.tripod.com/products_starter.html.
847. Security AutoRun, from <http://tcpmonitor.altervista.org/startup-manager/>.
848. Absolute Startup manager, from <http://www.absolutestartup.com/>.
849. ActiveStartup, from <http://www.hexilesoft.com/activestartup.html>.
850. StartEd Lite, from <http://www.outertech.com/en/windows-startup>.
851. Startup Inspector, from <http://www.windowsstartup.com/startupinspector.php>.
852. Program Starter, from <http://www.ab-tools.com/de/software/programmstarter/>.
853. Disable Startup, from <http://www.disablestartup.com/>.
854. StartupMonitor, from <http://www.mlin.net/StartupMonitor.shtml>.
855. Chameleon Startup Manager, from <http://www.chameleon-managers.com/downloads.php>.
856. Startup Booster, from http://www.smartpctools.com/startup_booster/.
857. FCIV, from <http://www.microsoft.com/en-us/download/details.aspx?id=11533>.
858. Tripwire, from <http://www.tripwire.com/it-security-software/security-configuration-management/file-integrity-monitoring/>.
859. FastSum, from <http://www.fastsum.com/download.php>.
860. WinMD5, from <http://www.blisstonia.com/software/WinMD5/#download>.
861. Advanced CheckSum Verifier (ACSV), from <http://www.irnis.net/>.
862. Fsum Fronted, from <http://fsumfe.sourceforge.net/>.
863. Verisys, from <http://www.ionx.co.uk/products/verisys>.
864. AFICK (Another File Integrity Checker), from <http://afick.sourceforge.net/>.
865. File Integrity Monitoring, from http://www.ncircle.com/index.php?s=products_ccm_file-integrity-monitoring.
866. Attribute Manager, from <http://www.miklsoft.com/attrman/index.html>.
867. PA File Sight, from <http://www.poweradmin.com/file-sight/index3.aspx>.
868. CSP File Integrity Checker, from http://www.tandemsecurity.com/solution_14.php.
869. ExactFile, from <http://www.exactfile.com/downloads/>.
870. OSSEC, from http://www.ossec.net/?page_id=19.
871. Windows Defender, from <http://www.microsoft.com/en-in/download/details.aspx?id=17>.
872. McAfee AntiVirus Plus, from <http://home.mcafee.com/store/free-antivirus-trials>.
873. Norton AntiVirus, from http://us.norton.com/downloads-trial-norton-antivirus?nid=us_hho_topnav_download_detail_nav.
874. Trojan Horse Construction Kit, from http://www.pestpatrol.com/zks/pestinfo/t/trojan_horse_construction_kit.asp.

875. Progenic Mail Trojan Construction Kit - PMT, from http://www.simovits.com/trojans/tr_data/y2630.html.
876. Pandora's Box, from <http://greece.mrdonn.org/greekgods/pandora.html>.
877. TrojanHunter, from <http://www.trojanhunter.com/trojanhunter/>.
878. Emsisoft Anti-Malware, from <http://www.emsisoft.in/en/software/antimalware/>.
879. Anti-Trojan Shield (ATS), from <http://www.atshield.com/?r=download>.
880. Spyware Doctor, from http://www.pctools.com/spyware-doctor/download/?src=lp_sd.
881. Anti Malware BOClean, from <http://www.comodo.com/home/internet-security/anti-malware.php>.
882. Anti Hacker, from <http://www.hide-my-ip.com/antihacker.shtml>.
883. XoftSpySE, from <http://www.paretologic.com/xoftspy/se/newlp/xray/>.
884. SPYWAREfighter, from <http://www.spamfighter.com/SPYWAREfighter/>.
885. Anti Trojan Elite, from http://www.remove-trojan.com/index_ate.php.
886. SUPERAntiSpyware, from <http://www.superantispyware.com/index.html>.
887. Trojan Remover, from <http://www.simplysup.com/tremover/download.html>.
888. Twister Antivirus, from <http://www.filseclab.com/en-us/>.

Module 07: Viruses and Worms

889. Types of Virus, from http://www.mindpride.net/root/Extras/Viruses/virus_protection_and_removal_ii.htm.
890. Vulnerabilities in Network Infrastructures and Prevention/Containment Measures, from <http://proceedings.informingscience.org/InSITE2012/InSITE12p053-067Awodele0012.pdf>.
891. Terminology, from http://www.f-secure.com/en/web/labs_global/terminology-f.
892. Virus Protection, from http://www.mindpride.net/root/Extras/Viruses/virus_protection_and_removal_iii.htm.
893. Paul Boutin, (2003), An inside view of the worm that crashed the Internet in 15 minutes, founder from <http://www.wired.com/wired/archive/11.07/slammer.html>.
894. Case Study: Microsoft Network Hacked by QAZ Trojan, from <http://www.msnbc.com/msn/482011.asp> Oct. 29, 2000.
895. Mark Russinovich, (2008), TCPView for Windows v2.53, from <http://www.sysinternals.com/Utilities/TcpView.html>.
896. Mark Russinovich and Bryce Cogswell, (2008), Autoruns for windows (v 9.32), from <http://www.sysinternals.com/Utilities/Autoruns.html>.
897. Merijn, (2005), Hijack This (System Checker) (v 1.99.1), from <http://www.majorgeeks.com/download.php?det=3155>.
898. Norman Book on Computer Viruses, from <http://download.norman.no/manuals/eng/BOOKON.PDF>.
899. Carey Nachenberg, Understanding and Managing Polymorphic Viruses from <http://www.symantec.com/avcenter/reference/striker.pdf>.
900. The Spread of the Sapphire/Slammer Worm, from <http://www.caida.org/publications/papers/2003/sapphire/sapphire.html>.
901. Mike Gunderloy, (2003), Microsoft Certified Professional Magazine Online | Newsletters, vol 2 #8, from <http://mcpmag.com/newsletter/article.asp?EditorialsID=153>.
902. R. A. Hettinga, (2003), Random Scanning Worms and Sapphire/Slammer's PRNG, from <http://www.mail-archive.com/cryptography@wasabisystems.com/msg03503.html>.

903. Information on a virus on campus, <http://security.uwo.ca/antivirus/infoHistory.html>.
904. Virus History - The Senior Most Virus!!, from www.optusnet.com.au/learning/email/virus.
905. Computer Knowledge Virus Tutorial, from www.mpl.org,eg/doc/eBOOKs/vtutor.pdf.
906. Dr. Alan Solomon and Robert M. Slade, 1990 - VX BBS & Little Black Book (AT&T Attack), 1991 - Tequila, 2001 - Gnuman, Winux Windows/Linux Virus, 2004 - Trojan.Xombe, Randex, Bizex, Witty, from www.cknow.com/vtutor/HistoryofViruses.html.
907. Michelangelo, DAME, & VCL, from <http://library.thinkquest.org/04oct/00460/malwareHistory.html>.
908. Honeypots, Honeynets, and Intrusion Detection, from <http://www.honeypots.net/>.
909. Featured Files, from <http://packetstormsecurity.org/>.
910. BinText, from <http://www.mcafee.com/apps/free-tools/termsofuse.aspx?url=/us/downloads/free-tools/bintext.aspx>.
911. UPX, from <http://upx.sourceforge.net/#downloadupx>.
912. Process Explorer, from <http://technet.microsoft.com/en-in/sysinternals/bb896653.aspx>.
913. RegShot, from <http://regshot.sourceforge.net/>.
914. OllyDbg, from <http://www.ollydbg.de/>.
915. ProcDump, from <http://technet.microsoft.com/en-us/sysinternals/dd996900.aspx>.
916. IDA Pro, from https://www.hex-rays.com/products/ida/support/download_demo.shtml.
917. VirusTotal, from <https://www.virustotal.com/en/>.
918. Anubis: Analyzing Unknown Binaries, from <http://anubis.iseclab.org>.
919. Avast! Online Scanner, from <http://onlinescan.avast.com>.
920. Malware Protection Center, from <http://www.microsoft.com/security/portal/>.
921. ThreatExpert, from <http://www.threatexpert.com>.
922. Dr. Web Online Scanners, from <http://vms.drweb.com>.
923. Metascan Online, from <http://www.metascan-online.com/>.
924. Bitdefender QuickScan, from <http://www.bitdefender.com/scanner/online/free.html>.
925. GFI SandBox, from <http://www.gfi.com/malware-analysis-tool>.
926. UploadMalware.com, from UploadMalware.com.
927. Fortinet, from http://www.fortiguard.com/antivirus/virus_scanner.html.
928. Immunet, from <http://www.immunet.com/free/index.html>.
929. AVG Antivirus, from <http://free.avg.com/in-en/homepage>.
930. BitDefender, from <http://www.bitdefender.com/Downloads/>.
931. Kaspersky Anti-Virus, from <http://www.kaspersky.com/trials>.
932. Trend Micro Internet Security Pro, from <http://apac.trendmicro.com>.
933. Norton AntiVirus, from http://us.norton.com/downloads-trial-norton-antivirus?inid=us_hho_topnav_download_detail_nav.
934. F-Secure Anti-Virus, from http://www.f-secure.com/en/web/home_global/anti-virus.
935. Avast Pro Antivirus, from <http://www.avast.com/pro-antivirus>.
936. McAfee AntiVirus Plus 2013, from <http://home.mcafee.com/store/free-antivirus-trials>.
937. ESET Smart Security 6, from <http://www.eset.com/download/home/detail/family/5/>.
938. Total Defense Internet Security Suite, from <http://www.totaldefense.com/shop/total-defense-internet-security-suite.aspx>.

939. What's Running, from <http://www.whatsrunning.net/>.
940. Winsonar, from <http://www.fewbyte.com/winsonar.html>.
941. Reg Organizer, from <http://www.chemtable.com/organizer.htm>.
942. Windows Service Manager (SrvMan), from <http://tools.sysprogs.org/srvman/>.
943. ServiWin, from <http://www.nirsoft.net/utils/serviwin.html>.
944. Starter, from http://codestuff.tripod.com/products_starter.html.
945. Security AutoRun, from <http://tcpmonitor.altervista.org/startup-manager/>.
946. FCIV, from <http://www.microsoft.com/en-us/download/details.aspx?id=11533>.

Module 08: Sniffing

947. What is Sniffer and how to detect sniffing in computer network, from <http://www.aboutonlinetips.com/sniffer-types-and-protecting-against-sniffing/>.
948. Anatomy of an ARP Poisoning Attack, from <http://www.unitedsystemsok.com/anatomy-of-an-arp-poisoning-attack>.
949. What is ARP?, from http://www.antiarp.com/english_94.html.
950. Modeling and Analysis of Wireless LAN Traffic, from http://www.dmcslab.hanyang.ac.kr/files/publication/journals/international/200911_08.pdf.
951. Dynamic ARP Inspection (DAI), from <http://daxm.net/ccienotes/20100131/dynamic-arp-inspection-dai>.
952. Overview of Layer 2 Switched Networks and Communication, from <http://www.sakunsharma.in/2011/07/overview-layer-2-switched-networks-communication/>.
953. Application Protocol IPv6, from http://www.ciscoexpo.ru/club/sites/default/files/seminar_attachments/ipv6.pdf.
954. Dynamic Host Configuration Protocol, from <http://www.ietf.org/rfc/rfc2131.txt>.
955. Understanding, Preventing, Defending Against Layer 2 Attacks, from <http://www.sanog.org/resources/sanog15/sanog15-yusuf-l2-security.pdf>.
956. A New Scheme to Check ARP Spoofing: Prevention of MAN-IN-THE-MIDDLE Attack, from <http://www.ijcsit.com/docs/Volume%202/vol2issue4/ijcsit2011020420.pdf>.
957. LAYER 2 ATTACKS & MITIGATION TECHNIQUES, from <http://www.sanog.org/resources/sanog7/yusuf-L2-attack-mitigation.pdf>.
958. Chris Martin, What is Sniffer and how to detect Sniffing in computer network, Available from <http://74.125.153.132/search?q=cache:Tu6yfsiaY3AJ:www.aboutonlinetips.com/sniffer-types-and-protecting-against-sniffing/+wire+sniffing+techniques&cd=25&hl=en&ct=clnk&gl=in&client=firefox-a>.
959. Adam Barth, Secure content sniffing for Web browsers or How to stop papers from reviewing themselves, Available from <http://www.adambarth.com/papers/2009/barth-caballero-song.pdf>.
960. Undetectable sniffing on Ethernet, Available from <http://www.askapache.com/security/sniffing-on-ethernet-undetected.html>.
961. Suhas A Desai, (2007), Techniques for Preventing Sniffing, Packet Sniffing: Sniffing Tools Detection Prevention Methods, Available from <http://e-articles.info/e/a/title/Packet-Sniffing:-Sniffing-Tools-Detection-Prevention-Methods/>.
962. Suhas A Desai, (2007), Tool to Detect Sniffers, Packet Sniffing: Sniffing Tools Detection Prevention Methods, Available from <http://e-articles.info/e/a/title/Packet-Sniffing:-Sniffing-Tools-Detection-Prevention-Methods/>.
963. Identifying Nonessential Services and Attacks > Attacks, from <http://www.informit.com/articles/article.asp?p=98121&seqNum=2>.

964. ARP cache poisoning /ARP spoofing, from <http://su2.info/doc/arpspoof.php>.
965. Network management, network discovery, SNMP, MIB and WMI browsers, from www.networkview.com/html/features.html.
966. Address Resolution Protocol (ARP), from www.erg.abdn.ac.uk/users/gorry/course/inet-pages/arp.html.
967. Angela D. Orebaugh, (2004), Top Ten Ethereal Tips and Tricks, from <http://www.onlamp.com/pub/a/security/2004/05/13/etherealtips.html>.
968. Packages, from <http://packages.debian.org/>.
969. Network Protocol Analysis, from <http://www.maatec.com/>.
970. The Hacker's Ethic, from <http://web.textfiles.com/ezines/HWA/hwa-hn34.txt>.
971. Jaromil, Dyne:II GNU/Linux User's Guide, from <http://dynebolic.org/dynebolic-man.pdf>.
972. Address Resolution Protocol (arp), from www.erg.abdn.ac.uk/users/gorry/course/inet-pages/arp.html.
973. Adam Barth, Juan Caballero and Dawn Song, Secure Content Sniffing for Web Browsers, or How to Stop Papers from Reviewing Themselves, <http://www.adambarth.com/papers/2009/barth-caballero-song.pdf>.
974. Alberto Ornaghi and Marco Valleri, Man in the middle attacks, <http://www.blackhat.com/presentations/bh-europe-03/bh-europe-03-valleri.pdf>.
975. Tom Olzak, (2006), DNS Cache Poisoning: Definition and Prevention, http://adventuresinsecurity.com/Papers/DNS_Cache_Poisoning.pdf.
976. Sean Whalen, (2001), An Introduction to Arp Spoofing, http://www.rootsecure.net/content/downloads/pdf/arp_spoofing_intro.pdf.
977. Daiji Sanai, (2001), Detection of Promiscuous Nodes using ARP packets, http://www.securityfriday.com/promiscuous_detection_01.pdf.
978. Network management, network discovery, SNMP, MIB and WMI browsers, from www.networkview.com/html/what_s_new.html.
979. Source Address Spoofing, from <http://www.networkcomputing.com/shared/article/showArticle.jhtml?articleId=8702815&classroom>.
980. Keith Brown, (1999), Security Briefs, from <http://www.microsoft.com/msj/0299/security/security0299.aspx>.
981. Corey Nachreiner, (2005), Anatomy of an ARP Poisoning Attack, from <http://www.watchguard.com/infocenter/editorial/135324.asp>.
982. macof, from <http://www.monkey.org>.
983. Yersinia, from <http://www.yersinia.net/download.htm>.
984. Dhcpstarv, from <http://dhcpstarv.sourceforge.net/>.
985. Gobbler, from <http://gobbler.sourceforge.net/>.
986. Cain & Abel, from <http://www.oxid.it/cain.html>.
987. WinArpAttacker, from <http://www.xfocus.org/index.html>.
988. Ufasoft Snif, from <http://ufasoft.com/sniffer/>.
989. XArp, from <http://www.chrismc.de/development/xarp/index.html>.
990. SMAC, from <http://www.klcconsulting.net/smac/index.html#download>.
991. Cascade Pilot, from <http://www.riverbed.com/products-solutions/products/performance-management/network-infrastructure/High-Speed-Packet-Analysis.html>.
992. Tcpdump, from <http://www.tcpdump.org/>.
993. WinDump, from <http://www.winpcap.org/windump/default.htm>.

994. Capsa Network Analyzer, from http://www.colasoft.com/download/products/capsa_free.php.
995. OmniPeek Network Analyzer, from http://www.wildpackets.com/products/omnipeek_network_analyzer.
996. Observer, from <http://www.networkinstruments.com/products/observer/index.php?tab=download>.
997. Sniff-O-Matic, from <http://www.kwakkelflap.com/sniffer.html>.
998. JitBit Network Sniffer, from <http://www.jitbit.com/networksniffer/>.
999. MSN Sniffer 2, from <http://www.msnsniffer.com/download/index.htm>.
1000. Ace Password Sniffer, from <http://www.effetech.com/aps/>.
1001. RSA NetWitness Investigator, from <http://www.emc.com/security/rsa-netwitness.htm#!freeware>.
1002. Big-Mother, from <http://www.tupsoft.com/download.htm>.
1003. EtherDetect Packet Sniffer, from <http://www.etherdetect.com/download.htm>.
1004. dsniff, from <http://monkey.org/~dugsong/dsniff/>.
1005. EffeTech HTTP Sniffer, from <http://www.effetech.com/download/>.
1006. Ntop, from <http://www.ntop.org/products/ntop/>.
1007. Ettercap, from <http://ettercap.sourceforge.net/downloads.html>.
1008. SmartSniff, from <http://www.nirsoft.net/utils/smsniff.html>.
1009. EtherApe, from <http://etherape.sourceforge.net/>.
1010. Network Probe, from <http://www.objectplanet.com/probe/>.
1011. Snort, from <http://www.snort.org/>.
1012. Sniff'em, from <http://www.sniff-em.com/download.shtml>.
1013. MaaTec Network Analyzer, from <http://www.maatec.com/mtna/download.html>.
1014. Alchemy Network Monitor, from http://www.mishelpers.com/network_monitor/index.html.
1015. CommView, from <http://www.tamos.com/download/main/index.php>.
1016. NetResident, from <http://www.tamos.com/products/netresident/>.
1017. AIM Sniffer, from <http://www.effetech.com/aim-sniffer/index.htm>.
1018. Netstumbler, from <http://www.netstumbler.com/downloads/>.
1019. IE HTTP Analyzer, from <http://www.ieinspector.com/http analyzer/>.
1020. MiniStumbler, from <http://www.netstumbler.com/downloads>.
1021. PacketMon, from <http://www.analogx.com/contents/download/Network/pmon/Freeware.htm>.
1022. NADetector, from http://www.nsauditior.com/network_monitoring/nadetector_traffic_analyzer.html.
1023. Microsoft Network Monitor, from <http://www.microsoft.com/en-us/download/details.aspx?id=4865>.
1024. NetworkMiner, from <http://www.netresec.com/?page=NetworkMiner>.
1025. Network Security Toolkit, from <http://www.networksecuritytoolkit.org/nst/index.html>.
1026. Ethereal, from <http://www.ethereal.com/>.
1027. KSniffer, from <http://ksniffer.sourceforge.net/index.php?section=download>.
1028. IPgrab, from <http://ipgrab.sourceforge.net/>.
1029. WebSiteSniffer, from http://www.nirsoft.net/utils/web_site_sniffer.html.
1030. ICQ Sniffer, from <http://www.etherboss.com/icq/download.htm>.
1031. URL Helper, from <http://www.urlhelper.com/index.htm>.
1032. WebCookiesSniffer, from http://www.nirsoft.net/utils/web_cookies_sniffer.html.

1033. York, from <http://thesz.diecru.eu/content/york.php>.
1034. IP Traffic Spy, from http://www.networkdls.com/Software/View/IP_Traffic_Spy/.
1035. SniffPass, from http://www.nirsoft.net/utils/password_sniffer.html.
1036. Cocoa Packet Analyzer, from <http://www.tastycoobytes.com/cpa/>.
1037. vxSniffer, from <http://www.cambridgevx.com/vxsniffer.html>.
1038. PromqryUI, from <http://www.microsoft.com/en-us/download/details.aspx?id=16883>.

Module 09: Social Engineering

1039. The use of Detailed Explanation of the the the working principle of of the port scanning tool and the the NMAP, from <http://www.boxueshe.org/read.php?tid=36>
1040. Sarah Granger, (2002), Social Engineering Fundamentals, Available from www.securityfocus.com/infocus/1533.
1041. Mika Tolvanen, (2006), F-Secure Trojan Information Pages, Available from http://www.f-secure.com/v-descs/redbrowser_a.shtml.
1042. Dancho Danchev, (2009), Social Engineering by a fake SMS spying tool, Available from <http://blogs.zdnet.com/security/?p=3162>.
1043. Growth on Use of Social Networking Sites, Available from http://www.pewinternet.org/~/media/Files/Reports/2009/PIP_Adult_social_networking_data_memo_FINAL.pdf.pdf.
1044. LinkedIn, Available from <http://www.linkedin.com/>.
1045. Micha Pekrul, (2009), Rogue LinkedIn Profiles Lead To Malware, Available from <http://www.avertlabs.com/research/blog/index.php/2009/01/06/rogue-linkedin-profiles-lead-to-malware/>.
1046. Bogdan Dumitru,(2009), Risks of Social Networking and the Corporate Network, Available from <http://www.itbusinessedge.com/cm/community/features/guestopinions/blog/the-risks-of-social-networking-and-the-corporate-network/?cs=33877>.
1047. Terry Turner, Social Engineering – Can Organizations Win the Battle?, from http://www.infosecwriters.com/text_resources/pdf/Social_Engineering_Can_Organizations_Win.pdf.
1048. Bruce Schneier, (2005), Schneier on Security: Weakest Link Security, from http://www.schneier.com/blog/archives/2005/12/weakest_link_se.html.
1049. Sharon Gaudin, Social Engineering: The Human Side Of Hacking, from <http://www.crime-research.org/library/Sharon2.htm>.
1050. Social Engineering Hackers-LAN Times 11/6/95, from http://www.security-protocols.com/textfiles/social-engineering/soc_eng2.html.
1051. Psychology of Social Engineering, from <http://cybercrimes.net/Property/Hacking/Social%20Engineering/PsychSocEng/PsySocEng.html>.
1052. Michael L. Snider, Articles, from <http://staff.rio.edu/msnider/?cat=7>.
1053. Wylie Wong, (2000), Oracle chief defends Microsoft snooping | CNET News.com, from http://news.com.com/Oracle+chief+defends+Microsoft+snooping/2100-1001_3-242560.html.
1054. Engineering Hackers-LAN, from http://www.security-protocols.com/textfiles/social-engineering/soc_eng2.html.
1055. Examples of Phishing Emails, from http://www.banksafeonline.org.uk/phishing_examples.html.
1056. Anti-Phishing Resources, from <http://www.antiphishing.org/resources.html>.

1057. Netcraft Toolbar, from <http://toolbar.netcraft.com/install>.
1058. PhishTank, from <http://www.phishtank.com/>.
1059. ReadNotify, from <http://www.readnotify.com/>.
1060. Social Engineering Toolkit (SET), from <https://www.trustedsec.com/downloads/social-engineer-toolkit/>.

Module 10: Denial-of-Service

1061. Distributed Denial of Service: Taxonomies of Attacks, Tools and Countermeasures, from <http://palms.ee.princeton.edu/PALMSopen/DDoS%20Final%20PDCS%20Paper.pdf>.
1062. Denial of Service Attack Detection Techniques, from [https://www.evernote.com/shard/s9/note/b11a8c31-8651-4d74-acf9-1fb1b3c0f090](https://www.evernote.com/shard/s9/note/b11a8c31-8651-4d74-acf9-1fb1b3c0f090/wishi/crazylazy#st=p&n=b11a8c31-8651-4d74-acf9-1fb1b3c0f090).
1063. Welcome to the new IP reality, from http://lukasz.bromirski.net/docs/prezos/confidence2008/new_ip_reality_bp.pdf.
1064. What Happened to Blue Security, from <http://slashdot.org/story/06/05/08/142229/what-happened-to-blue-security>.
1065. Remotely Triggered Black Hole Filtering in IP Version 6 for Cisco IOS, Cisco IOS XE, and Cisco IOS XR Software, from http://www.cisco.com/web/about/security/intelligence/ipv6_rtbh.html.
1066. Frank Kargl, Jörn Maier, Stefan Schlott, and Michael Weber , Protecting Web Servers from Distributed Denial of Service Attacks, from <http://www10.org/cdrom/papers/409/>.
1067. Denial of Service Attacks, from http://www.cert.org/tech_tips/denial_of_service.html.
1068. Craig A. Huegen, (2000), Smurf Attack Information, from <http://www.pentics.net/denial-of-service/white-papers/smurf.cgi>.
1069. Denial of service, from http://searchappsecurity.techtarget.com/sDefinition/0,290660,sid92_gci213591,00.html.
1070. Solucom, VPN (Virtual Private Network) and Internet Firewall ..., from <http://www.solucom.com/define.htm>.
1071. Vladimir Golubev, (2005), DoS attacks: crime without penalty, <http://www.crime-research.org/articles/1049/>.
1072. Gunter Ollmann, (2009), The Botnet vs. Malware Relationship, http://www.damballa.com/downloads/d_pubs/WP%20Many-to-many%20Botnet%20Relationships%20%282009-05-21%29.pdf.
1073. Gunter Ollmann, (2009), Botnet Communication Topologies, http://www.damballa.com/downloads/r_pubs/WP%20Botnet%20Communications%20Primer%20%282009-06-04%29.pdf.
1074. Kasey Efaw, Installing Snort 2.8.5.2 on Windows 7, http://www.snort.org/assets/135/Installing_Snort_2.8.5.2_on_Windows_7.pdf.
1075. Renaud BIDOU, Fighting the Botnet Ecosystem, <http://www.iv2-technologies.com/FightingBotnetEcosystem.pdf>.
1076. Ping of death, from http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci822096,00.html.
1077. Apostates of Islam :: View topic - FFI down again, from <http://www.apostatesofislam.com/forum/viewtopic.php?t=189&postdays=0&postorder=asc&start=225&sid=0e55c35186bbe87c48bdfe6f62e0e4a5>.
1078. Jason Anderson, An Analysis of Fragmentation Attacks, from <http://www-src.lip6.fr/homepages/Fabrice.Legond-Aubry/www.ouah.org/fragma.html>.

1079. [PDF] IEEE P1615™/D2 Draft Recommended Practice for Network ...,
http://grouper.ieee.org/groups/sub/wgc3/C3TF1%20Documents/drafts/P1615_draft2.pdf.
1080. [DOC] Abstract, from http://www.bridgeport.edu/sed/projects/cs597/Spring_2004/juilan/Jui-Lan_Network%20Security%20-%20Analysis%20of%20Attack%20and%20Defense%20Strategies.doc.
1081. Mariusz Burdach, (2003), Hardening the TCP/IP stack to SYN attacks, from
<http://www.securityfocus.com/infocus/1729>.
1082. Citations: TCP SYN Flooding and IP Spoofing Attacks (ResearchIndex), from
<http://citeseer.ist.psu.edu/context/141856/0>.
1083. Lasse Huovinen and Jani Hursti, from Denial of Service Attacks: Teardrop and Land,
<http://users.tkk.fi/~lhuovine/study/hacker98/dos.html>.
1084. Underground security systems research, from <http://www.ussrback.com/Win/>.
1085. Stephen Specht & Ruby Lee, (2003), from Taxonomies of Distributed Denial of Service Networks, Attacks ..., from
http://www.princeton.edu/~rblee/ELE572Papers/Fall04Readings/DDoS Survey Paper_20030516_Final.pdf
1086. David Dittrich, (1999), The DoS Project's "trinoo" distributed denial of service attack tool from
<http://www.donkboy.com/html/stuff.htm>.
1087. Anti Online's Fight- Back! Computer Security..., from http://www.antionline.com/fight-back/What_Are_DDOS_Attacks.php.
1088. Sven Dietrich, Analysis of the Shaft distributed Denial of Service tool, from
<http://www.securiteam.com/securitynews/5AP0F000IM.html>.
1089. Analyzing Distributed Denial Of Service Tools: The Shaft Case, from
<http://www.ece.cmu.edu/~adrian/630-f03/readings/shaft.pdf>.
1090. Distributed Denial of Service Tools, from <http://www.fz-juelich.de/jsc/net/security/infos/DDoS/IN-99-07.html>.
1091. David Moore Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, and Nicholas, (2003), Weaver Inside the Slammer Worm, from
<http://csdl2.computer.org/persagen/DLAbsToc.jsp?resourcePath=/dl/mags/sp/&toc=comp/mags/sp/2003/04/j4toc.xml&DOI=10.1109/MSECP.2003.1219056>.
1092. RudhraKumar Venkatesan and ShashidharLakkavalli, TCP/IP Vulnerabilities, from
<http://islab.oregonstate.edu/koc/ece478/00Report/LV.pdf>.
1093. Dave Dittrich, Bugtraq: Analysis of trin00, from <http://seclists.org/lists/bugtraq/1999/Dec/0093.html>
1094. Fravia denial of service attack tools, from www.searchlores.org/dod1.htm.
1095. David Dittrich, (1999), Trinoo Analysis, from <http://staff.washington.edu/dittrich/misc/trinoo.analysis>.
1096. John Michalski, Carrie Price, Eric Stanton, Erik Lee, CHUA, Kuan Seah, Wong, Yip Heng and TAN, and Chung Pheng, (2002), DYNAT TECHNOLOGIES ASSESSMENT REPORT, from
<http://www.sandia.gov/iorta/docs/SAND%202002-3613%20DYNAT.pdf>.
1097. <.....A.VERY..THING..IS..POSSIBLE..TO..ZEROGEEK.....>, from <http://mifwarz.blogspot.com/>.
1098. Jason Barlow and Woody Thrower, (2000), TFN2K - An Analysis Jason Barlow and Woody Thrower AXENT Security..., from http://packetstormsecurity.org/distributed/TFN2k_Analysis-1.3.txt.
1099. Jason Barlow and Woody Thrower, (2000), TFN2K - An Analysis (Revision : 1.3), from
http://www.symantec.com/avcenter/security/Content/2000_02_10_a.html.
1100. Gary C. Kessler, (2000), Distributed Denial-Of-Service, from
<http://www.garykessler.net/library/ddos.html>.

1101. David Dittrich, (1999), Stacheldraht Analysis, <http://staff.washington.edu/dittrich/misc/stacheldraht.analysis>.
1102. Bugtraq: Analysis of the Shaft distributed denial of service tool, from <http://seclists.org/lists/bugtraq/2000/Mar/0215.html>.
1103. Sven Dietrich, Neil Long, & David Dittrich, Analyzing Distributed Denial Of Service Tools: The Shaft Case, from http://www.usenix.org/event/lisa2000/full_papers/dietrich/dietrich_html/.
1104. IP: new DoS attack, from <http://www.interesting-people.org/archives/interesting-people/200009/msg00006.html>.
1105. Dave Farber, (2000), IP: new DoS attack, from http://www.princeton.edu/~rblee/DDoS%20Survey%20Paper_v7final.doc.
1106. David Dittrich, George Weaver, Sven Dietrich, and Neil Long, The mstream distributed denial of service attack tool, from <http://www.linuxsecurity.com/content/view/107513/2/>.
1107. The Distributed Reflection DoS Attack, from <http://www.grc.com/dos/drdoS.htm>.
1108. Steve Gibson, (2002), Distributed Reflection Denial of Service Bandwidth Consumption, from <http://cs-www.cs.yale.edu/homes/arvind/cs425/doc/drdoS.pdf>.
1109. SYN Attack, from www.ieee.org.
1110. Hang Chau, (2004), Network Security - Defense Against DoS/DDoS Attacks, from <http://www.securitydocs.com/library/2576>.
1111. Aaron Sullivan, 2001, An Audit of Active Directory Security, from <http://www.securityfocus.com/infocus/1293>.
1112. Xatrix Security, from <http://www.xatrix.org/download.php?id=28&r=1>.
1113. Denial of Service, from http://www.mycert.org.my/network_abuse/dos.html.
1114. Denial of Service Attack in NetBIOS Services, from <http://www.kb.cert.org/vuls/id/32650>.
1115. James Middleton, (2001), Cloaking system poses new security threat, from <http://www.iwr.co.uk/vnunet/news/2114991/cloaking-system-poses-security-threat>.
1116. NFR DDOS problems, from <http://www.shmoo.com/mail/ids/may01/msg00038.shtml>.
1117. Latest Windows Security Articles, from <http://www.windowsecurity.com/>.
1118. Gregg Keizer, (2006), Massive DoS Attacks Against ISPs On The Rise, from http://www.informationweek.com/story/showArticle.jhtml?articleID=192701817&cid=RSSfeed_IWKN.ws.
1119. Jason Barlow and Woody Thrower, AXENT Security, from http://packetstormsecurity.org/distributed/TFN2k_Analysis-1.3.txt.
1120. Fabrice LEGOND-AUBRY, An Analysis of Fragmentation Attacks, from <http://www-src.lip6.fr/homepages/Fabrice.Legond-Aubry>.
1121. Jui-Lan Lai, Network Security-- Analysis of Attack and Defense, from http://www.bridgeport.edu/sed/projects/cs597/Spring_2004/juilan/Jui-Lan_Network%20Security%20-%20Analysis%20of%20Attack%20and%20Defense%20Strategies.doc Strategies.
1122. Targa: [PDF] security, from <https://www.cis.strath.ac.uk/~gw/52507/security.pdf>.
1123. WORM_MYDOOM.B, Description and solution, from http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM_MYDOO.M.B.
1124. Information on a virus on campus, from <http://security.uwo.ca/antivirus/infoHistory.html>.
1125. [PDF] Microsoft PowerPoint - ISI_Malware.ppt, from http://www-t.zhwin.ch/it/isi/v/ISI_Malware.pdf.
1126. R. A. Hettinga, (2003), Random Scanning Worms and Sapphire/Slammer's PRNG..., from <http://www.mail-archive.com/cryptography@wasabisystems.com/msg03503.html>.

1127. Mike Gunderloy, (2003), Microsoft Certified Professional Magazine Online | Newsletters ..., from <http://mcpmag.com/newsletter/article.asp?EditorialsID=153>.
1128. The Spread of the Sapphire/Slammer Worm, from <http://www.caida.org/publications/papers/2003/sapphire/sapphire.html>.
1129. Norman Book on Computer Viruses, from <http://download.norman.no/manuals/eng/BOOKON.PDF>.
1130. IT Architect | Strategies & Issues: Honeypots - Sticking It to, from <http://www.itarchitect.com/article/NMG20030403S0005>.
1131. Roger A. Grimes, (2005), Honeypots for Windows, from <http://www.gtpcc.org/gtpcc/honeypotsforwindows.htm>.
1132. Honeypots [Infosecwriters.com], from <http://www.infosecwriters.com/texts.php?op=display&id=80>.
1133. J.A. Hamilton, Reflection of the Exploit, from http://www.eng.auburn.edu/users/hamilton/security/SE2/Directed_Reflection_DOS_Hamilton.pdf.
1134. Distributed Reflection Denial of Service Bandwidth Consumption ..., from <http://www.grc.com/files/drddos.pdf>.
1135. Kevin Houle & Chad Dougherty, (2000), CERT Incident Note IN-99-07: Distributed Denial of Service Tools, from http://www.cert.org/incident_notes/IN-2000-05.html.
1136. DDoS Resources, from <http://www.anml.iu.edu/ddos/tools.html>.
1137. Jason Barlow and Woody Thrower, (2000), AXENT : SWAT : TFN2K - An Analysis, from http://www.symantec.com/avcenter/security/Content/2000_02_10_a.html.
1138. David Dittrich, (1999), Trinoo Analysis, from <http://staff.washington.edu/dittrich/misc/trinoo.analysis>.
1139. CERT warns of networked denial of service attacks – Computerworld, from <http://www.computerworld.com/action/pages.do?command=viewPage&pagePath=/404>.
1140. Internet security, from http://www.fsa.ulaval.ca/personnel/vernag/EH/F/manif/lectures/internet_security.htm.
1141. Solucom VPN (Virtual Private Network) and Internet Firewall ..., from <http://www.solucom.com/define.htm>.
1142. Library Computer and Network Security: Library Security Principles ..., from http://www.infopeople.org/resources/security/basics/threats_vulnerabilities.html.
1143. Wireless DoS, from <http://www.cisco.com/en/US/docs/wireless/technology/wips/deployment/guide/wipsdep.html#wp150481>.
1144. Gary C. Kessler, (2000), "Defenses Against Distributed Denial of Service Attacks", from <http://www.garykessler.net/library/ddos.html>.
1145. Abhishek Singh, (2005), Demystifying Denial-Of-Service attacks, part one, from <http://www.symantec.com/connect/articles/demystifying-denial-service-attacks-part-one>.
1146. Denial-of-service attack, from http://en.wikipedia.org/wiki/Denial-of-service_attack#Incidents.
1147. Kevin Poulsen, (2010), New: Cyberattack Against WikiLeaks Was Weak, from <http://www.wired.com/threatlevel/2010/11/wikileaks-attack/>.
1148. PlugBot, from <http://theplugbot.com>.
1149. Illusion Bot and NetBot Attacker, from .
1150. DoS HTTP, from <http://socketsoft.net/products.asp?p=doshttp>.
1151. KFSensor, from <http://www.keyfocus.net/kfsensor/download/>.
1152. FortiDDoS-300A, from <http://www.fortinet.com/products/fortiddos/300A.html>.
1153. DDoS Protector, from <http://www.checkpoint.com/products/ddos-protector/>.

1154. Cisco Guard XT 5650, from http://www.cisco.com/en/US/prod/collateral/vpndevc/ps5879/ps6264/ps5888/product_data_sheet0900aecd800fa55e.html.
1155. Arbor Pravail: Availability Protection System, from <http://www.arbornetworks.com/products/pravail>.
1156. D-Guard Anti-DDoS Firewall, from <http://www.d-guard.com/>.
1157. NetFlow Analyzer, from <http://www.manageengine.com/products/netflow/download.html>.
1158. FortiDDoS, from <http://www.fortinet.com/products/fortiddos/>.
1159. SDL Regex Fuzzer, from <http://www.microsoft.com/en-us/download/confirmation.aspx?id=20095>.
1160. DefensePro, from http://www.radware.com/Products/ApplicationNetworkSecurity/DDoS_Attack_Protection.aspx.
1161. WANGuard Sensor, from <https://www.andrisoft.com/store/evaluation-request>.
1162. DOSarrest, from <http://www.dosarrest.com>.
1163. NetScaler Application Firewall, from http://www.citrix.com/products/netscaler-application-delivery-controller/try.html?ntref=header_try.
1164. Anti DDoS Guardian, from <http://www.beethink.com/antiddos.htm>.
1165. FortGuard DDoS Firewall, from <http://www.fortguard.com/ddosmonitor.html>.
1166. DDoSDefend, from <http://ddosdefend.com/ddos-protection.html>.
1167. Webserver Stress Tool, from <http://www.paessler.com/download/webstress>.
1168. Web Stress Tester, from <http://www.fastream.com/webstresstester.php>.
1169. JMeter, from http://jmeter.apache.org/download_jmeter.cgi.
1170. DoS HTTP, from <http://socketsoft.net/products.asp?p=doshttp>.
1171. Mail Bomber, from <http://www.getfreefile.com/bomber.html>.
1172. Advanced Mail Bomber, from <http://www.softheap.com/abomber.html>.

Module 11: Session Hijacking

1173. Steps in Session Hijacking, from <http://www.hackguide4u.com/2010/03/steps-in-session-hijacking.html>.
1174. Session Hijacking, from http://www.imperva.com/resources/glossary/session_hijacking.html.
1175. IP Hijack, from <http://dokfleed.net/duh/modules.php?name=News&file=article&sid=3>.
1176. Spoofing Vs Hijacking, from <http://www.hackguide4u.com/2010/03/spoofing-vs-hijacking.html>.
1177. Lee Lawson, (2005), Session Hijacking Packet Analysis, Available from <http://www.securitydocs.com/library/3479>.
1178. Dave Dittrich, Session hijack script, Available from <http://blinky-lights.org/script.html>.
1179. Session hijacking attack, Available from http://www.owasp.org/index.php/Session_hijacking_attack.
1180. Shray Kapoor, Session Hijacking Exploiting TCP, UDP and HTTP Sessions, http://www.infosecwriters.com/text_resources/pdf/SKapoor_SessionHijacking.pdf.
1181. David Endler, (2001), Brute-Force Exploitation of Web Application Session IDs, <http://www.cgisecurity.com/lib/SessionIDs.pdf>.
1182. Robert Auger, Credential and Session Prediction, Available from <http://projects.webappsec.org/Credential-and-Session-Prediction>.
1183. Trojan horse, Available from http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci213221,00.html.

1184. Jamie.riden, (2008), CLIENT-SIDE ATTACKS, Available from <http://www.honeynet.org/node/157>.
1185. Lee Lawson, (2005), Session Hijacking Packet Analysis, Available from <http://www.securitydocs.com/library/3479>.
1186. Addison Wesley, (2007), Fibre Channel and IP session hijacking assessment exercise, Available from http://searchstoragechannel.techtarget.com/generic/0,295582,sid98_gci1250226,00.html.
1187. Prevention from Session Hijacking, Available from <http://hydtechie.blogspot.com/2008/08/prevention-from-session-hijacking.html>.
1188. Session Hijacking, Available from <http://www.cs.binghamton.edu/~steflik/cs455/sessionhijacking.htm>.
1189. Hackerthreads.org security: View topic - Network Session Hijacking, from www.hackerthreads.org/phpbb/viewtopic.php?t=745.
1190. OpenSSH – SwiK swik.net/OpenSSH MOM 2005: IP Security (IPSec), from www.microsoft.com/technet/prodtechnol/mom/mom2005/Library/39cb2734-506c-4101-887c-c2d2146621c0.mspx.
1191. Microsoft Security Bulletin (MS99-046): Frequently Asked Questions, from www.microsoft.com/technet/security/bulletin/fq99-046.mspx.
1192. Laurent Joncheray, Simple Active Attack Against TCP Sequence Number Prediction, from <http://www.cert.org/advisories/CA-2001-09.html>.
1193. Term: S/key, from www.webopedia.com.
1194. Attacks against IIS, from <http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/0a199196-4ae9-41eb-b8c1-572251f9f550.mspx?mfr=true>.
1195. Joe Jenkins, (2000), Internet Security and Your Business - Knowing the Risks, from <http://www.securityfocus.com/infocus/1194>.
1196. Webapplication Attacks – Intro, from www.netprotect.ch/downloads/webguide.pdf.
1197. Alexia Tsotsis, (2010), How To Protect Your Login Information From Firesheep, from <http://techcrunch.com/2010/10/25/firesheep/>.
1198. zaproxy, from <https://code.google.com/p/zaproxy/downloads/list>.
1199. JHijack, from <http://sourceforge.net/projects/jhijack/files/latest/download?source=files>.
1200. Hamster, from <http://erratasec.blogspot.in/2009/03/hamster-20-and-ferret-20.html>.
1201. Ferret, from <http://erratasec.blogspot.in/2009/03/hamster-20-and-ferret-20.html>.
1202. Surf Jack, from <https://code.google.com/p/surfjack/downloads/detail?name=surfjack-0.2b.zip>.
1203. PerJack, from <http://packetstormsecurity.org>.
1204. Ettercap, from <http://sourceforge.net/projects/ettercap/files/latest/download?source=dll>.
1205. WhatsUp Gold Engineer's Toolkit, from http://www.whatsupgold.com/products/download/network_management.aspx?k_id=ping-sweep-tool.
1206. Hunt, from <http://packetstormsecurity.com/files/download/21968/hunt-1.5bin.tgz>.
1207. Juggernaut, from <http://www.securiteam.com>.
1208. TamperIE, from <http://www.bayden.com/TamperIE/>.
1209. Cookie Cadger, from https://www.cookiecadger.com/?page_id=19.

Module 12: Hacking Webservers

1210. Web Parameter Tampering, from https://www.owasp.org/index.php/Web_Parameter_Tampering.

1211. Web Server Security and Database Server Security, from <http://www.itura.net/training/19-web-server-security-and-database-server-security.pdf>.
1212. Configuring and organizing server security information, from <http://www.freepatentsonline.com/7712137.html>.
1213. Internet Security, from <http://wiki.winf.at/184216>.
1214. Securing applications, from <http://www.slideshare.net/florinc/application-security-1831714>.
1215. About Securing Applications, from http://docs.oracle.com/cd/E16348_01/books/SecurHarden/SecurHarden_AppSecur2.html.
1216. Insecure Configuration Management, from http://www.upenn.edu/computing/security/swat/SWAT_Top_Ten_A10.php.
1217. Server Misconfiguration, from <http://projects.webappsec.org/w/page/13246959/Server%20Misconfiguration>.
1218. Repairing system after following directions in attempt to clean virus, from <http://forum.hijackthis.de/archiv/18982-repairing-system-after-following-directions-attempt-clean-virus.html>.
1219. Header Manipulation, from http://www.hptenterprisesecurity.com/vulncat/en/vulncat/sql/header_manipulation.html.
1220. Cache Poisoning, from https://www.owasp.org/index.php/Cache_Poisoning.
1221. Improving Web Application Security: Threats and Countermeasures, from <http://msdn.microsoft.com/en-us/library/aa302418.aspx>.
1222. Best Practices for Applying Service Packs, Hotfixes and Security Patches, from <http://technet.microsoft.com/en-us/library/cc750077.aspx>.
1223. Securing Your Web Server, from <http://msdn.microsoft.com/en-us/library/ff648653.aspx>.
1224. Web Server Security and Database Server Security, from <http://www.acunetix.com/websitetecurity/webserver-security>.
1225. Windows IIS Server hardening checklist, from http://media.techtarget.com/searchSecurity/downloads/Windows_IIS_Server_hardening_checklist.pdf?track=L1AP.
1226. IIS Web Server Security, from <http://www.acunetix.com/websitetecurity/iis-security>.
1227. WEB SERVER SECURITY AND DATABASE SERVER SECURITY, from <http://www.itura.net/training/19-web-server-security-and-database-server-security.html>.
1228. Checklist: Securing Your Web Server, from <http://msdn.microsoft.com/en-us/library/ff648198.aspx>.
1229. HTTP Response Splitting, Web Cache Poisoning Attacks, and Related Topics, from http://www.ouah.org/whitepaper_httpresponse.pdf.
1230. Hacking Web Servers, from <http://www.scribd.com/doc/35607686/hacking-Module-11>.
1231. Terms used by Microsoft to describe the various software updates released by it, from <http://www.thewindowsclub.com/terms-used-by-microsoft-to-describe-the-various-software-updates-released-by-it>.
1232. Patch Management Best Practices, from <http://www.oracle.com/technetwork/systems/articles/patch-management-jsp-135385.html>.
1233. Directory Traversal Attacks, from <http://www.acunetix.com/websitetecurity/directory-traversal>.
1234. Jason Chan, (2004), Essentials of Patch Management Policy and Practice, from <http://www.patchmanagement.org/pmessentials.asp>.

1235. Managing Web Server Security, from www.microsoft.com/technet/prodtechnol/windows2000serv/technologies/iis/maintain/featusability/c05iis.mspx.
1236. There are basically three overlapping types of risk:, from <http://www.rduonline.com/webpolicy.mgi>.
1237. Frank Kargl, Jörn Maier, Stefan Schlott, Michael Weber, Protecting Web Servers from Distributed Denial of Service Attacks, from <http://www10.org/cdrom/papers/409/>.
1238. TCPflow (Analyzing Tool), from <http://www.circlemud.org/>.
1239. Radu State, (2008), Hacking Web2, <http://www.aims-conference.org/issnsm-2008/01-WebHacking.pdf>.
1240. Jeremiah Grossman, (2010), 10th Website Security Statistics Report, http://www.whitehatsec.com/home/assets/presentations/10PPT/PPT_stats0910.pdf.
1241. Reto E. Haeni, (1997), Firewall Penetration Testing, <http://bandwidthco.com/whitepapers/netforensics/penetration/Firewall%20Penetration%20Testing.pdf>.
1242. Ali Jahangiri, Google Hacking, <http://www.alijahangiri.org/publication/Google-Hacking-by-Ali-Jahangiri.pdf>.
1243. Networking the networks, from http://www.terena.org/activities/tf-csirt/iodef/docs/itaxonomy_terms.htm.
1244. Network Computing, from <http://www.networkcomputing.com/shared/article/showArticle.jhtml?articleId=8702815&c>.
1245. Barry Wheelbarger, Apache Security, from http://www.cs.ufw.edu/~wilde/StuPres200301/Apache_Security.ppt.
1246. Security issues affecting Apache httpd 2.0.40, from <http://www.apacheweek.com/features/security-v2.0.40>.
1247. Apache Web Server for Windows Lets Remote Users Crash the Web Server Application, from <http://www.securitytracker.com/alerts/2001/Oct/1002543.html>.
1248. The World Wide Web Security FAQ, from <http://www.zentek-international.com/mirrors/www-security-faq/wwwsf1.html>.
1249. HNS Newsletter, from <http://www.net-security.org/dl/newsletter/txt/issue066.txt>.
1250. Ethel the Blog, from http://stommel.tamu.edu/~baum/ethel/2000_12_03_ethel-archive.html.
1251. Survey and Analysis of Available Tools, from <http://www.securecoding.org/authors/articles/may2003/section7.php>.
1252. Information Security Products, from <http://www-935.ibm.com/services/us/index.wss/offerfamily/iss/a1029097>.
1253. Family of Load Balancers, from <http://www.redhillnetworks.com/products/webmux/load-balancer.htm>.
1254. Advanced Defect Tracking Web Edition, from <http://www.borderwave.com/>.
1255. Internet Security and Warfare (ISAW), from <http://technews-isaw.blogspot.com/>.
1256. Experimental Computer System lab, from <http://www.ecsl.cs.sunysb.edu/>.
1257. An Internet Encyclopedia, from <http://www.freesoft.org/CIE/Topics/ssl-draft/3-SPEC.HTM>.
1258. Apache httpd 2.0 vulnerabilities, from http://httpd.apache.org/security/vulnerabilities_20.html.
1259. Apache httpd 1.3 vulnerabilities, from http://httpd.apache.org/security/vulnerabilities_13.html.
1260. Web Hosting, E-commerce, and Domain Registration..., from <http://www.sidetrips.com/>.
1261. Computers, Networking, and Security, from <http://www.cromwell-intl.com/>.
1262. Tony Bradley, (2006), Secure Internet and Network Security, from <http://www.s3kur3.com/>.

1263. Saumil Shah, (2003), One-way Web Hacking, from http://netsquare.com/papers/one_way/one_way.html.
1264. (2010), Case Study: Congressional Web Site Defacements Follow the State of the Union, from <http://praetorianprefect.com/archives/2010/01/congressional-web-site-defacements-follow-the-state-of-the-union/>.
1265. Definition: WEB-SITES DEFACEMENT, from <http://www.freepatentsonline.com/y2010/0107247.html>.
1266. Bodvoc, (2010), An Overview of a Web Server, from <http://bodvoc.wordpress.com/2010/07/02/an-overview-of-a-web-server/>.
1267. (2009), IIS 7.0 Architecture, from <http://www.gandhipritesh.com/2009/05/iis-70-architecture.html>.
1268. (2001), Defaced Websites, from <http://attrition.org/mirror/attrition/>.
1269. Robert Auger, Server Misconfiguration, from <http://projects.webappsec.org/w/page/13246959/Server-Misconfiguration>.
1270. Insecure Configuration Management, from http://www.owasp.org/index.php/Insecure_Configuration_Management.
1271. (2009), hostmap 0.2 – Automatic Hostname & Virtual Hosts Discovery Tool, from <http://www.darknet.org.uk/tag/web-server-hacking/>.
1272. (2009), reDuh – TCP Redirection over HTTP, from <http://www.darknet.org.uk/tag/web-server-hacking/>.
1273. httprecon – Advanced Web Server Fingerprinting <http://www.darknet.org.uk/tag/web-server-hacking/>.
1274. Robert Auger, HTTP Response Splitting <http://projects.webappsec.org/w/page/13246931/HTTP-Response-Splitting>.
1275. HTTP Response Splitting, from http://www.owasp.org/index.php/HTTP_Response_Splitting.
1276. Introduction to HTTP Response Splitting, from <http://www.securiteam.com/securityreviews/5WP0E2KFGK.html>.
1277. Tunneling protocol, from http://en.wikipedia.org/wiki/Tunneling_protocol.
1278. Whois, from <http://tools.whois.net>.
1279. Traceroute, from <http://whatismyipaddress.com/traceroute-tool>.
1280. ActiveWhois, from <http://www.johnru.com/>.
1281. Netcraft, from <http://searchdns.netcraft.com/?host>.
1282. httprecon, from <http://www.computech.ch/projekte/httprecon/?s=download>.
1283. ID Serve, from <http://www.grc.com>.
1284. HTTrack Website Copier, from <http://www.httrack.com/page/2/>.
1285. WebCopier Pro, from http://www.maximumsoft.com/products/wc_pro/overview.html.
1286. BlackWidow, from <http://softbytelabs.com/us/downloads.html>.
1287. Hamster, from <http://erratasec.blogspot.in/2009/03/hamster-20-and-ferret-20.html>.
1288. Firesheep, from <http://codebutler.github.io/firesheep/>.
1289. Brutus, from <http://www.hoobie.net/brutus/brutus-download.html>.
1290. Metasploit, from <http://www.metasploit.com/download/>.
1291. WFetch, from <http://download.microsoft.com/download/d/e/5/de5351d6-4463-4cc3-a27c-3e2274263c43/wfetch.exe> (<http://www.microsoft.com/downloads/details.aspx?FamilyID=56fc92ee-a71a-4c73-b628-ade629c89499&DisplayLang=en>).
1292. Brutus, from <http://www.hoobie.net/brutus/brutus-download.html>.
1293. Internet Password Recovery Toolbox, from http://www.rixler.com/password_recovery_toolbox.htm.

1294. Microsoft Baseline Security Analyzer (MBSA), from <http://www.microsoft.com/en-us/download/details.aspx?id=7558>.
1295. Altiris Client Management Suite, from <http://www.symantec.com/client-management-suite/trialware>.
1296. Prism Patch Manager, from <http://www.newboundary.com/products/prism-patch-manager/trial>.
1297. MaaS360® Patch Analyzer Tool, from <http://www.maas360.com/tools-and-trials/downloads/>.
1298. Kaseya Security Patch Management, from <http://www.kaseya.com/features/patch-management.aspx#>.
1299. Secunia CSI, from <http://secunia.com/products/corporate/csi/>.
1300. ZENworks® Patch Management, from <http://www.novell.com>.
1301. Lumension® Patch and Remediation, from <http://www.lumension.com>.
1302. Security Manager Plus, from <http://www.manageengine.com/products/security-manager/download.html>.
1303. VMware vCenter Protect, from <http://www.shavlik.com/downloads.aspx>.
1304. Syhunt Dynamic, from <http://www.syhunt.com/?n=Syhunt.Dynamic>.
1305. N-Stalker Web Application Security Scanner , from <http://www.nstalker.com/products/editions/free/>.
1306. Wikto, from <http://www.sensepost.com>.
1307. Acunetix Web Vulnerability Scanner, from <http://www.acunetix.com/vulnerability-scanner/download.htm>.
1308. HackAlert, from http://www.armorize.com/index.php?link_id=register.
1309. QualysGuard Malware Detection, from <http://www.qualys.com/forms/trials/stopmalware/>.
1310. Retina CS, from <http://www.beyondtrust.com/Landers/TY-Page-RetinaCSCommunity/index.html>.
1311. Nscan, from <http://nscan.hypermart.net>.
1312. NetIQ Secure ConfigurationManager, from <https://www.netiq.com/products/secure-configuration-manager/>.
1313. SAINT, from <http://www.saintcorporation.com/products/software/saintScanner.html>.
1314. HP WebInspect, from <https://download.hpsmartupdate.com/webinspect/>.
1315. Arirang, from <http://www.monkey.org/~pilot/arirang/>.
1316. N-Stalker Web Application Security Scanner , from <http://www.nstalker.com/products/editions/free/>.
1317. Infiltrator, from <http://www.infiltration-systems.com/download.shtml>.
1318. WebCruiser, from <http://sec4app.com/download.htm>.
1319. dotDefender, from <http://www.aplicure.com/Products/>.
1320. Core Impact Professional, from <http://www.coresecurity.com>.
1321. Immunity CANVAS, from <http://www.immunitysec.com/downloads.shtml>.

Module 13: Hacking Web Applications

1322. Parameter Tampering, from http://www.imperva.com/resources/glossary/parameter_tampering.html.
1323. Connection String Injection Attacks, from <http://msdn.microsoft.com/en-us/library/ms254947.aspx>.
1324. A6 2004 Injection Flaws, from https://www.owasp.org/index.php/A6_2004_Injection_Flaws.
1325. Connection String Parameter Pollution Attacks, from http://blackhat.com/presentations/bh-dc-10/Alonso_Chema/Blackhat-DC-2010-Alonso-Connection-String-Parameter-Pollution-wp.pdf.
1326. Session Prediction, from https://www.owasp.org/index.php?title=Session_Prediction&setlang=en.
1327. Buffer Overflow, from <http://projects.webappsec.org/w/page/13246916/Buffer-Overflow>.

1328. Managed Application Firewall, from http://www.secureworks.com/resources/articles/other_articles/2010-waf.
1329. Do you write secure code?, from <http://www.slideshare.net/yuvalgo/do-you-write-secure-code-by-erez-metula>.
1330. Web Parameter Tampering, from https://www.owasp.org/index.php/Web_Parameter_Tampering.
1331. Path Traversal, from https://www.owasp.org/index.php/Path_traversal.
1332. Top 10 2010-A6-Security Misconfiguration, from https://www.owasp.org/index.php/Top_10_2010-A6-Security_Misconfiguration.
1333. Common Security Mistakes in Web Applications, from <http://roobon.net/2011/06/01/common-security-mistakes-in-web-applications>.
1334. LDAP Injection & BLIND LDAP Injection, from <http://www.blackhat.com/presentations/bh-europe-08/Alonso-Parada/Whitepaper/bh-eu-08-alonso-parada-WP.pdf>.
1335. Parameter Manipulation, from <http://www.cgisecurity.com/owasp/html/ch11s04.html>.
1336. Cross-site Scripting (XSS), from [https://www.owasp.org/index.php/Cross-site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)).
1337. XSS Filter Evasion Cheat Sheet, from https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet.
1338. Cross-Site Request Forgery (CSRF) Attack Lab, from http://www.cis.syr.edu/~wedu/seed/Labs/Attacks_CSRF/CSRF.pdf.
1339. Cross-Site Request Forgeries, from <http://www.shiflett.org/articles/cross-site-request-forgeries>.
1340. Webapplication Attack : DOS and DDOS attack, from <http://funwhichuwant.blogspot.in/2012/10/webapplication-attack-dos-and-ddos.html>.
1341. Buffer Overflow, from <http://projects.webappsec.org/w/page/13246916/Buffer%20Overflow>.
1342. Cookie Poisoning, from http://www.imperva.com/resources/glossary/cookie_poisoning.html.
1343. Wen Application Vulnerabilities, from <http://www.slideshare.net/technoplex/web-application-vulnerabilities>.
1344. Attacking XML Security Message Oriented Madness, XML Worms and Web Service Security Sanity, from <http://www.slideshare.net/yusufmotiwala/attacking-xml-security>.
1345. Managing Web Services, from <http://docs.oracle.com/cd/E19316-01/820-4335/gbbjk/index.html>.
1346. Web Services Hacking And Hardening, from <http://www.slideshare.net/rnewton/web-services-hacking-and-hardening>.
1347. Advanced Web Services Hacking, from <http://www.slideshare.net/shreeraj/advanced-web-services-hacking>.
1348. Hacking Web 2.0 - Defending Ajax and Web Service, from <http://www.slideshare.net/shreeraj/hacking-web-2.0-defending-ajax-and-web-services-hitb-2007-dubai>.
1349. All-Purpose Tools, from <http://www.securonet.biz/tools.htm>.
1350. Error executing child request for ChartImg.axd, from <http://social.msdn.microsoft.com/Forums/en-US/MSWinWebChart/thread/115d7f31-e4a8-4c09-b558-4db2cf1e83e7>.
1351. Session Prediction, from https://www.owasp.org/index.php?title=Session_Prediction&setlang=en.
1352. Building Connection Strings, from [http://msdn.microsoft.com/en-us/library/ms254947\(v=vs.80\).aspx](http://msdn.microsoft.com/en-us/library/ms254947(v=vs.80).aspx).
1353. DOS ATTACKS USING SQL WILDCARDS, from http://hax.tor.hu/read/MSSQL_DoS/wildcard_attacks.pdf.
1354. Understanding Web Services Attacks, from http://www.datacom.cz/files_datacom/understanding_webservicesattacks_0.pdf.
1355. Spheon JSOAP - InterOp: MS SOAP ToolKit 3.0 (typed), from <http://soap.fmui.de/interop/interop2001MSSOAPToolKitTyped.html>.

1356. Web Services Attacks & Countermeasures, from http://www.interop.com/lasvegas/2004/presentations/downloads/sc04_c_sima.pdf .
1357. Testing for HTTP Splitting/Smuggling (OWASP-DV-016), from https://www.owasp.org/index.php/Testing_for_HTTP_Exploit.
1358. Testing for SQL Wildcard Attacks (OWASP-DS-001), from [https://www.owasp.org/index.php/Testing_for_SQL_Wildcard_Attacks_\(OWASP-DS-001\)](https://www.owasp.org/index.php/Testing_for_SQL_Wildcard_Attacks_(OWASP-DS-001)) .
1359. Testing for DoS User Specified Object Allocation (OWASP-DS-004), from [https://www.owasp.org/index.php/Testing_for_DoS_User_Specified_Object_Allocation_\(OWASP-DS-004\)](https://www.owasp.org/index.php/Testing_for_DoS_User_Specified_Object_Allocation_(OWASP-DS-004)) .
1360. Testing for Storing too Much Data in Session (OWASP-DS-008), from [https://www.owasp.org/index.php/Testing_for_Storing_too_Much_Data_in_Session_\(OWASP-DS-008\)](https://www.owasp.org/index.php/Testing_for_Storing_too_Much_Data_in_Session_(OWASP-DS-008)).
1361. Testing for Naughty SOAP Attachments, from <http://nilminus.wordpress.com/web-application-penetration-testing/web-services-testing/testing-for-naughty-soap-attachments>.
1362. Testing for AJAX (OWASP-AJ-002), from [https://www.owasp.org/index.php?title=Testing_for_AJAX_\(OWASP-AJ-002\)&setlang=es](https://www.owasp.org/index.php?title=Testing_for_AJAX_(OWASP-AJ-002)&setlang=es).
1363. Common Web-Based Applications Attacks, Available from http://www.aplicure.com/Common_Web_Based_Applications_Attacks#2._Injection_Flaws.
1364. Bart Puype, WGET for Windows (win32), version 1.11.4, Available from <http://users.ugent.be/~bpuype/wget/>.
1365. Andrés Riancho, Web Application Attack and Audit Framework, Available from <http://w3af.sourceforge.net/>.
1366. Market Leading Protection for Web Applications, Available from <http://www.imperva.com/products/web-application-firewall.html>.
1367. Vulnerability, from <http://www.citi.umich.edu/projects/itss/lectures/lecture-20.pdf>.
1368. DATA IIS Vulnerability, from <http://www.ciac.org/ciac/bulletins/k-068.shtml>.
1369. RPC DCOM Vulnerability, from <http://seclists.org/bugtraq/2003/Oct/0151.html>.
1370. ASN Exploits, from www.itworldcanada.com.
1371. [PDF] Introduction to Web Applications and Security, from http://books.mcgraw-hill.com/downloads/products//007222438X/007222438X_ch01.pdf.
1372. The behaviors and tools of today's hacker, from www.symantec.com/symadvantage/014/hacker.html.
1373. Paper -- Cross Site Scripting, from www.technicalinfo.net/papers/CSS.html.
1374. Host Vulnerability, from <http://www.cit.cornell.edu/security/scanning/sample.html>.
1375. Joseph Seaman, (2003), Web Application Security from www.itsa.ufl.edu/slideshows/2003/WebAppSec.ppt.
1376. Vulnerability Management Commitment and Disclosure Policy, from <http://www.symantec.com/security/>.
1377. Bug Tracking Software Links, from <http://www.bug-track.com/main/links.jsp>.
1378. Mike Benham, (2002), Internet Explorer SSL Vulnerability, from <http://www.securiteam.com/windowsntfocus/5JP0E0081M.html>.
1379. The 21 Primary Classes of Web Application Threats, from www.netcontinuum.com/securityCentral/TopThreatTypes/index.cfm.
1380. Paper: HTML Code Injection and Cross-site scripting, from <http://www.technicalinfo.net/papers/CSS.html>.
1381. IS YOUR WEBSITE HACKABLE?, from www.acunetix.com/vulnerability-scanner/wvsbrochure.pdf.

1382. Regular Expressions: curl Simplifies Web Retrieval, from <http://www.unixreview.com/documents/s=1820/uni1011713175619/0201i.htm>.
1383. FWSM URL Filtering Solution TCP ACL Bypass Vulnerability, from www.cisco.com.
1384. Zero Day Exploits: The Holy Grail, from www.netsecurity.about.com.
1385. What is parameter tampering?, from www.imperva.com.
1386. AFITC 2001, from www.whitehatsec.com.
1387. Toelichting aanvalstechnieken, from www.nedsecure.nl.
1388. Cross-Site Scripting, Injection Flaws, OWASP Web Application Security Top Ten List and Buffer Overflow, from www.owasp.org.
1389. Hacker Protection from SQL Injection – SPI Dynamics, from www.spidynamics.com.
1390. Changing Your Password, How Hackers Get Hold of Passwords, from www.lockdown.co.uk/?pg=password_guide.
1391. George Shaffer, Modus Operandi of an Attacker Using a Password Cracker, from http://geodsoft.com/howto/password/cracking_passwords.htm.
1392. Robert J. Shimonski, (2002), Hacking techniques, from www.ibm.com/developerworks/library/s-crack.
1393. Mark Culphey, Query String, from www.cgisecurity.com/owasp/html/ch11s04.html.
1394. Edward Skoudis, Authforce, from (2005), http://searchsecurity.techtarget.com/searchSecurity/downloads/Skoudis_ch07.pdf.
1395. Sarah Granger, (2002), A Guide To Better Password Practices, from www.securityfocus.com/infocus/1537.
1396. Bad Password Examples, from <http://www.spy-hill.com/~myers/help/Passwords.html>.
1397. Microsoft Password Checker, from http://www.microsoft.com/athome/security/privacy/password_checker.mspx.
1398. Mehdi Mousavi, What an ISAPI extension is?, from http://www.codeproject.com/KB/ISAPI/isapi_extensions.aspx.
1399. Maximum Security - Chapter 10 - Password Crackers, from http://www.windowsecurity.com/whitepapers/Maximum_Security__Chapter_10__Password_Crackers_.html.
1400. Patch improves the TCP Initial Sequence Number Randomness, from <http://www.securiteam.com/windowsntfocus/3V5QBQKPPU.html>.
1401. Mark Russinovich, (2008), TCP View for Windows, from [http://technet.microsoft.com/sysinternals/bb897437\(en-us\).aspx](http://technet.microsoft.com/sysinternals/bb897437(en-us).aspx).
1402. Admin Knowledge Base section, from <http://www.windowsnetworking.com/kbase/WindowsTips/WindowsTips/WindowsNT/AdminTips/Utilities/TCPView.htm>.
1403. Web Application Security, from <http://www.securityfocus.com/archive/107/223386/2001-10-28/2001-11-02/0>.
1404. Nikola Strahija, (2002), Introduction to password cracking, from <http://www.xatrix.org/article.php?s=1758>.
1405. Password cracking, http://www-128.ibm.com/developerworks/security/library/s-crack/password_cracking.html.
1406. Password cracker, from http://searchfinancialsecurity.techtarget.com/sDefinition/0,,sid185_gci536994,00.html.

1407. David P. Kormann and Aviel D. Rubin, Risks of the Passport Single Signon Protocol, from <http://www.cs.jhu.edu/~rubin/courses/sp03/papers/passport.pdf>.
1408. Abel Banda, (2003), ASP.NET Forms Authentication, from <http://www.ondotnet.com/pub/a/dotnet/2003/01/06/formsauthp1.html>.
1409. Erika, (2006), Microsoft Security Bulletin MS02-048, from <http://www.microsoft.com/technet/security/Bulletin/MS02-048.mspx>.
1410. Jeff Williams, (2006), Cross-Site Scripting, Injection Flaws, OWASP Web Application Security Top Ten List and Buffer Overflow, from <http://www.owasp.org/>.
1411. Sarah Granger, (2002), A Guide To Better Password Practices, from <http://www.securityfocus.com/infocus/1537>.
1412. Gaining Access Using Application and Operating System Attacks, from http://searchsecurity.techtarget.com/searchSecurity/downloads/Skoudis_ch07.pdf.
1413. Rob Shimonski, (2002), Hacking techniques, from <http://www-128.ibm.com/developerworks/library/s-crack/>.
1414. Password Guidelines, from http://www.lockdown.co.uk/?pg=password_guide.
1415. Biometric Education: Fingerprint, from <http://www.barcode.ro/tutorials/biometrics/fingerprint.html>.
1416. Kimon Rethis, (2006), Biometrics Authentication, from <http://www.csun.edu/>.
1417. IPSec Authentication and Authorization Models, from <http://www.ciscopress.com/articles/article.asp?p=421514&seqNum=4%20-%2031k%20-&rl=1>.
1418. Digital Certificates, from <http://www.bitpipe.com/tlist/Digital-Certificates.html>.
1419. John, HTTP Authentication: Basic and Digest Access Authentication, from <http://www.ietf.org/rfc/rfc2617.txt>.
1420. Authentication, Authorization, and Access Control, from <http://httpd.apache.org/docs/>.
1421. Functions and Procedures: Basic Authentication, from <http://www.zeitungsjunge.de/delphi/mime/Help/DIMime.htm>.
1422. The Cross-Site Scripting (XSS) FAQ, from <http://www.cgisecurity.com/xss-faq.html>.
1423. Input Validation Cheat Sheet, from http://michaeldaw.org/input_validation_cheat_sheet.
1424. Quick Security Reference - Cross-Site Scripting.docx, from <http://download.microsoft.com/download/E/E/7/EE7B9CF4-6A59-4832-8EDE-B018175F4610/Quick%20Security%20Reference%20-%20Cross-Site%20Scripting.docx>.
1425. Web Application Penetration Testing, from http://www.owasp.org/index.php/Web_Application_Penetration_Testing.
1426. Jeff Orloff, The Big Website Guide to a Hacking Attack, from <http://www.aplicure.com/blog/big-website-guide-to-a-hacking-attack>.
1427. What is Cross-Site Scripting (XSS)?, from <http://www.aplicure.com/blog/what-is-cross-site-scripting>.
1428. LDAP Filters, from <http://www.selfadsi.org/ldap-filter.htm>.
1429. Paul Lee, (2002), Cross-site scripting, from <http://www.ibm.com/developerworks/tivoli/library/s-csscript/>.
1430. XSS (Cross Site Scripting) Prevention Cheat Sheet, from http://www.owasp.org/index.php/XSS_%28Cross_Site_Scripting%29_Prevention_Cheat_Sheet.
1431. Amit Klein, (2005), DOM Based Cross Site Scripting or XSS of the Third Kind, from <http://www.webappsec.org/projects/articles/071105.shtml>.
1432. Samoa: Formal Tools for Securing Web Services, from <http://research.microsoft.com/en-us/projects/samoa/>.

1433. RSnake "XSS (Cross Site Scripting) Cheat Sheet Esp: for filter evasion", from <http://ha.ckers.org/xss.html>.
1434. Microsoft's Anti-Cross Site Scripting Security Runtime Engine Sample - AntiXSS 3.1, from <http://davidhayden.com/blog/dave/archive/2009/09/22/antixsssample.aspx>.
1435. Philip Tellis, (2010), Common Security Mistakes in Web Applications, from <http://www.smashingmagazine.com/2010/10/18/common-security-mistakes-in-web-applications/>.
1436. J.D. Meier, Alex Mackman, Michael Dunner, Srinath Vasireddy, Ray Escamilla and Anandha Murukan, (2003), Improving Web Application Security: Threats and Countermeasures, from <http://msdn.microsoft.com/en-us/library/ff649874.aspx>.
1437. Alex Homer, Components and Web Application Architecture, from <http://technet.microsoft.com/en-us/library/bb727121.aspx>.
1438. Ryan Barnett, (2011), Web-Hacking-Incident-Database, from <http://projects.webappsec.org/Web-Hacking-Incident-Database#TopApplicationWeaknesses>.
1439. (2009), Path Traversal, from http://www.owasp.org/index.php/Path_Traversal.
1440. (2010), Web Parameter Tampering, from http://www.owasp.org/index.php/Web_Parameter_Tampering.
1441. Unvalidated Input, from http://www.owasp.org/index.php/Unvalidated_Input#Examples_and_References.
1442. Kevin Beaver, The importance of input validation, from http://searchsoftwarequality.techtarget.com/tip/0,289483,sid92_gci1214373_mem1,00.html.
1443. (2010), Validating Input, from <http://developer.apple.com/library/ios/#documentation/Security/Conceptual/SecureCodingGuide/Articles/ValidatingInput.html>.
1444. Seth Fogie, (2006), Code Injection Explained, from <http://www.informit.com/guides/content.aspx?g=security&seqNum=226>.
1445. Code injection, from http://en.wikipedia.org/wiki/Code_injection.
1446. Injection Prevention Cheat Sheet, from http://www.owasp.org/index.php/Injection_Prevention_Cheat_Sheet.
1447. Remote file inclusion, from http://en.wikipedia.org/wiki/Remote_file_inclusion.
1448. Robert Auger, (2011), LDAP Injection, from <http://projects.webappsec.org/LDAP-Injection>.
1449. Testing for LDAP Injection (OWASP-DV-006), from http://www.owasp.org/index.php/Testing_for_LDAP_Injection_%28OWASP-DV-006%29.
1450. Shreeraj Shah, (2006), Top 10 Web 2.0 Attack Vectors, from <http://www.net-security.org/article.php?id=949>.
1451. Robert Auger, (2010), Threat Classification, from <http://projects.webappsec.org/Threat-Classification>.
1452. (2006), Preventing HTML form tampering, from <http://advosys.ca/papers/web/60-form-tampering.html>.
1453. (2010), Cross-site Scripting (XSS), from http://www.owasp.org/index.php/Cross-site_Scripting_%28XSS%29.
1454. Paul Lee, Cross-site scripting, from <http://www.ibm.com/developerworks/tivoli/library/s-csscript/>.
1455. Cross-site scripting, from http://en.wikipedia.org/wiki/Cross-site_scripting.
1456. DOM Based XSS, from http://www.owasp.org/index.php/DOM_Based_XSS.
1457. Phil Haack, (2009), CSRF Attacks and Web Forms, from <http://haacked.com/archive/2009/04/02/csrf-webforms.aspx>.
1458. Chris Shiflett, (2004), Cross-Site Request Forgeries, from <http://shiflett.org/articles/cross-site-request-forgeries>.

1459. Robert Auger, (2010), The Cross-Site Request Forgery (CSRF/XSRF) FAQ, from <http://www.cgisecurity.com/csrf-faq.html>.
1460. Application Denial of Service, from http://www.owasp.org/index.php/Application_Denial_of_Service.
1461. Cookie Poisoning, from http://www.imperva.com/resources/glossary/cookie_poisoning.html.
1462. Cookie Poisoning how to, from <http://forum.intern0t.net/security-tutorials-guides/2270-cookie-poisoning-how.html>.
1463. Broken Authentication and Session Management, from http://www.owasp.org/index.php/Broken.Authentication_and.Session_Management.
1464. Robert Auger, (2010), Buffer Overflow, from <http://projects.webappsec.org/w/page/13246916/Buffer-Overflow>.
1465. (2009), Buffer Overflow, from http://www.owasp.org/index.php/Buffer_Overflow.
1466. Free XML tools and software, from <http://www.garshol.priv.no/download/xmltools/>.
1467. SYS-CON tv, (2005), Anatomy of a Web Services Attack, from <http://education.sys-con.com/node/80899>.
1468. Robert Auger, (2010), Brute Force, from <http://projects.webappsec.org/w/page/13246915/Brute-Force>.
1469. Ian de Villiers, sensepost j-baah, from <http://www.sensepost.com/labs/tools/pentest/j-baah>.
1470. (2009), Session Prediction, from http://www.owasp.org/index.php/Session_Prediction.
1471. Robert Auger, (2010), XPath Injection, from <http://projects.webappsec.org/w/page/13247005/XPath-Injection>.
1472. (2009), XPATH Injection, from http://www.owasp.org/index.php/XPATH_Injection.
1473. SmartWhois, from <http://www.tamos.com/download/main/index.php>.
1474. Netcraft, from <http://searchdns.netcraft.com/?host>.
1475. Whois, from <http://tools.whois.net>.
1476. DNSstuff, from <http://www.dnsstuff.com>.
1477. dnsstuff, from <http://www.dnsstuff.com/>.
1478. network-tools, from <http://network-tools.com/>.
1479. DNS, from <http://e-dns.org>.
1480. DomainTools, from <http://www.domaintools.com>.
1481. WhatsUp PortScanner Tool, from http://www.whatsupgold.com/products/download/network_management.aspx?k_id=port-scan.
1482. hping, from <http://www.hping.org/download.php>.
1483. Sandcat Browser, from <http://www.syhunt.com/?n=Sandcat.Browser>.
1484. Netcat, from <http://sourceforge.net/projects/netcat/files/latest/download?source=files>.
1485. ID Serve, from <http://www.grc.com>.
1486. Netcraft, from <http://toolbar.netcraft.com>.
1487. OWASP Zed Attack Proxy, from https://code.google.com/p/zaproxy/downloads/detail?name=ZAP_2.0.0_Windows.exe&can=2&q=.
1488. Burp Spider, from <http://blog.portswigger.net/2008/11/mobp-all-new-burp-spider.html>.
1489. WebScarab, from https://www.owasp.org/index.php/Category:OWASP_WebScarab_Project.
1490. Burp Suite, from <http://blog.portswigger.net/2008/11/mobp-all-new-burp-spider.html>.
1491. Brutus, from <http://www.hoobie.net/brutus/brutus-download.html>.
1492. Sensepost's Crowbar, from <http://research.sensepost.com/tools/web/j-baah>.

1493. UrlScan, from
<http://www.microsoft.com/web/gallery/install.aspx?appsxml=&appid=UrlScan%3bUrlScan>.
1494. Nikto, from <http://www.cirt.net/nikto2>.
1495. Nessus, from <http://www.tenable.com/products/nessus/select-your-operating-system>.
1496. Acunetix Web Vulnerability Scanner, from <http://www.acunetix.com/vulnerability-scanner/download.htm>.
1497. WebInspect, from <https://download.hpsmartupdate.com/webinspect/>.
1498. HttpPrint, from <http://net-square.com/httprint.html>.
1499. WebScarab, from https://www.owasp.org/index.php/Category:OWASP_WebScarab_Project.
1500. GNU Wget, from <ftp://ftp.gnu.org/gnu/wget/>.
1501. Teleport Pro, from <http://www.tenmax.com/teleport/pro/download.htm>.
1502. BlackWidow, from <http://softbytelabs.com/us/downloads.html>.
1503. Brutus, from <http://www.hoobie.net/brutus/brutus-download.html>.
1504. THC-Hydra, from <http://www.thc.org/thc-hydra/>.
1505. soapUI, from <http://www.soapui.org/>.
1506. CookieDigger, from <http://www.mcafee.com/apps/free-tools/termsofuse.aspx?url=/us/downloads/free-tools/cookiebuster.aspx>.
1507. WebScarab, from https://www.owasp.org/index.php/Category:OWASP_WebScarab_Project.
1508. Instant Source, from <http://www.blazingtools.com/downloads.html#is>.
1509. HttpBee, from <http://www.o0o.nu/projects/stif>.
1510. w3af, from <http://w3af.sourceforge.net>.
1511. Teleport Pro, from <http://www.tenmax.com/teleport/pro/download.htm>.
1512. GNU Wget, from <ftp://ftp.gnu.org/gnu/wget/>.
1513. WebCopier Pro, from http://www.maximumsoft.com/products/wc_pro/overview.html.
1514. HTTrack Website Copier, from <http://www.httrack.com/page/2/>.
1515. BlackWidow, from <http://softbytelabs.com/us/downloads.html>.
1516. cURL, from <http://curl.haxx.se/download.html>.
1517. MileSCAN ParosPro, from
http://www.milescan.com/hk/index.php?option=com_content&view=article&id=15&Itemid=157.
1518. Acunetix Web Vulnerability Scanner, from <http://www.acunetix.com/vulnerability-scanner/download.htm>.
1519. Watcher Web Security Tool, from <http://websecuritytool.codeplex.com/downloads/get/62386>.
1520. Netsparker, from <http://www.mavitunasecurity.com/>.
1521. N-Stalker Web Application Security Scanner , from <http://www.nstalker.com/products/editions/free/>.
1522. VampireScan, from <http://www.vampiretech.com/store/?product=vampirescan-cloud-security-standard-edition>.
1523. SandcatMini, from <http://www.syhunt.com/?n=Syhunt.Mini>.
1524. Websecurify, from
<https://code.google.com/p/websecurify/downloads/detail?name=Websecurify%20Suite%201.0.0.exe&can=2&q=>.
1525. OWASP ZAP, from
https://code.google.com/p/zaproxy/downloads/detail?name=ZAP_2.0.0_Windows.exe&can=2&q=.

1526. NetBrute, from <http://www.rawlogic.com/netbrute/>.
1527. skipfish, from <https://code.google.com/p/skipfish/>.
1528. X5s, from <http://xss.codeplex.com/downloads/get/115610>.
1529. SecuBat Vulnerability Scanner, from <http://secubat.codeplex.com/>.
1530. WSSA - Web Site Security Scanning Service, from <https://secure.beyondsecurity.com/vulnerability-scanner-signup?step=1>.
1531. SPIKE Proxy, from <http://www.immunitysec.com/resources-freesoftware.shtml>.
1532. Ratproxy, from <https://code.google.com/p/ratproxy/>.
1533. Wapiti, from <http://wapiti.sourceforge.net/>.
1534. Syhunt Hybrid, from <http://www.syhunt.com/?n=Syhunt.Dynamic>.
1535. WebWatchBot, from <http://www.exclamationsoft.com/ExclamationSoft/download/instructions/html.asp?product=WebWatchBot&fe=no>.
1536. Exploit-Me, from <http://labs.securitycompass.com/exploit-me/>.
1537. KeepNI, from <http://www.keepni.com/>.
1538. WSDigger, from <http://www.mcafee.com/apps/free-tools/termsofuse.aspx?url=/us/downloads/free-tools/wsdigger.aspx>.
1539. Arachni, from <http://arachni-scanner.com/latest>.
1540. XSSS, from <http://www.sven.de/xsss/>.
1541. Vega, from http://www.subgraph.com/vega_download.php.
1542. dotDefender, from <http://www.aplicure.com/Products/>.
1543. ServerDefender VP, from <http://www.port80software.com/products/serverdefendervp/try>.
1544. Radware's AppWall, from <http://www.radware.com/Products/ApplicationDelivery/AppWall/default.aspx>.
1545. Barracuda Web Application Firewall, from <https://www.barracuda.com/products/webapplicationfirewall>.
1546. ThreatSentry, from http://www.privacyware.com/TS_Registration.html.
1547. Stingray Application Firewall, from .
1548. QualysGuard WAF, from <http://www.qualys.com/forms/web-application-firewall/>.
1549. IBM Security AppScan, from <http://www-01.ibm.com/software/awdtools/appscan/>.
1550. ThreatRadar, from http://www.imperva.com/products/wsc_threatradar-reputation-services.html.
1551. Trustwave WebDefend, from <https://www.trustwave.com/web-application-firewall/#overview>.
1552. ModSecurity, from <http://www.modsecurity.org/download/>.
1553. Cyberoam's Web Application Firewall, from <http://www.cyberoam.com/webapplicationfirewall.html>.
1554. Burp Proxy, from <http://blog.portswigger.net/2008/11/mobp-all-new-burp-spider.html>.
1555. WebScarab, from https://www.owasp.org/index.php/Category:OWASP_WebScarab_Project.
1556. TamperIE, from <http://www.bayden.com/tamperie/>.
1557. Tamper Data, from <https://addons.mozilla.org/en-US/firefox/addon/tamper-data/eula/79565?src=dp-btn-primary>.
1558. Amap, from <http://www.thc.org/thc-amap/>.
1559. Netcat, from <http://sourceforge.net/projects/netcat/files/latest/download?source=files>.

1560. OWASP CAL9000, from https://www.owasp.org/index.php/Category:OWASP_Download.
1561. Hackvertor, from <https://hackvertor.co.uk/public>.
1562. BeEF, from <http://beefproject.com/>.
1563. XSS-Proxy, from <http://sourceforge.net/projects/xss-proxy/files/latest/download>.
1564. Backframe, from <http://www.gnucitizen.org/blog/backframe/>.
1565. XSS Assistant, from <https://code.google.com/p/xss-assistant/>.
1566. SWFIntruder, from <https://code.google.com/p/swfintruder/downloads/detail?name=swfintruder-0.9.1.tgz&can=2&q=>.
1567. Flare, from <http://www.nowrap.de/flare.html>.
1568. MTASC, from <http://www.mtasc.org/>.
1569. Flasm, from <http://flasm.sourceforge.net/>.
1570. swfmill, from <http://swfmill.org/>.
1571. Debugger Version of Flash Plugin/Player, from <http://www.adobe.com/support/flashplayer/downloads.html#fp11>.
1572. SQLiX, from https://www.owasp.org/index.php/Category:OWASP_SQLiX_Project.
1573. sqlninja, from <http://sqlninja.sourceforge.net/download.html>.
1574. Sqldumper, from <http://sqldumper.ruizata.com/>.
1575. sqldftools, from <http://packetstormsecurity.com/files/download/43795/sqlbftools-1.2.tar.gz>.
1576. Softerra LDAP Browser, from <http://www.ldapadministrator.com/download.htm>.
1577. Hibernate, from <http://www.hibernate.org/downloads>.
1578. NHibernate, from <http://nhforge.org/>.
1579. Ruby On Rails, from <http://rubyinstaller.org/downloads>.
1580. String searcher: grep, from http://sourceforge.net/projects/gnuwin32/files/grep/2.5.4/grep-2.5.4-bin.zip/download?use_mirror=nchc.
1581. wsChess, from <http://www.net-square.com/wsChess.html>.
1582. Soaplite, from <http://soaplite.com/download.html>.
1583. cURL, from <http://curl.haxx.se/download.html>.
1584. Perl, from <http://www.activestate.com/activeperl/downloads/thank-you?dl=http://downloads.activestate.com/ActivePerl/releases/5.16.3.1603/ActivePerl-5.16.3.1603-MSWin32-x64-296746.msi>.
1585. UDDI Browser, from http://sourceforge.net/projects/uddibrowser/files/uddibrowser/UDDI%20Browser%200.2%20Binaries/u b-0.2-bin.zip/download?use_mirror=nchc&download=1.
1586. WSIndex, from <http://www.wsindex.org/>.
1587. Xmethods, from <http://www.xmethods.net/ve2/index.po>.
1588. WSDigger, from <http://www.mcafee.com/apps/free-tools/termsofuse.aspx?url=/us/downloads/free-tools/wsdigger.aspx>.
1589. Sprajax, from https://www.owasp.org/index.php/Category:OWASP_Sprajax_Project.

Module 14: SQL Injection

1590. Advanced SQL Injection, from <http://www.slideshare.net/Sandra4211/advanced-sql-injection-3958094>.
1591. Advanced SQL Injection, from <http://www.slideshare.net/devteev/advanced-sql-injection-eng>.

1592. SQL injection, from <http://searchsqlserver.techtarget.com/feature/SQL-injection>.
1593. SQL Injection, from http://hakipedia.com/index.php/SQL_Injection.
1594. ERROR SQL INJECTION – DETECTION, from <http://www.evilsq.com/main/page2.php>.
1595. What is SQL Injection?, from <http://www.secpoint.com/what-is-sql-injection.html>.
1596. Securing Oracle Database from Search Engines Attack, from <http://www.ijana.in/papers/V4I2-1.pdf>.
1597. Stop SQL Injection Attacks Before They Stop You, from <http://msdn.microsoft.com/en-us/magazine/cc163917.aspx>.
1598. Rise in SQL Injection Attacks Exploiting Unverified User Data Input, from <http://technet.microsoft.com/en-us/security/advisory/954462>.
1599. Injection Protection, from [http://msdn.microsoft.com/en-us/library/aa224806\(v=sql.80\).aspx](http://msdn.microsoft.com/en-us/library/aa224806(v=sql.80).aspx).
1600. Understanding SQL Injection, from http://www.cisco.com/web/about/security/intelligence/sql_injection.html.
1601. SQL INJECTION – Tutorial, from <http://elitezone.forumotion.bz/t77-sql-injection-tutorial>.
1602. System testing, from http://en.wikipedia.org/wiki/System_testing.
1603. Testing for SQL Injection (OWASP-DV-005), from [https://www.owasp.org/index.php/Testing_for_SQL_Injection_\(OWASP-DV-005\)](https://www.owasp.org/index.php/Testing_for_SQL_Injection_(OWASP-DV-005)).
1604. SQL Injection Cheat Sheet, from <http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>.
1605. SQL Injection Introduction, from <http://www.controllingtheinter.net/forums/viewtopic.php?f=45&t=8>.
1606. Classification of SQL Injection Attacks, from http://courses.ece.ubc.ca/412/term_project/reports/2007-fall/Classification_of_SQL_Injection_Attacks.pdf.
1607. CHAPTER 1: INTRODUCTION, from <http://isea.nitk.ac.in/publications/web.pdf>.
1608. Dmitry Evteev, (2009), Advanced SQL Injection, from <http://www.ptsecurity.com/download/PT-devteev-Advanced-SQL-Injection-ENG.zip>.
1609. Krzysztof Kotowicz, (2010), SQL Injection: Complete walkthrough (not only) for PHP developers, from <http://www.slideshare.net/kkotowicz/sql-injection-complete-walkthrough-not-only-for-php-developers>.
1610. Nick Merritt, SQL Injection Attacks, from <http://www.evilsq.com/main/page1.php>.
1611. SQL Injection Cheat Sheet, from <http://michaeldaw.org/sql-injection-cheat-sheet>.
1612. Sagar Joshi, 2005, SQL Injection Attack and Defence, Available from <http://www.securitydocs.com/library/3587>.
1613. Kevin Spett, Blind SQL Injection-Are your web applications vulnerable?, from http://www.net-security.org/dl/articles/Blind_SQLInjection.pdf.
1614. Cameron Hotchkies, (2004), Blind SQL Injection Automation Techniques from <http://www.blackhat.com/presentations/bh-usa-04/bh-us-04-hotchkies/bh-us-04-hotchkies.pdf>.
1615. San-Tsai Sun, Ting Han Wei, Stephen Liu, and Sheung Lau, Classification of SQL Injection Attacks, from http://courses.ece.ubc.ca/412/term_project/reports/2007-fall/Classification_of_SQL_Injection_Attacks.pdf.
1616. SQL Injection, from <http://msdn.microsoft.com/en-us/library/ms161953.aspx>.
1617. SQL INJECTION, from <http://www.authorstream.com/Presentation/useful-155975-sql-injection-hacking-computers-22237-education-ppt-powerpoint/>.
1618. SQL Injection Cheat Sheet, from <http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/#UnionInjections>.
1619. SQL Injection, from http://hakipedia.com/index.php/SQL_Injection.

1620. K. K. Mookhey and Nilesh Burghate, (2004), Detection of SQL Injection and Cross-site Scripting Attacks, from <http://www.symantec.com/connect/articles/detection-sql-injection-and-cross-site-scripting-attacks>.
1621. Ferruh Mavituna, Deep Blind SQL Injection, from http://docs.google.com/viewer?a=v&q=cache:uvo9RU3T0v8J:labs.portcullis.co.uk/download/Deep_Blind_SQL_Injection.pdf+deep+blind+sql+injection&hl=en&gl=in&pid=bl&srcid=ADGEESgQ9uwIz-eJFM-k3zGP5NJQmHXmfP7UIW0SNTBI0hAV9h2pUWlvibONoFpu0MndYi_3Y-K0xT9sizjU2ljvSzVi4w4Xc_rDMKVFNszpj60kXYsYwUQ48OEW9IV-0ijwWFwYnOJW&sig=AHIetbT-5rxBYONib02-gShdro-oVnzYeA
1622. Debasish Das, Utpal Sharma, and D.K. Bhattacharyya, (2009), An Approach to Detection of SQL Injection Attack Based on Dynamic Query Matching from <http://www.ijcaonline.org/journal/number25/pxc387766.pdf>.
1623. (2010), Quick Security Reference: SQL Injection from <http://download.microsoft.com/download/E/E/7/EE7B9CF4-6A59-4832-8EDE-B018175F4610/Quick%20Security%20Reference%20-%20SQL%20Injection.docx>.
1624. Ferruh Mavituna, One Click Ownage, Adventures of a lazy pentester, from http://www.owasp.org/images/8/8e/One_Click_Ownage-Ferruh_Mavituna.pdf.
1625. Alexander Kornbrust, (2009), ODTUG - SQL Injection Crash Course for Oracle Developers from http://www.red-database-security.com/wp/OWW2009_sql_crashcourse_for_developers.pdf.
1626. Pankaj Sharma,(2005), SQL Injection Techniques & Countermeasures from <http://www.cert-in.org.in/Downloader?pageid=7&type=2&fileName=ciwp-2005-06.pdf>.
1627. Pankaj Sharma, (2005), SQL Injection Techniques & Countermeasures, from http://docs.google.com/viewer?a=v&q=cache:OKkPI9h11R8J:www.cert-in.org.in/knowledgebase/whitepapers/ciwp-2005-06.pdf+sql+injection+countermeasures&hl=en&gl=in&pid=bl&srcid=ADGEESjfo76m-84G_nbZgOQax9yl5HEXkM9ZSyLN-a0_kJfDD4v4PLvO41fByd3YJk3GcTKoczBFU2WiNWNMK13Tc28CJ4WcO-2bHxqldlWzR0GGmHSRmT39qkpqM2yhKpmfkQNCE80g&sig=AHIetbR4WivS8bCzwK13BkKQXXHlepLWqw.
1628. William G.J. Halfond, Jeremy Viegas, & Alessandro Orso, (2006), A Classification of SQL Injection Attack Techniques and Countermeasures, from <http://www.cc.gatech.edu/~orso/papers/halfond.viegas.orso.ISSSE06.presentation.pdf>.
1629. Code Injection, Available from http://www.owasp.org/index.php/Code_Injection.
1630. Understanding SQL Injection, Available from http://www.cisco.com/web/about/security/intelligence/sql_injection.html.
1631. VIVEK KUMBHAR, (2009), From Mind To Words, <http://blogs.msdn.com/vivekkum/default.aspx>.
1632. Reviewing Code for SQL Injection, Available from http://www.owasp.org/index.php/Reviewing_Code_for_SQL_Injection.
1633. Cross Site Scripting - OWASP, from www.owasp.org/index.php/Cross_Site_Scripting.
1634. Injection Flaws - OWASP, from www.owasp.org/index.php/Injection_Flaws.
1635. Application Security Guidelines on Kavi Community, from http://community.kavi.com/developers/security_standards/.
1636. J. Howard Beales, III, (2003), OWASP Web Application Security Top Ten List, from www.owasp.org/images/c/ce/OWASP_Top_Ten_2004.doc.
1637. Web Attacks - Cookie poisoning, from www.lodoga.co.uk/attackinfo/thethreat/examples/cook.htm.
1638. Victor Chapela,(2005), Advanced SQL Injection, from http://www.owasp.org/images/7/74/Advanced_SQL_Injection.ppt.

1639. Chema Alonso, (2008), RFD (Remote File Downloading) using Blind SQL Injection Techniques, from http://www.toorcon.org/tcx/16_Alonso.pdf.
1640. [PPT] AFITC 2001, from www.whitehatsec.com/presentations/AFITC_2001/afitc_2001.ppt.
1641. What is parameter tampering?, from www.imperva.com/application_defense_center/glossary/parameter_tampering.html.
1642. D.E. Chadbourne, Post office break in..., from <http://olduvai.blu.org/pipermail/discuss/2004-January/043138.htm>.
1643. Blind SQL Injection, from <http://www.securitydocs.com/library/2651>.
1644. Jrubner, (2006), 'SQL injection' attacks on the rise in Atlanta, from <http://www.bizjournals.com/atlanta/stories/2006/06/12/story8.html>.
1645. BSQLHacker, from <http://labs.portcullis.co.uk/application/bsql-hacker/>.
1646. Marathon Tool, from <http://marathontool.codeplex.com>.
1647. SQL Power Injector, from <http://www.sqlpowerinjector.com/download.htm>.
1648. Havij, from <http://www.itsecteam.com>.
1649. SQL Brute, from <http://www.gdssecurity.com/l/t.php>.
1650. BobCat, from <http://www.northern-monkee.co.uk/pub/bobcat.html>.
1651. Sqlninja, from <http://sqlninja.sourceforge.net/download.html>.
1652. sqlget, from <http://www.darknet.org.uk/2007/07/sqlget-v100-blind-sql-injection-tool-in-perl/>.
1653. Absinthe, from <http://www.darknet.org.uk/2006/07/absinthe-blind-sql-injection-toolssoftware/>.
1654. Blind Sql Injection Brute Forcer, from <http://code.google.com/p/bsqlbf-v2/>.
1655. sqlmap, from <http://sqlmap.org/>.
1656. SQL Injection Digger, from <http://sqid.rubyforge.org>.
1657. Pangolin, from <http://nosec.org/en/evaluate/>.
1658. SQLPAT, from <http://www.cquare.net/wp/tools/password-recovery/sqlpat/>.
1659. FJ-Injector Framework, from <http://sourceforge.net/projects/injection-fwk/>.
1660. Exploiter (beta), from http://www.ibm.com/developerworks/rational/downloads/08/appscan_exploiter/.
1661. SQLIer, from <http://bcable.net/project.php?sqlier>.
1662. sqlsus, from <http://sqlsus.sourceforge.net>.
1663. SQLEXEC() Function, from [http://msdn.microsoft.com/en-us/library/1x933c7s\(v=vs.80\).aspx](http://msdn.microsoft.com/en-us/library/1x933c7s(v=vs.80).aspx).
1664. SqlInjector, from http://www.woanware.co.uk/?page_id=19.
1665. Automagic SQL Injector, from <http://www.securiteam.com/tools/6P00LOAEKQ.html>.
1666. SQL Inject-Me, from <http://labs.securitycompass.com/exploit-me/sql-inject-me/>.
1667. NTO SQL Invader, from <http://www.ntobjectives.com/go/nto-sql-invader-free-download/>.
1668. The Mole, from <http://themole.nasel.com.ar/?q=downloads>.
1669. Microsoft Source Code Analyzer, from <http://www.microsoft.com/en-us/download/details.aspx?id=16305>.
1670. Microsoft UrlScan Filter, from <http://www.microsoft.com/en-in/download/details.aspx?id=5728>.
1671. dotDefender, from <http://www.aplicure.com/download-latest>.
1672. IBM Security AppScan, from <http://www.ibm.com/developerworks/downloads/r/appscan/>.
1673. WebCruiser, from <http://sec4app.com/>.

1674. HP WebInspect, from <http://www.hpenterprisesecurity.com/products/hp-fortify-software-security-center/hp-webinspect>.
1675. SQLDict, from <http://ntsecurity.nu/toolbox/sqldict/>.
1676. HP Scrawlr, from <https://h30406.www3.hp.com/campaigns/2008/wwcampaign/1-57C4K/index.php>.
1677. SQL Block Monitor, from <http://sql-tools.net/blockmonitor/>.
1678. Acunetix Web Vulnerability Scanner, from <http://www.acunetix.com/vulnerability-scanner/>.
1679. GreenSQL Database Security, from <http://www.greensql.com/content/greensql-database-security#&slider1=1>.
1680. Microsoft Code Analysis Tool .NET (CAT.NET), from <http://www.microsoft.com/en-us/download/details.aspx?id=5570>.
1681. NGS SQuirreL Vulnerability Scanners, from <http://www.nccgroup.com/en/our-services/security-testing-audit-compliance/information-security-software/ngs-squirrel-vulnerability-scanners/>.
1682. WSSA - Web Site Security Scanning Service, from <http://www.beyondsecurity.com/sql-injection.html>.
1683. N-Stalker Web Application Security Scanner, from <http://www.nstalker.com/products/editions/free/>.

Module 15: Hacking Wireless Networks

1684. The ABCs of IEEE 802.11, from <http://home.comcast.net/~timgroth/abc.htm>.
1685. Wi-Fi Hotspot Networks Sprout Like Mushrooms, from <http://spectrum.ieee.org/telecom/wireless/wifi-hotspot-networks-sprout-like-mushrooms/abc>.
1686. A list of wireless network attacks, from <http://searchsecurity.techtarget.com/feature/A-list-of-wireless-network-attacks>.
1687. Rogue Access Point Setups on Corporate Networks, from <http://www.infosecurity-magazine.com/view/10516/comment-rogue-access-point-setups-on-corporate-networks-/>.
1688. Advanced SQL Injection, from <http://blog.pages.kr/1341>.
1689. Identifying Rogue Access Points, from <http://www.wi-fiplanet.com/tutorials/article.php/1564431>.
1690. Bluetooth Security Risks and Tips to Prevent Security Threats, from <http://www.brighthub.com/computing/smb-security/articles/30045.aspx>.
1691. Cisco Unified Wireless Network Architecture—Base Security Features, from http://www.cisco.com/en/US/docs/solutions/Enterprise/Mobility/emob41dg/ch4_Secu.html.
1692. Tutorial: My Plate is Compatible?, from http://www.aircrack-ng.org/doku.php?id=pt-br:compatible_cards.
1693. Wireless Networking Security, from <http://technet.microsoft.com/en-us/library/bb457019.aspx>.
1694. Path Traversal and URIs, from <http://phucjimy.wordpress.com/category/document-security/>.
1695. How to Cheat at Securing a Wireless Network, from <http://www.sciencedirect.com/science/article/pii/B9781597490870500572>.
1696. Eliminating interference thru Wi-Fi spectrum analysis, from <http://searchmobilecomputing.techtarget.com/tip/Eliminating-interference-thru-Wi-Fi-spectrum-analysis>.
1697. How to Surf Safely on Public Wi-Fi, from <http://technology.inc.com/2007/07/01/how-to-surf-safely-on-public-wi-fi/>.
1698. Understanding WiFi Hotspots... from <http://www.scambusters.org/wifi.html>.
1699. WLAN Glossary, from <http://www.lever.co.uk/wlan-glossary.html>.

1700. Basic Service Set Identity (BSSID), from <http://www.interwifi.co.uk/glossary/b/basic-service-set-identity.html>.
1701. DSSS - Direct Sequence Spread Spectrum, from <http://www.telecomabc.com/d/dsss.html>.
1702. Frequency-hopping spread spectrum, from https://www.princeton.edu/~achaney/tmve/wiki100k/docs/Frequency-hopping_spread_spectrum.html
1703. 802.11x Modules, Dev Kits Can Help Simplify Wireless Design Efforts, from <http://www.digikey.com/us/en/techzone/wireless/resources/articles/802-11x-modules-dev-kits.html>.
1704. Antennas, from <http://82.157.70.109/mirrorbooks/wireless/0321202171/ch03lev1sec3.html>.
1705. How 802.11 Wireless Works, from <http://technet.microsoft.com/en-us/library/cc757419%28v=WS.10%29.aspx>.
1706. TKIP (Temporal Key Integrity Protocol), from <http://www.tech-faq.com/tpkip-temporal-key-integrity-protocol.html>.
1707. WPA2, from <http://www.wi-fi.org/knowledge-center/glossary/wpa2%E2%84%A2>.
1708. Cisco Unified Wireless Network Architecture—Base Security Features, from http://www.cisco.com/en/US/docs/solutions/Enterprise/Mobility/emob41dg/ch4_Secu.html.
1709. Understanding WEP Weaknesses, from <http://www.dummies.com/how-to/content/understanding-wep-weaknesses.html>.
1710. V.802.11 Wireless LAN, from <http://www.apl.jhu.edu/~hhsu/cs771/cs771-II.pdf>.
1711. 7 Things Hackers Hope You Don't Know, from <http://www.esecurityplanet.com/views/article.php/3891716/7-Things-Hackers-Hope-You-Dont-Know.htm>.
1712. Rogue Wireless Access Point, from <http://www.tech-faq.com/rogue-wireless-access-point.html>.
1713. How to Surf Safely on Public Wi-Fi, from <http://technology.inc.com/2007/07/01/how-to-surf-safely-on-public-wi-fi>.
1714. Wireless Network Security Tools, from <http://www.wirelessnetworktools.com/index.html>.
1715. How to War Drive, from <http://www.wikihow.com/War-Drive>.
1716. Tools for analyzing WLAN traffic abound, from http://www.computerworld.com.au/article/273427/tools_analyzing_wlan_traffic_abound/.
1717. Tutorial: Is My Wireless Card Compatible? from http://www.aircrack-ng.org/doku.php?id=compatible_cards.
1718. MITM Attack, from <https://wilder.hq.sk/OpenWeekend-2005/foil14.html>.
1719. Security Threats of Smart Phones and Bluetooth, from http://www.aaronfrench.com/coursefiles/ucommerce/Loo_2009.pdf.
1720. Tips for using Bluetooth Securely, from <http://www.brighthub.com/computing/smb-security/articles/30045.aspx>.
1721. Carrumba, (2009), How to Crack WPA/WPA2, from <http://www.megapanzer.com/2009/10/02/how-to-crack-wpawpa2/>.
1722. Introduction to Wireless Network, Available from http://media.wiley.com/product_data/excerpt/02/07645973/0764597302.pdf.
1723. Prabhaker Mateti , Hacking Techniques in Wireless Networks, Available from <http://www.cs.wright.edu/~pmateti/InternetSecurity/Lectures/WirelessHacks/Mateti-WirelessHacks.htm>.
1724. Cisco Unified Wireless Network Architecture—Base Security Features, from http://www.cisco.com/en/US/docs/solutions/Enterprise/Mobility/emob41dg/ch4_Secu.html.

1725. (2008), WLAN Security Threats, from http://enterasys.com/company/literature/WLAN%20Security%20Today-Siemens%20whitepaper_EN.pdf.
1726. Hack Wireless WPA Network, Available from <http://mediakey.dk/~cc/hack-wireless-wpa-network/>
1727. Hack Wireless WEP Network, Available from <http://mediakey.dk/~cc/hack-wireless-network-crack/>
1728. Brian Kuebler, Wireless Wrongs; Hacking WiFi, Available from <http://www.abc2news.com/news/local/story/Wireless-Wrongs-Hacking-WiFi/0anuc7U-kOaxjPfcjw0uw.cspx>.
1729. Tony Northrup , 10 tips for improving your wireless network, Available from <http://www.microsoft.com/athome/setup/wirelesstips.aspx>.
1730. Bradley Mitchell, Wired vs. Wireless Networking, Available from <http://compnetworking.about.com/cs/homenetworking/a/homewiredless.htm>.
1731. Wireless Network Image, Available from http://www.hvitsolutions.com/main/images/building_diagram_wireless.jpg.
1732. Wireless networking standards, Available from http://reviews.cnet.com/4520-7605_7-6871493-2.html.
1733. What are the types of Wireless Networks?, Available from <http://www.tech-faq.com/wireless-networks.shtml>.
1734. Different Types of Wireless Network, Available from <http://www.greyfriars.net/gcg/greyweb.nsf/miam/article01>.
1735. Hacking Techniques in Wireless Networks, Available from <http://www.mundowifi.com.br/forum/thread1181.html>.
1736. Evan McKinney, Disadvantages of Wireless Networks, Available from http://www.ehow.com/facts_4809373_disadvantages-wireless-networks.html.
1737. Bradley Mitchell, Wireless Standards - 802.11b 802.11a 802.11g and 802.11n, Available from <http://compnetworking.about.com/cs/wireless80211/a/aa80211standard.htm>.
1738. Wireless Network Devices, Available from <http://www.pcrush.com/category/65/Wireless-Network-Devices>.
1739. WEP (wired equivalent privacy), Available from <http://www.networkworld.com/details/715.html>.
1740. Wi-Fi Protected Access, Available from http://searchmobilecomputing.techtarget.com/sDefinition/0,,sid40_gci887323,00.html.
1741. WPA (Wi-Fi Protected Access), Available from <http://www.tech-faq.com/wpa-wi-fi-protected-access.shtml>.
1742. Paul Arana, (2006), Benefits and Vulnerabilities of Wi-Fi Protected Access 2 (WPA2), Available from http://cs.gmu.edu/~yhwang1/INFS612/Sample_Projects/Fall_06_GPN_6_Final_Report.pdf.
1743. The Wi-Fi Protected Access 2 (WPA2)/Wireless Provisioning Services Information Element (WPS IE) update for Windows XP with Service Pack 2 is available, Available from <http://support.microsoft.com/kb/893357>.
1744. TKIP (Temporal Key Integrity Protocol), Available from <http://www.tech-faq.com/tkip-temporal-key-integrity-protocol.shtml>.
1745. Renaud Deraison,, (2009), Using Nessus to Detect Wireless Access Points, Available from <http://www.nessus.org/whitepapers/wap-id-nessus.pdf>.
1746. WLAN Networking / 802.11, Available from <http://www.wardrive.net/>.
1747. Cracking WEP and WPA Wireless Networks, Available from http://docs.lucidinteractive.ca/index.php/Cracking_WEP_and_WPA_Wireless_Networks.
1748. Cracking WEP using Backtrack, Available from <http://ryanunderdown.com/linux/cracking-wep-using-backtrack.php>.

1749. Wireless Hacking, Available from <http://www.darknet.org.uk/category/wireless-hacking/>.
1750. WiFi Wireless Hacking, Available from <http://www.hackerscatalog.com/Products/CD-ROMS/WiFiHacking.html>.
1751. Bradley Morgan, (2006), Wireless Cracking Tools, Available from <http://www.windowsecurity.com/whitepapers/Wireless-Cracking-Tools.html>.
1752. What Are Rogue Access Points?, Available from <http://www.manageengine.com/products/wifi-manager/rogue-access-point.html>.
1753. Rick Doten, Wireless Security and Wireless Security Monitoring, Available from <http://www.issae.org/documents/ISSARogueAPpresentationBoston.ppt>.
1754. Gary Wollenhaupt, How Cell Phone Jammers work, Available from <http://electronics.howstuffworks.com/cell-phone-jammer1.htm>.
1755. Brian R. Miller & Booz Allen Hamilton, 2002, Issues in Wireless security, Available from <http://www.acsac.org/2002/case/wed-c-330-Miller.pdf>.
1756. Justin Montgomery, How WPA wireless networks are hacked, and how to protect yourself, Available from <http://tech.blorge.com/Structure:%20/2009/02/07/how-wpa-wireless-networks-are-hacked-and-how-to-protect-yourself/>.
1757. Jonathan Hassell, (2004), Wireless Attacks and Penetration Testing, Available from <http://www.securityfocus.com/infocus/1783>.
1758. Robert J. Shimonski, (2003), Wireless Attacks Primer, Available from http://www.windowsecurity.com/articles/Wireless_Attacks_Primer.html.
1759. Wireless Network Attack Methodology, Available from <http://www.wirelessnetworktools.com/>.
1760. Martin Beck & TU-Dresden, (2008), Practical attacks against WEP and WPA, Available from <http://dl.aircrack-ng.org/breakingwepandwpa.pdf>.
1761. Simple Steps To Basic Wireless Hacking, Available from <http://mixeduperic.com/Windows/Hacks/simple-steps-to-basic-wireless-hacking.html>.
1762. LE Webmaster , (2005), Wireless Scanning Wardriving / Warchalking, Available from <http://www.linuxexposed.com/content/view/42/52/>.
1763. Finding cloaked access points, (Chapter 9) , Available from http://books.google.com/books?id=wGJhDNspE3wC&pg=PA333&lpg=PA333&dq=cloaked+access+point&source=bl&ots=ZDkHSykDNV&sig=1sLKIx-1ZcqkhUdr1WpFaqYczyl&hl=en&ei=V8R2Ss35Oo2e6gP59viqCw&sa=X&oi=book_result&ct=result&resnum=3#v=onepage&q=cloaked%20access%20point&f=false.
1764. Wireless Scanning Wardriving / Warchalking, Available from <http://www.it-observer.com/wireless-scanning-wardriving-warchalking.html>.
1765. Zamzom Wireless Network Tool, Available from http://www.freewarehome.com/index.html?http%3A//www.freewarehome.com/Internet/Networking/Network_Monitoring_t.html.
1766. 5 – Wireless Network, Available from <http://www.hackingtheuniverse.com/information-security/attack-vs-defense/attack-vs-defense-on-an-organizational-scale/5-wireless-network>.
1767. Hacking the Invisible Network, Available from <http://www.net-security.org/dl/articles/Wireless.pdf>.
1768. Michael Roche, Wireless Attack Tools, Available from http://www.cse.wustl.edu/~jain/cse571-07/ftp/wireless_hacking.pdf.
1769. Joshua Wright, Detecting Wireless LAN MAC Address Spoofing, Available from <http://forskningsnett.uninett.no/wlan/download/wlan-mac-spoof.pdf>.
1770. How to Break WEP Encryption, Available from http://www.ehow.com/how_2209766_break-wep-encryption.html.

1771. Daniel V. Hoffman, Essential Wireless Hacking Tools, Available from <http://www.ethicalhacker.net/content/view/16/24/>.
1772. Protecting your wireless network from hacking, Available from http://www.businessknowledgesource.com/technology/protecting_your_wireless_network_from_hacking_025027.html.
1773. Eric Janszen, (2002), Understanding Basic WLAN Security Issues, from <http://www.wi-fiplanet.com/tutorials/article.php/953561>.
1774. RTX NEWS JANUARY 2003 NO.1, from www.rtx.dk/Admin/Public/DWSDownload.aspx?File=Files%2FFiler%2Fannouncements%2Fnewsletter%2F4_SCREEN.pdf.
1775. Agustina, J.V. Peng Zhang, and Kantola, (2003), Performance evaluation of GSM handover traffic in a GPRS/GSM network, from http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?isnumber=27298&arnumber=1214113&count=217&index=21.
1776. Service set identifier, from http://searchmobilecomputing.techtarget.com/sDefinition/0,,sid40_gci853455,00.html.
1777. Antenna Cabling Guide, from <http://wireless.gumph.org/content/3/12/011-antenna-cabling.html>.
1778. Wireless Security Auditor (WSA), from <http://www.research.ibm.com/gsal/wsa/>.
1779. NAI's Sniffer Wireless to Support 802.11a and 802.11b Networks, from http://www.findarticles.com/p/articles/mi_zd4168/is_200202/ai_n9515340.
1780. RADIUS Protocol Security and Best Practices, from <http://www.microsoft.com/technet/prodtechnol/windows2000serv/maintain/security/radiussec.mspx>.
1781. Wi-Fi Security, from <http://main.wi-fi.org/OpenSection/secure.asp?TID=2#Radius>.
1782. WarChalking (Screen Shots), from <http://www.blackbeltjones.com>.
1783. Eavesdropping Detection Audits, from <http://www.spybusters.com>.
1784. Anton T. Rager, (2001), WEPCrack, AirSnort, from <http://wepcrack.sourceforge.net/>.
1785. Wireless Communication Policy - Rensselaer Help Desk, from <http://helpdesk.rpi.edu/update.do?artcenterkey=545>.
1786. Responsibility for Wireless Access Points, from <http://cals.arizona.edu/calsnet/security/ua-wireless-guidelines.htm>.
1787. Telephone tapping or wire tapping, from http://en.wikipedia.org/wiki/Telephone_tapping.
1788. Connie J. Sadler, NetBIOS NULL Sessions, from <http://www.brown.edu/Facilities/CIS/CIRT/help/netbiosnull.html>.
1789. The Hacker's Choice, from <http://freeworld.thc.org/releases.php?o=1&s=4%20-%2017k.\>.
1790. DOS_in_Wireless_Routing_Proocols_Hamilton, from <http://www.eng.auburn.edu/users/hamilton/security/SE2/>.
1791. Network Security Library, http://www.windowsecurity.com/whitepapers/windows_security/The_Unofficial_NT_Hack_FAQ/The_Uncofficial_NT_Hack_FAQ__Section_05.html.
1792. Oren Chapo, (1999), Network Management Protocols, from <http://www.chapo.co.il/articles/snmp/>.
1793. DaAnZeR, (2004), End to End Security for Windows 2000 Server, <http://www.securitydocs.com/library/2647>.
1794. Pascal Etienne, (2001), Weekly Security Tools Digest, from <http://boran.linuxsecurity.com/security/sp/toolsdigest/2001/tools20010426.html>.

1795. DumpAcl dumps NTs permissions and audit settings, from <http://www.windowsnetworking.com/kbase/WindowsTips/WindowsNT/AdminTips/Miscellaneous/DumpAclNTsPermissionsAndAuditSettings.html>.
1796. Windows 2000 Server, from [http://technet.microsoft.com/hi-in/windowsserver/2000/default\(en-us\).aspx](http://technet.microsoft.com/hi-in/windowsserver/2000/default(en-us).aspx).
1797. Definitions & Terms, from <http://www.bytepile.com/definitions-d.php>.
1798. VPN & Internet Security Solutions, from <http://www.solucom.com/define.htm>.
1799. Stephen M. Specht & Ruby B.Lee, Distributed Denial of Service: Taxonomies of Attacks, Tools and Countermeasures, from <http://palms.ee.princeton.edu/PALMSopen/DDoS%20Final%20PDCS%20Paper.pdf>.
1800. Craig A. Huegen, 2005, Denial of Service Attacks: "Smurfing", from <http://www.windowsecurity.com/whitepapers/Denial-of-Service-Attacks-Smurfing.html>.
1801. Wireless LAN Security 802.11b and Corporate Networks, from http://documents.iss.net/whitepapers/wireless_LAN_security.pdf.
1802. The Wireless Intrusion detection system, from http://www.forum-intrusion.com/widz_design.pdf.
1803. Wireless LAN Security, 802.11/Wi-Fi Wardriving & Warchalking, from <http://www.wardrive.net/>.
1804. Jim Geier, (2003), Identifying Rogue Access Points, from <http://www.wi-fiplanet.com/tutorials/article.php/1564431>.
1805. Lisa Phifer, Service set identifier, http://searchmobilecomputing.techtarget.com/sDefinition/0,,sid40_gci853455,00.html.
1806. WIRELESS COMMUNICATION POLICY, from http://www.longwood.edu/vpac/final_policy_base/6000/6124.htm.
1807. Kevin D. Murray, (2006), Security Scrapbook, from <http://www.spybusters.com/SS0402.html>.
1808. post office break in..., from <http://olduvai.blu.org/pipermail/discuss/2004-January/043138.html>.
1809. Venky, (2006), Wireless LAN Security, from http://www.iss.net/documents/whitepapers/wireless_LAN_security.pdf.
1810. WAVEMON, from <http://www.janmorgenstern.de/projects-software.html>.
1811. Patrik Karlsson, (2002), WaveStumbler, 802.11 Network Mapper, from <http://www.securiteam.com/tools/5GP002K6BM.html>.
1812. Egsander, (2006), WIRELESS DATA CONNECTIVITY GUIDELINE, from <http://cals.arizona.edu/calsnet/security/ua-wireless-guidelines.htm>.
1813. NPS Information Technology Policy/Standard, from <https://www.nps.navy.mil/ITACS/New05/ITPolicy/NPSITPolicy202.pdf>.
1814. Simple Active Attack Against TCP, from <https://db.usenix.org/>.
1815. Humphrey Cheung, (2005), How To Crack WEP - Part 1: Setup & Network Recon, from <http://www.tomsguide.com/us/how-to-crack-wep,review-451.html>.
1816. Humphrey Cheung, (2005), How To Crack WEP - Part 2: Performing the Crack, from <http://www.tomsguide.com/us/how-to-crack-wep,review-459.html>.
1817. Humphrey Cheung, (2005), How To Crack WEP - Part 3: Securing your WLAN, from <http://www.tomsguide.com/us/how-to-crack-wep,review-471.html>.
1818. Advantages and Disadvantages of WLANs, from <http://www.wireless-center.net/Wi-Fi-Security/Advantages-and-Disadvantages-of-WLANS.html>.
1819. Advantages vs. Disadvantages of WiFi, from <http://mason.gmu.edu/~fkondolo/page3>.

1820. Alrady, How to Use WIFI Hotspots with Security, from http://www.ehow.com/how_5287862_use-wifi-hotspots-security.html.
1821. James Kendrick, (2010), Smartphone Wi-Fi Usage on the Rise, from <http://jkontherun.com/2010/08/24/smartphone-wi-fi-usage-on-the-rise/>.
1822. Chris Weber and Gary Bahadur, Wireless Networking Security, from <http://technet.microsoft.com/en-us/library/bb457019.aspx>.
1823. Barb Bowman (2003), WPA Wireless Security for Home Networks, from http://www.microsoft.com/windowsxp/using/networking/expert/bowman_03july28.mspx.
1824. (2005), Wi-Fi Adoption, from http://www.businessweek.com/technology/tech_stats/wifi051003.htm.
1825. Christopher Elliott, 6 wireless threats to your business, from <http://www.microsoft.com/business/en-us/resources/technology/broadband-mobility/6-wireless-threats-to-your-business.aspx#wirelessthreatstoyourbusiness>.
1826. Warchalking Symbols, from http://www.hackerskitchen.com/mac-old/wifi/War_Chalk/.
1827. The RSN Protocol Process, from <http://www.tech-faq.com/rsn-robust-secure-network.html>.
1828. (2003), WEP encryption Process, from <http://technet.microsoft.com/en-us/library/cc757419%28WS.10%29.aspx>.
1829. (2006), WPA2: Second Generation WiFi Security, from <http://pcquest.ciol.com/content/technology/2006/106050803.asp>.
1830. The Four-Way Handshake, from <http://www.answers.com/topic/ieee-802-11i-2004>.
1831. Brandon Teska, (2008), How To Crack WPA / WPA2, from <http://www.smallnetbuilder.com/wireless/wireless-howto/30278-how-to-crack-wpa--wpa2>.
1832. Eric Geier, (2010), 7 Things Hackers Hope You Don't Know, from <http://www.esecurityplanet.com/views/article.php/3891716/7-Things-Hackers-Hope-You-Dont-Know.htm>.
1833. "Wireless LAN SecurityChecklist", from <http://www.wardrive.net/>.
1834. 802.11 Security Tools, from <http://www.wardrive.net/security/tools>.
1835. Wireless Security Tools, from http://www.corecom.com/html/wlan_tools.html.
1836. Lisa Phifer, (2010), Top Ten Free Wi-Fi Security Test Tools, from <http://www.esecurityplanet.com/views/article.php/3881181/Top-Ten-Free-Wi-Fi-Security-Test-Tools.htm>.
1837. Free Wireless Security Tools, from <http://netsecurity.about.com/cs/hackertools/a/aafreewifi.htm>.
1838. Bryan, (2005), Cracking WEP and WPA Wireless Networks, from http://docs.lucidinteractive.ca/index.php/Cracking_WEP_and_WPA_Wireless_Networks#WPA_Crackin.
1839. (2006), A Comprehensive Review of 802.11 Wireless LAN Security and the Cisco Wireless Security Suite, from http://www.cisco.com/warp/public/cc/pd/witc/ao1200ap/prodlist/wswpf_wp.htm#wp39475.
1840. (2006), How To Crack WEP and WPA Wireless Networks, from <http://121space.com/index.php?showtopic=3376>.
1841. Cisco Unified Wireless Network Architecture—Base Security Features, from http://www.cisco.com/en/US/docs/solutions/Enterprise/Mobility/emob41dg/ch4_Secu.html#wp1018984.
1842. Wireless DoS, from <http://www.cisco.com/en/US/docs/wireless/technology/wips/deployment/guide/wipsdep.html#wp150481>.
1843. (2009), How to prevent wireless DoS attacks, from http://searchsecurity.techtarget.com/generic/0,295582,sid14_gci1173628_mem1,00.html.

1844. Jim Geier, (2003), Denial of Service a Big WLAN Issue, from <http://www.esecurityplanet.com/trends/article.php/2200071/Denial-of-Service-a-Big-WLAN-Issue.htm>.
1845. Jonathan Hassell, (2004), Wireless Attacks and Penetration Testing (part 1 of 3), from <http://www.symantec.com/connect/articles/wireless-attacks-and-penetration-testing-part-1-3>.
1846. (2009), A list of wireless network attacks, from http://searchsecurity.techtarget.com/generic/0,295582,sid14_gci1167611_mem1,00.html.
1847. Lisa Phifer, (2009), A wireless network vulnerability assessment checklist, from http://searchsecurity.techtarget.com/generic/0,295582,sid14_gci1167666_mem1,00.html.
1848. Lisa Phifer, (2009), Hunting for rogue wireless devices, from http://searchsecurity.techtarget.com/generic/0,295582,sid14_gci1167664_mem1,00.html.
1849. PreciousJohnDoe, List of Wireless Network Attacks, from <http://www.brighthub.com/computing/smb-security/articles/53949.aspx>.
1850. Security Disciplines for Objective 3: Detection and Recovery, from <http://www.it.ojp.gov/documents/asp/wireless/section3-3-1.htm>.
1851. (2010), Wireless Sniffer, from <http://www.personaltelco.net/WirelessSniffer>.
1852. How to: Sniff Wireless Packets with Wireshark, from http://www.wirelessnets.com/resources/tutorials/sniff_packets_wireshark.html.
1853. WifiEagle Single- and Dual-Band 802.11 Channel Analyzers, from <http://www.nutsaboutnets.com/performance-wifi/products/product-wifieagle-wifi-channel-analyzer.htm>.
1854. Creating A Cheap Bluetooth Sniffer, from <http://thewifihack.com/blog/?p=27>.
1855. WLAN Analyzer and Protocol Decoder - CommView for WiFi, from <http://www.tamos.com/htmlhelp/commwifi/packets.htm>.
1856. Jim Geier, (2002), Understanding 802.11 Frame Types, from <http://www.wi-fiplanet.com/tutorials/article.php/1447501/Understanding-80211-Frame-Types.htm>.
1857. Laurent Oudot, (2004), Wireless Honeypot Countermeasures, from <http://www.symantec.com/connect/articles/wireless-honeypot-countermeasures>.
1858. (2009), Fragmentation Attack, from <http://www.aircrack-ng.org/doku.php?id=fragmentation>.
1859. Andrei A. Mikhailovsky, Konstantin V. Gavrilenco, and Andrew Vladimirov, (2004), The Frame of Deception: Wireless Man-in-the-Middle Attacks and Rogue Access Points Deployment, from <http://www.informit.com/articles/article.aspx?p=353735&seqNum=7>.
1860. Comment: Rogue Access Point Setups on Corporate Networks, from <http://www.infosecurity-us.com/view/10516/comment-rogue-access-point-setups-on-corporate-networks-/>.
1861. Kevin Beaver and Peter T. Davis, Understanding WEP Weaknesses, from <http://www.dummies.com/how-to/content/understanding-wep-weaknesses.html>.
1862. (2007), Cracking WEP Using Backtrack: A Beginner's Guide, from <http://ryanunderdown.com/linux/cracking-wep-using-backtrack.php>.
1863. (2009), FakelIKEd – Fake IKE Daemon Tool for MITM, from <http://www.darknet.org.uk/2009/08/fakeiked-fake-ike-daemon-tool-for-mitm/>.
1864. Renee Oricchio, How to Surf Safely on Public Wi-Fi, from <http://technology.inc.com/telecom/articles/200707/WiFi.html>.
1865. Aircrack-ng for Windows - Aircrack, from http://www.wirelessdefence.org/Contents/Aircrack-ng_WinAircrack.htm.
1866. (2010), Crack WEP key via connected client, from <http://carpeblunte.com/>.

1867. Using Cain and the AirPcap USB adapter to crack WPA/WPA2, from <http://www.irongeek.com/i.php?page=videos/airpcap-cain-wpa-cracking>.
1868. WiFi Hopper, from <http://wifihopper.com/overview.html>.
1869. PhoneSnoop: Spying on Blackberry Users, from <http://www.symantec.com/connect/blogs/phonesnoop-spionage-blackberry-users>.
1870. What is BlueJacking, from <http://www.newmobilemedia.com/bluejacking-2.htm>.
1871. , from <http://www.oxid.it/cain.html>.
1872. KisMAC, from <http://kismac-ng.org/>.
1873. inSSIDer, from <http://www.metageek.net/products/inssider/>.
1874. NetSurveyor, from <http://www.performancewifi.net/performance-wifi/products/netsurveyor-network-discovery.htm>.
1875. Vistumbler, from <http://www.vistumbler.net/>.
1876. WirelessMon, from <http://www.passmark.com/products/wirelessmonitor.htm>.
1877. WiFi Hopper, from <http://www.wifihopper.com/download.html>.
1878. Wavestumbler, from <http://www.cquare.net/wp/tools/other/wavestumbler/>.
1879. iStumbler, from <http://www.istumbler.net/>.
1880. WiFinder, from http://www.pgmsoft.com/apps/wifinder_for_android/.
1881. Meraki WiFi Stumbler, from <http://www.meraki.com/products/wireless/wifi-stumbler>.
1882. Wellenreiter, from <http://wellenreiter.sourceforge.net/>.
1883. AirCheck Wi-Fi Tester, from <http://www.flukenetworks.com/enterprise-network/network-testing/AirCheck-Wi-Fi-Tester>.
1884. AirRadar 2, from <http://www.koingosw.com/products/airradar.php>.
1885. Xirrus Wi-Fi Inspector, from <http://www.xirrus.com/Products/Wi-Fi-Inspector>.
1886. Wifi Analyzer, from <http://a.farproc.com/wifi-analyzer>.
1887. WiFiFoFum - WiFi Scanner, from <http://www.wififofum.net/downloads>.
1888. Network Signal Info, from http://www.kaubits-software.com/product_networksignal.htm.
1889. WiFi Manager, from <http://kmansoft.com/>.
1890. OpenSignalMaps, from <http://opensignal.com/>.
1891. WIGLE, from <http://wigle.net/gps/gps/main/download/>.
1892. Skyhook, from <http://www.skyhookwireless.com/location-technology/sdk.php>.
1893. jiWire, from <http://v4.jewire.com/search-hotspot-locations.htm>.
1894. WeFi, from <http://www.wefi.com/download/>.
1895. Wireshark, from <http://www.wireshark.org/download.html>.
1896. Cascade Pilot, from <http://www.riverbed.com/products-solutions/products/performance-management/network-infrastructure/High-Speed-Packet-Analysis.html>.
1897. OmniPeek, from http://www.wildpackets.com/products/omnipeek_network_analyzer.
1898. Sniffer Portable Professional Analyzer, from http://www.netscout.com/products/enterprise/Sniffer_Portable_Analyzer/Sniffer_Portable_Professional_Analyzer/Pages/default.aspx.
1899. Capsa WiFi, from http://www.colasoft.com/download/products/capsa_free.php.
1900. ApSniff, from <http://www.monolith81.de/apsniff.html>.

1901. NetworkMiner, from <http://www.netresec.com/?page=NetworkMiner>.
1902. Airscanner Mobile Sniffer, from <http://www.airscanner.com/products/sniffer/>.
1903. Observer, from <http://www.networkinstruments.com/products/observer/index.php?tab=download>.
1904. WifiScanner, from <http://wifiscanner.sourceforge.net/>.
1905. Mognet, from <http://www.monolith81.de/mognet.html>.
1906. Iperf, from <http://iperf.sourceforge.net/>.
1907. Aircrack-ng, from <http://www.aircrack-ng.org/>.
1908. SMAC, from <http://www.klcconsulting.net/smac/>.
1909. KisMAC, from <http://kismac-ng.org/>.
1910. Elcomsoft Wireless Security Auditor, from <http://www.elcomsoft.com/ewsa.html>.
1911. WepAttack, from <http://wepattack.sourceforge.net/>.
1912. Wesside-ng, from <http://www.aircrack-ng.org/doku.php?id=wesside-ng>.
1913. WEPCrack, from <http://wepcrack.sourceforge.net/>.
1914. WepDecrypt, from <http://wepdecrypt.sourceforge.net/>.
1915. Portable Penetrator, from <http://www.secpoint.com/portable-penetrator.html>.
1916. CloudCracker, from <https://www.cloudcracker.com/>.
1917. coWPAtty, from <http://wirelessdefence.org/Contents/coWPAttyMain.htm>.
1918. Wifite, from <https://code.google.com/p/wifite/downloads/detail?name=wifite-2.0r85.tar.gz>.
1919. WepOff, from <http://www.ptsecurity.ru/download/wepoff.tar.gz>.
1920. ApSniff, from <http://www.monolith81.de/apsniff.html>.
1921. WiFiFoFum, from <http://www.aspecto-software.com/rw/applications/wififofum/>.
1922. WarLinux, from <http://sourceforge.net/projects/warlinux/>.
1923. MacStumbler, from <http://www.macstumbler.com/>.
1924. WiFi-Where, from <http://www.threejacks.com/?q=node/13>.
1925. AirFart, from <http://airfart.sourceforge.net/>.
1926. AirTraf, from <http://airtraf.sourceforge.net/>.
1927. 802.11 Network Discovery Tools, from <http://wavelan-tools.sourceforge.net/>.
1928. NetworkManager, from <http://projects.gnome.org/NetworkManager/>.
1929. KWiFiManager, from <http://kwifimanager.sourceforge.net/>.
1930. NetworkControl, from <http://www.arachnoid.com/NetworkControl/index.html>.
1931. KOrinoco, from <http://korinoco.sourceforge.net/>.
1932. Sentry Edge II, from <http://www.tek.com/document/news-release/tektronix-advances-rf-monitoring-sentry-edge-ii>.
1933. WaveNode, from <http://www.wavenode.com/>.
1934. xosview, from <http://xosview.sourceforge.net/>.
1935. RF Monitor, from <http://www.newsteo.com/gb/data-logger/features/monitoring-software.php>.
1936. DTC-340 RFXpert, from <http://www.dektec.com/products/Apps/DTC-340/index.asp>.
1937. Home Curfew RF Monitoring System, from http://solutions.3m.com/wps/portal/3M/en_US/ElectronicMonitoring/Home/ProductsServices/OurProducts/HomeCurfewRFMonitoringSystem/.

1938. RFProtect Spectrum Analyzer, from <http://www.arubanetworks.com/products/arubaos/rfprotect-spectrum-analyzer/>.
1939. AirMagnet WiFi Analyzer, from <http://www.flukenetworks.com/enterprise-network/wireless-network/AirMagnet-WiFi-Analyzer>.
1940. OptiView® XG Network Analysis Tablet, from <http://www.flukenetworks.com/enterprise-network/network-monitoring/optiview-xg-network-analysis-tablet>.
1941. Network Traffic Monitor & Analyzer CAPSA, from <http://www.javvin.com/packet-traffic.html>.
1942. Observer, from <http://www.networkinstruments.com/products/observer/index.php?tab=download>.
1943. Ufasoft Snif, from <http://ufasoft.com/sniffer/>.
1944. vxSniffer, from <http://www.cambridgevx.com/vxsniffer.html>.
1945. OneTouch™ AT Network Assistant, from <http://www.flukenetworks.com/enterprise-network/network-testing/OneTouch-AT-Network-Assistant>.
1946. SoftPerfect Network Protocol Analyzer, from <http://www.softperfect.com/products/networksniffer/>.
1947. WirelessNetView, from http://www.nirsoft.net/utils/wireless_network_view.html.
1948. Airview, from <http://airview.sourceforge.net>.
1949. RawCap, from <http://www.netresec.com/?page=RawCap>.
1950. Cisco Spectrum Expert, from <http://www.cisco.com/en/US/products/ps9393/index.html>.
1951. AirMedic® USB, from <http://www.flukenetworks.com/enterprise-network/wireless-network/AirMedic>.
1952. AirSleuth-Pro, from <http://nutsaboutnets.com/airsleuth-spectrum-analyzer/>.
1953. BumbleBee-LX Handheld Spectrum Analyzer, from <http://www.bvsystems.com/Products/Spectrum/BumbleBee-LX/bumblebee-lx.htm>.
1954. Wi-Spy, from <http://www.metageek.net/products/wi-spy/>.
1955. Super Bluetooth Hack, from <http://gallery.mobile9.com/f/317828/>.
1956. BTBrowser, from <http://wireless.klings.org/BTBrowser/>.
1957. BH Bluejack, from <http://croozeus.com/blogs/?p=33>.
1958. Bluediving, from <http://bluediving.sourceforge.net/>.
1959. Blooover, from http://trifinite.org/trifinite_stuff_blooover.html.
1960. BTScanner, from http://www.pentest.co.uk/downloads.html?cat=downloads§ion=01_bluetooth.
1961. CIHwBT, from <http://sourceforge.net/projects/cih-with-bt/files/>.
1962. BT Audit, from http://trifinite.org/trifinite_stuff_btaudit.html.
1963. BlueAlert, from http://www.insecure.in/bluetooth_hacking_02.asp.
1964. AirMagnet WiFi Analyzer, from <http://www.flukenetworks.com/enterprise-network/wireless-network/AirMagnet-WiFi-Analyzer>.
1965. AirDefense, from <http://www.airdefense.net/products/servicesplatform/index.php>.
1966. Adaptive Wireless IPS, from <http://www.cisco.com/en/US/products/ps9817/index.html>.
1967. Aruba RFProtect WIPS, from <http://www.arubanetworks.com/products/arubaos/rfprotect-wireless-intrusion-protection>.
1968. Enterasys® Intrusion Prevention System, from <http://www.enterasys.com/products/advanced-security-apps/dragon-intrusion-detection-protection.aspx>.
1969. RFProtect Wireless Intrusion Protection, from <http://www.arubanetworks.com/products/arubaos/rfprotect-wireless-intrusion-protection>.
1970. SonicWALL Wireless Networking, from <http://o-www.sonicwall.com/us/en/solutions/4224.html>.

1971. HP TippingPoint IPS, from [http://h17007.www1.hp.com/us/en/products/network-security/HP_S_Intrusion_Prevention_System_\(IPS\)_Series/index.aspx](http://h17007.www1.hp.com/us/en/products/network-security/HP_S_Intrusion_Prevention_System_(IPS)_Series/index.aspx).
1972. AirTight WIPS, from <http://www.airtightnetworks.com/home/products/AirTight-WIPS.html>.
1973. Network Box IDP, from <http://www.network-box.co.uk/technology/threatmanagement/IDP>.
1974. AirMobile Server, from http://www.aimobile.se/airmobile_server.htm.
1975. WLS Manager, from http://www.airpatrolcorp.com/products/wls_manager.php.
1976. Wireless Policy Manager (WPM), from <http://airpatrolcorp.com/airpatrol-products/wpmwec/>.
1977. ZENworks® Endpoint Security Management, from <http://www.novell.com/products/zenworks/endpointsecuritymanagement/features/>.
1978. AirMagnet Planner, from <http://www.flukenetworks.com/enterprise-network/wireless-network/AirMagnet-Planner>.
1979. Cisco Prime Infrastructure, from <http://www.cisco.com/en/US/products/ps12239/index.html>.
1980. AirTight Planner, from <http://www.airtightnetworks.com/home/products/AirTight-Planner.html>.
1981. LANPlanner, from http://www.motorola.com/Business/US-EN/Business+Product+and+Services/Software+and+Applications/WLAN+Management+and+Security+Software/LANPlanner_US-EN.
1982. RingMaster, from <http://www.juniper.net/us/en/products-services/software/network-management-software/ringmaster/>.
1983. Connect EZ Predictive RF CAD Design, from http://www.connect802.com/suite_spot.htm#.
1984. Ekahau Site Survey (ESS), from <http://www.ekahau.com/products/ekahau-site-survey/overview.html>.
1985. ZonePlanner, from <http://www.ruckuswireless.com/products/zoneplanner>.
1986. Wi-Fi Planning Tool, from <http://www.aerohive.com/planner>.
1987. TamoGraph Site Survey, from <http://www.tamos.com/products/wifi-site-survey/wlan-planner.php>.
1988. OSWA, from <http://securitystartshere.org/page-downloads.htm>.
1989. WiFiZoo, from <http://community.corest.com/~hochoa/wifizoo/index.html#download>.
1990. Network Security Toolkit, from <http://networksecuritytoolkit.org/nst/index.html>.
1991. Nmap Community Edition, from <http://www.rapid7.com/products/nmap/compare-downloads.jsp>.
1992. WiFish Finder, from <http://www.airtightnetworks.com/home/resources/knowledge-center/wifish-finder.html>.
1993. Penetrator Vulnerability Scanning Appliance, from <http://www.secpoint.com/penetrator.html>.
1994. SILICA, from <http://www.immunityinc.com/downloads.shtml>.
1995. Wireless Network Vulnerability Assessment, from <http://www.secnap.com/products/audits/wireless-assessment.html>.
1996. Karma, from <http://www.theta44.org/karma/>.
1997. Hotspotter, from <http://www.wirelessdefence.org/Contents/hotspotter.htm>.
1998. Airsnarf, from <http://airsnarf.shmoo.com/>.
1999. Asleap, from <http://www.willhackforsushi.com/Asleap.html>.
2000. THC-LEAP Cracker, from <http://wirelessdefence.org/Contents/THC-LEAPcracker.htm>.
2001. Airsnort, from <http://airsnort.shmoo.com/>.
2002. Void11, from <http://www.wirelessdefence.org/Contents/Void11Main.htm>.
2003. Technitium MAC Address Changer (TMAC), from <http://www.technitium.com/tmac/index.html>.

Module 16 : Hacking Mobile Platforms

2004. Delivering enterprise information securely on Android and Apple iOS devices, from http://www.citrix.com/site/resources/dynamic/additional/iPad_Technical_Guide_US_WP.pdf.
2005. Understanding the Security Changes in Windows Phone 8, from <http://www.mobilejaw.com/articles/2012/08/understanding-the-security-changes-in-windows-phone-8/>.
2006. HOW TO HACK YOUR ANDROID PHONE, from <http://www.mobilenyou.in/2010/10/hack-your-android-phone.html>.
2007. Windows Phone 8, from http://en.wikipedia.org/wiki/Windows_Phone_8.
2008. Delivering corporate data securely on employee iPads, from http://resources.idgenterprise.com/original/AST-0043716_iPad_Technical_Guide_US_WP_2_.pdf.
2009. Working guide to Root Android Phones Easy with SuperOneClick, from <http://fixlife.in/23/working-guide-root-android-devices-phones-easy-way-superoneclick>.
2010. How to Hack Your Android Phone (and Why You Should Bother), from http://readwrite.com/2010/01/27/how_to_hack_your_android_phone.
2011. New Android Trojan Masquerades as Google Library, Taps Device Administration API, from http://www.netqin.com/en/security/newsinfo_4595_2.html%20.
2012. Security Alert: New SMS Android Trojan -- DroidLive -- Being Disguised as a Google Library, from <http://www.csc.ncsu.edu/faculty/jiang/DroidLive/>.
2013. SuperOneClick, from <http://shortfuse.org/>.
2014. Superboot, from <http://www.modaco.com/topic/348161-superboot-galaxy-nexus-root-solution/>.
2015. Unrevoked, from <http://unrevoked.com/recovery/>.
2016. Universal Androot, from <http://android.org.in/2012/08/universal-androot-root-android-in-5-sec/>.
2017. Unlock Root, from <http://www.unlockroot.com/products.html>.
2018. DroidSheep, from <http://droidsheep.de>.
2019. FaceNiff, from <http://faceniff.ponury.net>.
2020. Google Apps Device Policy, from <https://play.google.com>.
2021. DroidSheep Guard, from <http://droidsheep.de>.
2022. X-Ray, from <http://www.xray.io>.
2023. Android Network Toolkit - Anti, from <http://www.zantiapp.com>.
2024. Find My Phone, from <http://findmyphone.mangobird.com>.
2025. Prey Anti-Theft, from <http://preyproject.com>.
2026. Android Anti Theft Security, from <http://www.snuko.com>.
2027. Wheres My Droid, from <http://wheresmydroid.com>.
2028. iHound, from <https://www.ihoundssoftware.com>.
2029. GadgetTrak Mobile Security, from <http://www.gadgettrak.com>.
2030. Total Equipment Protection App, from <https://protection.sprint.com>.
2031. AndroidLost.com, from <http://www.androidlost.com>.
2032. Redsn0w, from <http://blog.iphone-dev.org>.
2033. Absinthe, from <http://greenpois0n.com>.
2034. Sn0wbreeze , from <http://www.idownloadblog.com/download/>.
2035. PwnageTool, from <http://blog.iphone-dev.org>.

2036. LimeRa1n, from <http://www.limera1n.com>.
2037. Jailbreakme, from <http://www.jailbreakme.com>.
2038. Blackra1n, from <http://blackra1n.com>.
2039. Spirit, from <http://spiritjb.com>.
2040. Find My iPhone , from <https://itunes.apple.com>.
2041. iHound, from <https://www.ihoundssoftware.com>.
2042. GadgetTrak iOS Security, from <http://www.gadgettrak.com>.
2043. iLocalis, from <http://ilocalis.com>.
2044. MaaS360 Mobile Device Management (MDM), from <http://www.maas360.com>.
2045. Citrix XenMobile MDM, from <http://www.zenprise.com>.
2046. Absolute Manage MDM, from <http://www.absolute.com>.
2047. SAP Afaria , from <http://www.sybase.com>.
2048. Device Management Centre, from <http://www.sicap.com>.
2049. AirWatch, from <http://www.air-watch.com>.
2050. Good Mobile Manager , from <http://www1.good.com>.
2051. MobileIron, from <http://www.mobileiron.com>.
2052. Rule Mobility , from <http://www.tangoe.com>.
2053. TARMAC, from <http://www.tarmac-mdm.com>.
2054. MediaContact, from <http://www.device-management-software.com>.
2055. BullGuard Mobile Security, from <http://www.bullguard.com>.
2056. Lookout, from <https://www.lookout.com>.
2057. WISeID, from <http://www.wiseid.mobi>.
2058. McAfee Mobile Security, from <https://www.mcafeemobilesecurity.com>.
2059. AVG AntiVirus Pro for Android, from <http://www.avg.com>.
2060. avast! Mobile Security , from <http://www.avast.com>.
2061. Norton Mobile Security, from <http://us.norton.com>.
2062. ESET Mobile Security, from <http://www.eset.com>.
2063. Kaspersky Mobile Security, from <http://www.kaspersky.com>.
2064. F-Secure Mobile Security , from <http://www.f-secure.com>.
2065. Trend Micro™ Mobile Security , from <http://www.trendmicro.com>.
2066. Webroot Secure Anywhere Mobile, from <http://www.webroot.com>.
2067. NetQin Mobile Security, from <http://en.nq.com/mobilesecurity/download>.
2068. AnDOSid, from http://apps.opera.com/en_us/andosid.html.
2069. ComDroid, from <http://www.comdroid.org/>.
2070. Woodpecker, from <http://www.firmhouse.com/>.
2071. iPhoneSimFree, from <http://www.iphonesimfree.com/>.
2072. anySIM, from <https://code.google.com/p/devteam-anySIM/downloads/list>.
2073. Metasploit, from <http://www.metasploit.com/>.
2074. Cain & Abel , from <http://www.oxid.it/cain.html>.
2075. WindowBreak, from <http://windowsphonehacker.com/windowbreak/>.

2076. BBProxy, from http://www.symantec.com/security_response/writeup.jsp?docid=2006-081416-4756-99.
2077. Elcomsoft Phone Password Breaker , from <http://www.elcomsoft.com/eppb.html>.

Module 17: Evading IDS, Firewalls, and HoneyPots

2078. Intrusion Detection System (IDS) Evasion, from http://complianceandprivacy.com/WhitePapers/iDefense-IDS-Evasion/iDefense_IDSEvasion_20060510.pdf.
2079. Evading NIDS, from http://www.bandwidthco.com/sf_whitepapers/penetration/Evading%20NIDS%20Revisited.pdf.
2080. Intrusion detection system evasion techniques, from http://en.wikipedia.org/wiki/Intrusion_detection_system_evasion_techniques.
2081. How to bypass a firewall, from <http://www.bit.uni-bonn.de/Wob/images/49692243.pdf>.
2082. Wired and wireless intrusion detection system: Classifications, good characteristics and state-of-the-art, from <http://www.sciencedirect.com/science/article/pii/S092054890500098X>.
2083. Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection, from http://www.windowsecurity.com/whitepapers/intrusion_detection/Insertion_Evasion_and_Denial_of_Service_Eluding_Network_Intrusion_Detection_.html.
2084. SmartDefense, from <http://www.sciencedirect.com/science/article/pii/B9781597492454000076>.
2085. How to configure Internet Explorer to use a proxy server, from <http://support.microsoft.com/kb/135982>.
2086. Defeating Sniffers and Intrusion Detection Systems, from <http://www.phrack.org/issues.html?issue=54&id=10>.
2087. Techniques used for bypassing firewall systems, from <http://www.terena.org/activities/tfc-sirt/meeting9/gowdiak-bypassing-firewalls.pdf>.
2088. Firewalking, from <http://www.webopedia.com/TERM/F/firewalking.html>.
2089. IT Infrastructure Security Plan, from <http://www.sciencedirect.com/science/article/pii/B9781597490887500098>.
2090. What is a firewall? from <http://kb.iu.edu/data/aoru.html>.
2091. Functionalities of Firewalls, from <http://www.cs.ucsb.edu/~koc/ns/projects/04Reports/He.pdf>.
2092. Updating snort with a customized controller to thwart port scanning, from http://www.aloul.net/Papers/faloul_scn10.pdf.
2093. Firewalls, from http://www.techrepublic.com/i/tr/downloads/home/0072260815_chapter_9.pdf.
2094. Firewalking, from <http://www.webopedia.com/TERM/F/firewalking.html>.
2095. What is HoneyPot? from <http://www.securityhunk.com/2010/06/what-is-honeypot.html>.
2096. Honeypots - Definitions and Value of Honeypots , from <http://infosecwriters.com/texts.php?op=display&id=80>.
2097. How to Set Up a Honey Pot, from http://www.ehow.com/how_5245821_set-up-honey-pot.html.
2098. Snort 2.8.5.2 : Intrusion Detection Tool, from http://ashwintumma.files.wordpress.com/2010/11/is_snort.pdf.
2099. Writing Snort Rules, from http://paginas.fe.up.pt/~mgi98020/pgr/writing_snort_rules.htm.
2100. Insertion, Evasion, and Denial of Service:Eluding Network Intrusion Detection, from <http://www.creangel.com/papers/Eluding%20Network%20Intrusion%20Detection.pdf>.
2101. Intrusion detection system evasion techniques, from http://en.wikipedia.org/w/index.php?title=Intrusion_detection_system_evasion_techniques&oldid=311670246.

2102. Evading NIDS, revisited, from <http://www.symantec.com/connect/articles/evading-nids-revisited>.
2103. How To Access Blocked / Bypass Blocked Websites, from <http://wwwcomputingunleashed.com/how-to-access-blocked.html>.
2104. How do I use a Proxy Server? from <http://whatismyipaddress.com/using-proxies>.
2105. How to configure Internet Explorer to use a proxy server, from <http://support.microsoft.com/kb/135982>.
2106. Firewall Fairytales, from http://www.iqmtm.com/PDF_presentations/IQ_Firewall_Fairytales_June2010-1.pdf.
2107. Intrusion detection systems IDS, from <http://www.geeksgate.com/blog/812.html>.
2108. Defending Against Network IDS Evasion, from <http://www.raid-symposium.org/raid99/PAPERS/Paxson.pdf>.
2109. Sumit Siddharth, (2005), Evading NIDS, revisited, Available from <http://www.securityfocus.com/infocus/1852>.
2110. Alexis Lawrence, How to Set Up a Honey Pot, Available from http://www.ehow.com/how_5245821_set-up-honey-pot.html.
2111. GHH, What is GHH?, Available from <http://ghh.sourceforge.net/>.
2112. Phrack Magazine Volume Seven, Issue Forty-Nine File 06 of ..., from <http://www.phrack.org/phrack/49/P49-06>.
2113. Kasey Efaw, Installing Snort 2.8.5.2 on Windows 7, http://www.snort.org/assets/135/Installing_Snort_2.8.5.2_on_Windows_7.pdf.
2114. (2006), Intrusion Detection System (IDS) Evasion, http://complianceandprivacy.com/WhitePapers/iDefense-IDS-Evasion/iDefense_IDSEvasion_20060510.pdf.
2115. Brian Caswell, Writing Snort Rules A quick guide, <http://www.shmoo.com/~bmc/presentations/2004/honeynet/caswell-writing-snort-rules.ppt>.
2116. Unblock Blocked Websites like Myspace, Bebo and Orkut, from <http://www.clazh.com/unblock-blocked-websites-like-myspace-bebo-and-orkut/>.
2117. Firewalls, from <http://hacker-dox.net/Que-Certified.Ethical.Hacker.E/0789735318/ch10lev1sec5.html>.
2118. Firewall Basics, <http://www.unixgeeks.org/security/newbie/security/firewall.html>.
2119. Honeypots, from <http://www.infosecwriters.com/texts.php?op=display&id=80>.
2120. Dale Farris, (2005), Honeypots for Windows, from <http://www.gtpcc.org/gtpcc/honeypotsforwindows.htm>.
2121. Mike Neuman, (1995), Bugtraq: ANNOUNCE: Freely available TTY monitoring/control program, from <http://seclists.org/bugtraq/1995/Jun/0049.html>.
2122. Web Application Attacks, [PDF] Guide, from www.netprotect.ch/downloads/webguide.pdf.
2123. David Endler & Michael Sutton, [PPT] iDEFENSE Labs, from www.blackhat.com/presentations/bh-usa-02/endler/bh-us-02-endler-brute.ppt.
2124. Intrusion detection, from http://www.networkworld.com/links/Downloads/Security/Intrusion_detection/.
2125. Tony Bradley, Free Intrusion Detection (IDS) and Prevention (IPS) Software, from <http://netsecurity.about.com/od/intrusiondetectionid1/a/aafreeids.htm>.
2126. AIDE, from <http://www.cryptomancer.de/programme/aide-en.html>.
2127. The Evolution of Intrusion Detection System, from www.securityfocus.com/infocus.
2128. Navy Information Assurance Website, from https://.../ps/?t=infosecprodsservices/infosecprodsservices.tag&bc=/infosecprodsservices/bc_ids.html.

2129. Firewalking, from <http://www.webopedia.com/TERM/F/firewalking.html>.
2130. Vinay, (2009), How to Bypass Firewalls Restrictions using Proxy Servers, from <http://www.ihackintosh.com/2009/03/how-to-bypass-firewalls-restrictions-using-proxy-servers/>.
2131. Adam Gowdiak, (2003), Firewall Attack Techniques, from <http://www.terena.org/activities/tf-csirt/meeting9/gowdiak-bypassing-firewalls.pdf>.
2132. How to bypass the firewall (Bypassing from external sources and MITM attacks), from www.b-it-center.de/Wob/images/81134082.ppt.
2133. Bypassing Firewalls, <http://flylib.com/books/en/3.500.1.95/1/>.
2134. Intrusion detection system - EnterpriseNetworkingPlanet, from http://networking.webopedia.com/TERM/I/intrusion_detection_system.html.
2135. An Introduction to IDS, from www.securityfocus.com/infocus/.
2136. Network security, from www.njcpu.net/security.htm.
2137. Hacking Through IDSs, from www.airscanner.com/pubs/ids.pdf.
2138. INTRUSION DETECTION -BISS Forums, from www.bluetack.co.uk/forums/index.php?showtopic.
2139. iSecurityShop, from www.isecurityshop.com/.
2140. Enterasys Dragon Host Sensor, from www.enterasys.com/products/ids/DSHSS-xxx/.
2141. MJohnson, Vanguard Security Solutions - Vanguard Integrity Professionals, from www.go2vanguard.com/software.
2142. Thomas H. Ptacek, Insertion, Evasion and Denial of Service: Eluding Network Intrusion Detection, from www.insecure.org/stf/secnet_ids/secnet_ids.pdf.
2143. Insertion, Evasion, and Denial of Service, from www.snort.org/docs/idspaper/.
2144. Evading Intrusion Detection, from www.tux.org/pub/tux/storm/ids-simple.doc.
2145. IDS: Re: Polymorphic Shellcode detection, from seclists.org/lists/focus-ids/2003/May/0019.
2146. Hardware Firewalls, from <http://cybercoyote.org/security/hardware.shtml>.
2147. Circuit-Level Gateway, from www.softheap.com/internet/circuit-level-gateway.html.
2148. Vicomsoft Firewall Q&A, from www.vicomsoft.com/knowledge/reference/firewalls1.html.
2149. Statoo.htm: some simple stalking tools, from www.searchlores.org/statoo.htm.
2150. van Hauser, Placing Backdoors Through Firewalls, from www.thc.segfault.net/download.php?t=p&f=fw-backd.htm.
2151. J. Christian Smith, (2000), Introduction, www.gray-world.net/papers/covertshells.txt.
2152. BSD Search.Com - Applications: Networking, from www.bsdsearch.com/dir//applications/networking.php.
2153. Peter Kieltyka, ICMP Shell, from icmpshell.sourceforge.net/.
2154. Measuring Security Threats with Honeypot, from www.honeynet.org/papers/individual/sane-2004.
2155. Lance Spitzner, (2003), Open Source Honeypots: Learning with Honeyd, from www.securityfocus.com/infocus/1659.
2156. Honeypot Software, Honeypot Products, Deception Software, from www.honeypots.net/honeypots/products.
2157. Measuring Security Threats with Honeypot Technology, from www.honeynet.org/papers/individual/sane-2004.
2158. Know Your Enemy: Sebek, from www.honeynet.org/papers/sebek.pdf.

2159. Intrusion Detection System (IDS) Evasion, from http://complianceandprivacy.com/WhitePapers/iDefense-IDS-Evasion/iDefense_IDSEvasion_20060510.pdf.
2160. Intrusion Detection/Prevention, from http://www.protectpoint.com/services_intrusiondetection.htm.
2161. Intrusion Detection Systems, from <http://www.cra.org/Activities/craw/dmp/awards/2003/Tan/research/rules.html>.
2162. Managed Security Services- Intrusion Detection Programs, from http://www.gus.net/Intrusion_Detection.htm.
2163. Nick DeClario, Keep Out : Host Intrusion Detection, from <http://www.linuxsecurity.com/content/view/112852/151/>.
2164. Yona Hollander, The Future of Web Server Security, from http://www.mcafee.com/us/local_content/white_papers/wp_future.pdf.
2165. Unauthorized remote users can read IIS files, from <http://xforce.iss.net/xforce/xfdb/2663>.
2166. Jan Meijer, Multiple Microsoft IIS Vulnerabilities, from <http://cert-nl.surfnet.nl/s/2000/S-00-35.htm>.
2167. Automated Web Interface Scans IIS for Multiple Vulnerabilities, from <http://www.ciac.org/ciac/bulletins/k-068.shtml>.
2168. Mark Burnett, (2001), Running Snort on IIS Web Servers Part 2: Advanced Techniques, from <http://www.securityfocus.com/infocus/1316>.
2169. Spanish Honeypot Project, from <http://www.honeynet.org.es/project/>.
2170. Lance Spitzner, 2003, Honeypots, from <http://www.ip97.com/tracking-hackers.com/papers/honeypots.html>.
2171. A Virtual Honeypot Framework, from <http://www.citi.umich.edu/techreports/reports/citi-tr-03-1.pdf>.
2172. What is SPECTER, from <http://www.specter.ch/introduction50.shtml>.
2173. Shaheem Motlekar, (2004), Honeypot FAQ, from <http://www.tracking-hackers.com/misc/faq.html>.
2174. Honeytokens: The Other Honeypot, from <http://www.securityfocus.com/infocus/1713>.
2175. van Hauser, Placing Backdoors Through Firewalls, from <http://www.securitymap.net/sdm/docs/attack/fw-backd.htm>.
2176. Placing Backdoors Through Firewalls, from http://www.windowsecurity.com/whitepapers/Placing_Backdoors_Through_Firewalls.html.
2177. Honeypots: Three new tools related to IDS, forensics, honeypots, from <http://seclists.org/honeypots/2003/q2/0279.html>.
2178. Network Intrusion Detection Using Snort, from <http://www.linuxsecurity.com/content/view/117497/49/>.
2179. Intrusion Detection, from http://www.ctssg.com/ids_p.htm.
2180. INTRUSION DETECTION, from <http://www.pafis.shh.fi/~tantit01/isac2002/ce03/password.html>.
2181. Paul Innella, 2001, An Introduction to IDS, from <http://www.securityfocus.com/infocus/1520>.
2182. Ricky M. Magalhaes, (2003), Host-Based IDS vs Network-Based IDS, from http://www.windowsecurity.com/articles/Hids_vs_Nids_Part2.html.
2183. Intrusion detection system, from http://www.webopedia.com/TERM/I/intrusion_detection_system.html.
2184. Paul Innella, 2001, The Evolution of Intrusion Detection Systems, from <http://www.securityfocus.com/infocus/1514>.
2185. Host, from <http://lists.debian.org/>.

2186. Spammers use Word files to bypass filters, from http://www.zdnet.com.au/news/security/soa/Spammers_use_Word_files_to_bypass_filters/0,130061744,139267487,00.htm.
2187. Know Your Enemy: Sebek, from <http://www.honeynet.org/papers/sebek.pdf>.
2188. Honeyd - Network Rhapsody for You, from <http://www.citi.umich.edu/u/provos/honeyd/>.
2189. SPECTER Introduction, from <http://www.specter.ch/introduction50.shtml>.
2190. Measuring Security Threats with Honeypot Technology, from <http://www.honeynet.org/papers/individual/sane-2004.pdf>.
2191. Lance Spitzner, (2003), SecurityFocus: Honeytokens -The Other Honeypot, from <http://www.securityfocus.com/infocus/1713>.
2192. Andrea Barisani, vol5issue6, from <http://www.tisc-insight.com/newsletters/56.html>.
2193. Peter Kieltyka, (2006), ICMP Shell, from <http://icmpshell.sourceforge.net/>.
2194. ntsecurity.nu - ack tunneling, from <http://ntsecurity.nu/papers/acktunneling/>.
2195. SecuriTeam™ - ACK Tunneling Trojans, from <http://www.securiteam.com/securityreviews/5OP0P156AE.html>.
2196. Placing Backdoors Through Firewalls, from <http://www.thc.segfault.net/papers/fw-backd.htm>.
2197. Mike, Firewalk, from www.blackhat.com/presentations/bh-usa-99/Route/bh-us-99-schiffman.ppt.
2198. Hardware Firewalls, from <http://cybercoyote.org/security/hardware.shtml>.
2199. Evading Intrusion Detection, from www.tux.org/pub/tux/storm/ids-simple.doc.
2200. Insertion, Evasion, and Denial of Service, from <http://www.snort.org/docs/idspaper/>.
2201. Securing IT Assets with Linux, from www.bass-inc.com/presentations/arp21_2004/linuxsecurity.ppt.
2202. Linux Security Quick Reference Guide, from http://www.tldp.org/REF/ls_quickref/QuickRefCard.pdf.
2203. Vanguard Security Solutions – Vanguard Integrity Professionals, from <http://www.go2vanguard.com/software/>.
2204. iSecurityShop, from <http://www.isecurityshop.com/>.
2205. Going on the Defensive: Intrusion Detection Systems, from <http://www.airscanner.com/pubs/ids.pdf>.
2206. Network Security, from <http://www.njcpu.net/security.htm>.
2207. Tipping Point, from <http://h10163.www1.hp.com>.
2208. Security Network Intrusion Prevention System, from <http://www-01.ibm.com>.
2209. Enterprise, from <http://www.tripwire.com/it-security-software/security-configuration-management/file-integrity-monitoring/>.
2210. Specter, from <http://www.specter.com/default50.htm>.
2211. Honeyd, from <http://www.honeyd.org/>.
2212. KFSensor, from <http://www.keyfocus.net/kfsensor/>.
2213. Symantec Decoy Server, from <http://www.symantec.com/press/2003/n030623b.html>.
2214. Tiny Honeypot, from <http://freecode.com/projects/thp>.
2215. LaBrea, from <http://labrea.sourceforge.net/labrea-info.html>.
2216. PatriotBox, from <http://www.alkasis.com/?action=products&pid=6>.
2217. Kojoney, from <http://kojoney.sourceforge.net/>.
2218. HoneyBOT, from <http://www.atomicsoftwaresolutions.com/honeybot.php>.
2219. Google Hack Honeypot, from <http://ghh.sourceforge.net/>.

2220. WinHoneyd, from <http://www2.netvigilance.com/winhoneyd>.
2221. HIHAT, from <http://hihat.sourceforge.net/>.
2222. Argos, from <http://www.few.vu.nl/argos/?page=2>.
2223. Glastopf, from <http://glastopf.org/>.
2224. Send-Safe Honeypot Hunter, from <http://www.send-safe.com/honeypot-hunter.html>.
2225. IBM Security Network Intrusion Prevention System, from <http://www-01.ibm.com/software/tivoli/products/security-network-intrusion-prevention/>.
2226. Peek & Spy, from <http://networkingdynamics.com/peek-spy/peekspx/>.
2227. INTOUCH INSA-Network Security Agent, from http://www.ttinet.com/doc/insa_v15_025.html.
2228. Strata Guard, from <http://www.stillsecure.com/strataguard>.
2229. IDP8200 Intrusion Detection and Prevention Appliances, from <https://www.juniper.net/in/en/products-services/security/idp-series/idp8200/>.
2230. OSSEC, from http://www.ossec.net/?page_id=19.
2231. Cisco Intrusion Prevention Systems, from http://www.cisco.com/en/US/products/ps5729/Products_Sub_Category_Home.html.
2232. AIDE (Advanced Intrusion Detection Environment), from <http://aide.sourceforge.net/>.
2233. SNARE (System iNtrusion Analysis & Reporting Environment), from <http://www.intersectalliance.com/>.
2234. Vanguard Enforcer, from <http://www.go2vanguard.com/enforcer.php>.
2235. Check Point Threat Prevention Appliance, from <http://www.checkpoint.com/products/threat-prevention-appliances/>.
2236. fragroute, from <http://www.monkey.org/~dugsong/fragroute/>.
2237. Next-Generation Intrusion Prevention System (NGIPS), from <http://www.sourcefire.com/security-technologies/network-security/next-generation-intrusion-prevention-system>.
2238. Outpost Network Security, from <http://www.agnitum.com/products/networksecurity/index.php>.
2239. Check Point IPS-1, from <http://www.checkpoint.com/products/ips-1/>.
2240. FortiGate, from <http://www.fortinet.com/solutions/ips.html>.
2241. Enterasys® Intrusion Prevention System, from <http://www.enterasys.com/products/advanced-security-apps/dragon-intrusion-detection-protection.aspx>.
2242. StoneGate Virtual IPS Appliance, from <http://www.stonesoft.com/en/products/appliances/virtual-ips.html>.
2243. Cyberoam Intrusion Prevention System, from <http://www.cyberoam.com/ips.html>.
2244. McAfee Host Intrusion Prevention for Desktops, from <http://www.mcafee.com/us/products/host-ips-for-desktop.aspx>.
2245. ZoneAlarm PRO Firewall, from <http://www.zonealarm.com/security/en-us/zonealarm-pro-firewall-anti-spyware.htm>.
2246. Check Point Firewall Software Blade, from <http://www.checkpoint.com/products/firewall-software-blade/index.html>.
2247. eScan Enterprise Edition, from http://www.escanav.com/english/content/products/corp_enterprise/escan_enterprise.asp.
2248. Jetico Personal Firewall, from <http://www.jetico.com/firewall-jetico-personal-firewall/>.
2249. Outpost Security Suite, from <http://free.agnitum.com/#>.
2250. Novell BorderManager, from <http://www.novell.com/products/bordermanager/>.

2251. Firewall UTM, from <http://www.esoft.com/network-security-appliances/instagate/>.
2252. Sonicwall, from <http://www.tribecaexpress.com/products/by-manufacturers/sonicwall-firewalls-price.htm>.
2253. Comodo Firewall, from <http://personalfirewall.comodo.com/>.
2254. Online Armor, from <http://www.online-armor.com/products-online-armor-free.php>.
2255. FortiGate-5101C, from http://www.fortinet.com/press_releases/120227.html.
2256. proxyfy, from <http://proxyfy.com>.
2257. spysurfing, from <http://www.spysurfing.com>.
2258. alienproxy, from <http://alienproxy.com>.
2259. zendproxy, from [http://zendproxy.com/](http://zendproxy.com).
2260. anonymous, from <http://anonymous.org>.
2261. anonymizer, from <http://www.anonymizer.com>.
2262. webproxyserver, from [http://www.webproxyserver.net/](http://www.webproxyserver.net).
2263. boomproxy, from [http://www.boomproxy.com/](http://www.boomproxy.com).
2264. Loki ICMP tunneling, from http://www.iss.net/security_center/reference/vuln/Loki.htm.
2265. AckCmd, from [http://ntsecurity.nu/toolbox/ackcmd/](http://ntsecurity.nu/toolbox/ackcmd).
2266. HTTPTunnel, from <http://www.nocrew.org/software/httptunnel.html>.
2267. Send-Safe Honeypot Hunter, from <http://www.send-safe.com/honeypot-hunter.html>.
2268. Traffic IQ Professional, from <http://www.idappcom.com/downloads.php>.
2269. tcp-over-dns, from <http://analogbit.com/software/tcp-over-dns>.
2270. Snare Agent for Windows, from <http://www.intersectalliance.com/projects/BackLogNT/>.
2271. AckCmd, from [http://ntsecurity.nu/toolbox/ackcmd/](http://ntsecurity.nu/toolbox/ackcmd).
2272. Tomahawk, from [http://tomahawk.sourceforge.net/](http://tomahawk.sourceforge.net).
2273. Your Freedom, from <http://www.your-freedom.net/index.php?id=downloads>.
2274. Atelier Web Firewall Tester, from [http://www.atelierweb.com/products/firewall-tester/](http://www.atelierweb.com/products/firewall-tester).
2275. Freenet, from [https://freenetproject.org/](https://freenetproject.org).
2276. GTunnel, from <http://gardennetworks.org/download>.
2277. Hotspot Shield, from <http://www.anchorfree.com/hotspot-shield-VPN-download-windows.php>.
2278. Proxifier, from [http://www.proxifier.com/](http://www.proxifier.com).
2279. Vpn One Click, from <http://www.vpnoneclick.com/download/index.html>.
2280. Multi-Generator (MGEN), from <http://cs.itd.nrl.navy.mil/work/mgen/index.php>.
2281. Net-Inspect, from <http://search.cpan.org/~sullr/Net-Inspect/lib/Net/Inspect/L3/IP.pm>.
2282. NConvert, from [http://www.xnview.com/en/nconvert/](http://www.xnview.com/en/nconvert).
2283. fping 3, from [http://fping.org/](http://fping.org).
2284. pktgen, from <http://www.linuxfoundation.org/collaborate/workgroups/networking/pktgen>.
2285. PacketMaker, from <http://www.jdsu.com/en-us/Test-and-Measurement/Products/a-z-product-list/Pages/packetmaker-sas-sata-tester.aspx>.

Module 18: Buffer Overflow

2286. Understanding Buffer Overruns, from <http://uk.sys-con.com/node/33998>.

2287. Exploits: Heap, from <http://www.sciencedirect.com/science/article/pii/B9781597499972500066>.
2288. Exploits: Stack, from <http://www.sciencedirect.com/science/article/pii/B9781597499972500054>.
2289. Writing Exploits II, from <http://www.sciencedirect.com/science/article/pii/B9781597499972500091>.
2290. Hacking Unix, from http://media.techtarget.com/searchEnterpriseLinux/downloads/Hacking_Exp_ch7.pdf.
2291. Testing for Stack Overflow, from https://www.owasp.org/index.php/Testing_for_Stack_Overflow.
2292. Heap Corruption, from <http://www.sciencedirect.com/science/article/pii/B9781932266672500463>.
2293. Buffer Overflow Attacks—Detect, Exploit, Prevent, from http://newark.pardey.org/deck/book/buffer_overflow_attacks.pdf.
2294. Hack Proofing Your Network-8, from <http://forum.slime.com.tw/thread117254.html>.
2295. Statically Detecting Likely Buffer Overflow Vulnerabilities, from <http://lclint.cs.virginia.edu/usenix01.html>.
2296. Buffer Overflow - OWASP, from www.owasp.org/index.php/Buffer_Overflow.
2297. NedSecure Solutions - Toelichting aanvalstechnieken, from www.nedsecure.nl/index.php?option=com_content&task=view&id=111&Itemid=44&lang=.
2298. Tony Bradley, Zero Day Exploits: The Holy Grail, from <http://netsecurity.about.com/od/newsandeditorial1/a/aazeroday.htm>.
2299. FWSM URL Filtering Solution TCP ACL Bypass Vulnerability [Products ...], from www.cisco.com/en/US/products/products_security_advisory09186a0080464d00.shtml.
2300. Roger Gustavsson, (2006), Buffer overflow, from <http://idenet.bth.se/servlet/download/news/23644/Gustavsson++Buffer+Overflows.pdf>.
2301. Stack Smashing Defense: A Buffer Overflow Lab Exercise, from http://cisa.umbc.edu/CDX/Will/stack_smash_proposal.pdf.
2302. US-CERT Vulnerability Note VU#726198, from <http://www.kb.cert.org/vuls/id/726198>.
2303. David Litchfield, Windows Heap Overflows, www.blackhat.com/presentations/win.../bh-win-04-litchfield.ppt.
2304. Ronnie Johndas, Steps Involved in Exploiting a Buffer Overflow Vulnerability using a SEH Handler, http://www.infosecwriters.com/text_resources/pdf/RJohndas_Buffer_Overflow_SEH_Handler.pdf.
2305. Microsoft Index Server ISAPI Extension Buffer Overflow, from <http://www.ciac.org/ciac/bulletins/I-098.shtml>.
2306. Mehdi Mousavi, What an ISAPI extension is?, from http://www.codeproject.com/KB/ISAPI/isapi_extensions.aspx.
2307. Fireproofing Against DoS Attacks, from <http://www.networkcomputing.com/1225/1225f38.html>.
2308. Unchecked Buffer in ISAPI Extension Enables Remote Compromise of IIS 5.0 Server, from <http://www.securiteam.com/windowsntfocus/5CP010K4AK.html>.
2309. Heap Overrun in HTR Chunked Encoding Could Enable Web Server Compromise, from <http://www.securiteam.com/windowsntfocus/5IP0C1P7FC.html>.
2310. Unchecked Buffer in Index Server ISAPI Extension Leads to Web Server Compromise, from <http://www.securiteam.com/windowsntfocus/5FP0B2K4KU.html>.
2311. Testing for Heap Overflow, from http://www.owasp.org/index.php/Testing_for_Heap_Overflow.
2312. Tom Chmielarski, (2010), Enhanced Mitigation Experience Toolkit reduces buffer overflow attacks, from http://searchmidmarketsecurity.techtarget.com/tip/0,289483,sid198_gci1520906,00.html.
2313. ADMmutate, from <http://www.ktwo.ca/security.html>.
2314. GDB, from <http://www.gnu.org/software/gdb/>.

2315. Netcat, from <http://netcat.sourceforge.net/download.php>.
2316. LCLint, from <http://www.linuxjournal.com/article/3599>.
2317. Code::Blocks, from <http://www.codeblocks.org/>.
2318. eEye Retina, from <http://www.eeye.com/>.
2319. Spike, from http://spike.lazypics.de/dl_index_en.html.
2320. Brute Force Binary Tester (BFB), from <http://bfbstester.sourceforge.net/>.
2321. Immunity CANVAS, from <http://www.immunityinc.com/products-canvas.shtml>.
2322. Immunity Debugger, from <http://www.immunityinc.com/products-immdbg.shtml>.
2323. Splint, from <http://www.splint.org/download.html>.
2324. Flawfinder, from <http://www.dwheeler.com/flawfinder/>.
2325. BLAST, from <http://mtc.epfl.ch/software-tools/blast/index-epfl.php>.
2326. Stack Shield, from <http://www.angelfire.com/sk/stackshield/download.html>.
2327. Valgrind, from <http://valgrind.org/downloads/current.html>.
2328. PolySpace C Verifier, from <http://www.mathworks.in/products/polyspace/>.
2329. Insure++, from <http://www.parasoft.com/jsp/products/insure.jsp?itemId=63>.
2330. /GS, from <http://microsoft.com>.
2331. BufferShield, from <http://www.sys-manage.com/PRODUCTS/BufferShield/tabid/61/Default.aspx>.
2332. DefenseWall, from <http://www.softsphere.com/online-help/defenceplus/>.
2333. TIED, from http://www.security.iitk.ac.in/index.php?page=contents/projects/tied_libsafe/tied_libsafeplus.
2334. LibsafePlus, from http://www.security.iitk.ac.in/index.php?page=contents/projects/tied_libsafe/tied_libsafeplus.
2335. Comodo Memory Firewall, from http://www.comodo.com/news/press_releases/16_01_08.html.
2336. Clang Static Analyzer, from <http://clang-analyzer.llvm.org/>.
2337. FireFuzzer, from <https://code.google.com/p/firefuzzer/>.
2338. BOON, from <http://www.cs.berkeley.edu/~daw/boon/>.
2339. The Enhanced Mitigation Experience Toolkit, from <http://www.microsoft.com/en-us/download/details.aspx?id=29851>.
2340. CodeSonar® Static Analysis Tool, from <http://www.grammatech.com/codesonar>.
2341. CORE IMPACT Pro, from <http://www.coresecurity.com/core-impact-pro>.

Module 19: Cryptography

2342. MD5 - message digest (fingerprint, checksum) , from <http://www.akadia.com/services/md5.html>.
2343. Web App Security, from <http://www.hackerscenter.com/archive/view.asp?id=25264>.
2344. Cryptography, from <http://www.crcnetbase.com/doi/abs/10.1201/9780203507872.ch6>.
2345. Integrated Technologies, from <http://www.crcnetbase.com/doi/abs/10.1201/9780203330708.ch8>.
2346. Cracking S/MIME encryption using idle CPU time, from <http://www.securiteam.com/tools/3J5PRQ0PPQ.html>.
2347. Check Point RealSecure Attack Signatures Glossary, from http://www.checkpoint.com/support/technical/documents/realsecure/Attack_Signatures.pdf.
2348. Mark J Cox, from <http://www.awe.com/mark/talks/apachecon2003us.html>.

2349. (2001), Announcing the ADVANCED ENCRYPTION STANDARD (AES),
<http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.
2350. MICHAEL WELSCHENBACH, Cryptography in C and C++, Second Edition,
apress.com/book/view/1590595025.
2351. Rolf Oppliger, (2005), Contemporary Cryptography, <http://www.free-ebook-download.net/technical-book/8574-contemporary-cryptography.html>.
2352. R. F. Churchhouse, (2001), Codes and ciphers (Julius Caesar, the Enigma and the internet),
http://ebookee.org/Codes-and-Ciphers-Julius-Caesar-the-Enigma-and-the-Internet_128588.html.
2353. John Talbot and Dominic Welsh, (2006), Complexity and Cryptography an introduction,
http://www.cambridge.org/gb/knowledge/item1172875/?site_locale=en_GB.
2354. BRUCE SCHNEIER, Applied Cryptography, Second Edition, <http://www.schneier.com/book-applied.html>.
2355. JAMES BAMFORD, (2002), Body of Secrets, <http://sandiego.indymedia.org/media/2007/02/125027.pdf>.
2356. T. W. Korner, (1998), Coding and Cryptography, <http://www.dpmms.cam.ac.uk/~twk/>.
2357. Kenneth W. Dam and Herbert S. Lin, (1996), Cryptography's Role In Securing The Information Society,
<http://www.comms.scitech.susx.ac.uk/fft/crypto/cryptorole.pdf>.
2358. Peter Gutmann, Cryptography and Data Security,
<http://www.comms.scitech.susx.ac.uk/fft/crypto/CryptoTutorial/part1.pdf>.
2359. Documentation and Encryption, from <http://www.linuxsecurity.com/content/view/17/70/>.
2360. Josh Ryder, Introduction to Encryption, from <http://www.developer.com/tech/article.php/630681>.
2361. Authentication Technologies, from
<http://www.techarch.state.ar.us/domains/security/resources/techlist.htm>.
2362. Jari Arkko, Vesa Torvinen, Aki Niemi, (2002), HTTP Authentication with EAP, from
<http://www.arkko.com/publications/draft-torvinen-http-eap-01.txt>.
2363. Ralf Junker, Functions and Procedures: Basic Authentication, from
<http://www.zeitungsjunge.de/delphi/mime/Help/DIMime.htm>.
2364. Authentication, Authorization, and Access Control, from httpd.apache.org/docs.
2365. John Franks, (1999), HTTP Authentication: Basic and Digest Access Authentication, from
<http://www.ietf.org/rfc/rfc2617.txt>.
2366. Jeff Kercher, Edward Jezierski, (2001), Authentication in ASP.NET: .NET Security Guidance, from
<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/authaspdotnet.asp>.
2367. Digital Certificates, from www.bitpipe.com/tlist/Digital-Certificates.html.
2368. Vijay Bollapragada, IPSec Authentication and Authorization Models, from
www.ciscopress.com/articles/article.asp?p=421514&seqNum=4 - 31k -.
2369. Certificate-based Authentication, from <http://www.microsoft.com/technet/security/Bulletin/MS02-048.mspx>.
2370. Abel Banda, (2003), Forms-based Authentication, from
www.ondotnet.com/pub/a/dotnet/2003/01/06/formsauthp1.html.
2371. Kimon Rethis Biometrics Authentication, from www.csun.edu.
2372. Fingerprint-based Identification, from www.barcode.ro/tutorials/biometrics/fingerprint.html
2373. Michael Anissimov, Retina Scanning, from www.wisegeek.com/how-does-a-retinal-scan-work.htm.
2374. Afghan Woman Recognized After 17 Years, from
<http://www.ct.gov/dss/cwp/view.asp?a=2349&q=304748>.
2375. Bill Gates at the RSA Conference 2006, from <http://www.microsoft.com/billgates/speeches>.

2376. Security Awareness is Rising while Security Protections are falling, from www.miora.com/articles/awareness.htm.
2377. Soumyadip Rakshit, Biometric access control, from <http://www.jiskha.com/science/biology/biometrics.html>.
2378. Prof. Ausif Mahmood RSA (Rivest, Shamir, and Adleman), from www.bridgeport.edu/sed/projects/cs597/Spring_2004/sbhumana/index.htm.
2379. BlowFish, from www.answers.com/topic/blowfish.
2380. Algorithms and Security, from www.tcuconnect.com/help/faq_security.asp.
2381. SHA (Secure Hash Algorithm), from http://www.safeexim.safescrypt.com/SafeDoXX_User_Manual.pdf.
2382. Christopher Allen, Tim Dierks, SSL Handshake Protocol Flow, from <https://www.ipa.go.jp/security/rfc/RFC2246-07EN.html>.
2383. Cryptography Attacks, from www.hack.gr/users/dij/crypto/overview/terminology.html.
2384. Cryptography Attacks, from <http://ieeexplore.ieee.org/iel5/10600/33508/01590056.pdf?isnumber=33508&arnumber=1590056>.
2385. What is a hash function?, from <http://www.rsa.com/rsalabs/node.asp?id=2176>.
2386. What is Capstone?, from <http://www.rsa.com/rsalabs/node.asp?id=2317>.
2387. What are RC5 and RC6?, from <http://www.rsa.com/rsalabs/node.asp?id=2251>.
2388. This challenge is no longer active, from <http://www.rsa.com/rsalabs/node.asp?id=2094>.
2389. Dorothy E. Denning & Dennis K. Branstad, 1996, A Taxonomy for Key Escrow Encryption Systems, from <http://www.cosc.georgetown.edu/~denning/crypto/Taxonomy.html>.
2390. Parameter Tampering, from http://www.imperva.com/resources/glossary/parameter_tampering.html.
2391. Pascal Meunier, (2004), Programming Issues, from www.cerias.purdue.edu/secprog/class2/7.Canon_&_DT.ppt.
2392. About Secure Shell, from <http://www.onsight.com/faq/ssh/ssh-faq-1.html>.
2393. PGP Attack FAQ: The asymmetric cipher <http://www.iusmentis.com/technology/encryption/pgp/pgpattackfaq/asymmetric/>.
2394. Digital Signature Guidelines, from Tutorial <http://www.abanet.org/scitech/ec/isc/dsg-tutorial.html>.
2395. What is public-key cryptography?, from <http://www.rsa.com/rsalabs/node.asp?id=2165>.
2396. What is Public-Key Cryptography?, from <http://www.x5.net/faqs/crypto/q3.html>.
2397. Security FAQs, from http://www.tcuconnect.com/help/faq_security.asp.
2398. RSA Security- 2.1.1 What is public-key cryptography?, from <http://www.rsasecurity.com/rsalabs/node.asp?id=2165>.
2399. Mahmood, (2006), Encryption and Decryption using RSA, from http://www.bridgeport.edu/sed/projects/cs597/Spring_2004/sbhumana/index.htm.
2400. HashCalc, from <http://www.slavasoft.com/hashcalc/>.
2401. MD5 Calculator, from <http://www.bullzip.com/products/md5/info.php>.
2402. HashMyFiles, from http://www.nirsoft.net/utils/hash_my_files.html.
2403. Advanced Encryption Package, from <http://www.aeppro.com/>.
2404. BCTextEncoder, from <http://www.jetico.com/encryption-bctextencoder>.
2405. CommuniCrypt File Encryption Tools, from <http://www.communicrypt.com>.
2406. Steganos LockNote, from <https://www.steganos.com/us/products/for-free/locknote/overview/>.

2407. AxCrypt, from <http://www.axantum.com/axcrypt/>.
2408. AutoKrypt, from <http://www.hiteksoftware.com/autokrypt/data-encryption-software.htm>.
2409. CryptoForge, from <http://www.cryptoforge.com/>.
2410. NCrypt XL, from <http://www.littleelite.net/ncryptxl/>.
2411. ccrypt, from <http://ccrypt.sourceforge.net/>.
2412. WinAES, from <http://fatlyz.com/en/winaes/>.
2413. EncryptOnClick, from <http://www.2brightsparks.com/onclick/eoc.html>.
2414. GNU Privacy Guard, from <http://www.gnupg.org>.
2415. comodo, from <http://www.comodo.com/>.
2416. thawte, from <http://www.thawte.com/>.
2417. verisign, from <http://www.verisign.com>.
2418. entrust, from <http://www.entrust.net/>.
2419. TrueCrypt, from <http://www.truecrypt.org/>.
2420. GiliSoft Full Disk Encryption, from <http://www.gilisoft.com/product-full-disk-encryption.htm>.
2421. DriveCrypt, from http://www.securstar.com/products_drivecrypt.php.
2422. ShareCrypt, from http://www.securstar.com/products_sharecrypt.php.
2423. PocketCrypt, from http://www.securstar.com/products_pocketcrypt.php.
2424. Rohos Disk Encryption, from <http://www.rohos.com/products/rohos-disk-encryption/>.
2425. R-Crypto, from http://www.r-tt.com/data_security_software/.
2426. SafeBit Disk Encryption, from <http://www.safabit.net/>.
2427. DiskCryptor, from http://diskcryptor.net/wiki/Main_Page/en.
2428. alertsec, from <http://www.alertsec.com/software-overview/>.
2429. Symantec Drive Encryption, from <http://www.symantec.com/whole-disk-encryption>.
2430. DriveCrypt Plus Pack, from http://www.securstar.com/products_drivecryptpp.php.
2431. CrypTool, from <http://www.cryptool.org/en>.
2432. CryptoBench, from <http://www.addario.org/cryptobench/>.
2433. JCrypTool, from <http://www.cryptool.org/en/jcryptool>.
2434. Ganzúa, from <http://ganza.sourceforge.net/en/index.html>.
2435. Crank, from <http://crank.sourceforge.net/index.html>.
2436. EverCrack, from <http://evercrack.sourceforge.net/>.
2437. AlphaPeeler, from <http://alphapeeler.sourceforge.net/index1.htm>.
2438. Draft Crypto Analyzer, from <http://www.literatecode.com/draca>.
2439. Linear Hull Cryptanalysis of PRESENT, from <http://www.ecrypt.eu.org/tools/present-linear-hull>.
2440. mediggo, from <http://code.google.com/p/mediggo/>.
2441. SubCypher, from <http://www.esclepiusllc.com/index.php?page=subcypher>.
2442. MD5 Decrypt, from <http://www.md5decrypt.org/>.
2443. MD5Cracker, from <http://md5crack.com/>.
2444. MD5 Hash Cracker, from <http://www.tmtu.org/pages/passwordtools/hashcracker/>.
2445. Hash Cracker, from <http://www.hash-cracker.com/>.
2446. MD5Decrypter, from <http://www.md5decrypter.com/>.

2447. OnlieHashCrack.com, from <http://www.onlinehashcrack.com/index.php>.
2448. MD5Decrypter.co.uk, from <http://www.md5decrypter.co.uk/>.
2449. Md5.My-Addr.com, from http://md5.my-addr.com/md5_decrypt-md5_cracker_online/md5_decoder_tool.php.
2450. cmd5.org, from <http://www.cmd5.org/>.
2451. Crypt and Decrypt Online Tool Conversion, from <http://myeasywww.appspot.com/utility/free/online/Crypt-and-Decrypt-tool-online/en?command=UTILITY&ID=2>.

Module 20: Penetration Testing

2452. Assessing Network Security, from <http://www.scribd.com/doc/24594933/Assessing-Network-Security>.
2453. Technical (Bottom-Up) Methodology, from <http://www.crcnetbase.com/doi/abs/10.1201/9780203503041.ch6>.
2454. Auditing, from <http://www.crcnetbase.com/doi/abs/10.1201/9781420000047.ch3>.
2455. Automated Penetration Testing – False Sense of Security, from <http://www.it-observer.com/automated-penetration-testing-false-sense-security.html>.
2456. Application Assessment Questioning, from <http://www.technicalinfo.net/papers/AssessmentQuestions.html>.
2457. How are Penetrating Testing conducted?, from www.corsaire.com .
2458. Categories of security assessments, from <http://safari.oreilly.com/0735618682/part06>.
2459. Assessing Network Security, from <http://safari.phptr.com/0735620334/ch01lev1sec3>.
2460. Penetration testing guide, from <http://www.penetration-testing.com/>.
2461. COMPUTER SECURITY PERFORMANCE TESTEXAMPLE, from http://seclists.org/lists/pen-test/2003/Feb/att-0015/Pennetraction_Test_Agreement_txt.
2462. Service Level Agreements, from <http://it.usu.edu/htm/hardware/service-level-agreements>.
2463. Jeff Forrista, (2001), Fireproofing Against DoS Attacks, from <http://www.networkcomputing.com/1225/1225f38.html>.
2464. Konstantinos Karagiannis, Pen-Test Using FoundScan Hardware Appliances, from <http://www.eweek.com/cobrand/0,3223,a=27473&s=1610&ap=,00.asp>.
2465. Pen-Test Using NetRecon, from <http://www.net-security.org/dl/newsletter/txt/issue059.txt>.
2466. Pen-Test Using SATAN, SARA and Security Analyzer, from <http://www.ciac.org/ciac/ToolsUnixNetSec.html>.
2467. E- Commerce Security, from http://netdesignplus.net/publications/victor_sawma_thesis.pdf.
2468. Design Guidelines for Secure Web Applications, from <http://msdn.microsoft.com/library/en-us/dnnetsec/html/thcmch04.asp?frame=true>.
2469. KEN BRANDT, STU GREEN, ENRIQUE ZÚÑIGA, Activity: Escalating Privileges, from <http://infosecuritymag.techtarget.com/ar>.
2470. The Professional Security Testers (PST) Warehouse: Web Proxy, from http://www.professionalsecuritytesters.org/modules.php?name=News&new_topic=16.
2471. Microsoft Security Bulletin (MS99-046) Frequently Asked Questions, <http://www.microsoft.com/technet/security/bulletin/fq99-046.mspx>.
2472. Penetration testing guide, from <http://www.penetration-testing.com/>.
2473. Netscape, from <http://netscape.aol.com/>.

2474. Kyle Lai, (2002), Change MAC Address on Win2K & XP, from <http://seclists.org/pen-test/2002/Nov/0025.html>.
2475. Anatomy of an ARP Poisoning Attack, from <http://www.watchguard.com/infocenter/editorial/135324.asp>.
2476. Hacking Lexicon, from http://www.cybersoft.com/whitepapers/reference/hacking_lexicon.shtml
2477. Information Security Magazine, from http://infosecuritymag.techtarget.com/articles/march01/features4_battle_plans.shtml.
2478. Finding and Fixing Network Vulnerabilities, from <http://www.eweek.com/cobrand/0,3223,a=27473&s=1610&ap=,00.asp>.
2479. Fireproofing against DoS Attacks, from <http://www.networkcomputing.com/1225/1225f38.html>.
2480. Get quality service from your suppliers, from <http://www.businesslink.gov.uk/bdotg/action/detail?type=RESOURCES&itemId=1073792560>.
2481. Stephen, (2006), USU Help Desk, from <http://helpdesk.usu.edu/content/hardware/sla.contracts.php>.
2482. Computer Security Performance Test example Independent Oversight Cyber Security Performance Test, from http://seclists.org/lists/pen-test/2003/Feb/att-0015/Pennetration_Test_Agreement_txt.
2483. Safari Books Online- Microsoft® Windows® Security Resource Kit, from <http://safari.oreilly.com/0735618682/part06>.
2484. Christopher R. Russel, (2001), Penetration Testing with dsniff, from <http://www.ouah.org/dsniffintr.htm>.
2485. IDA, from <https://www.hex-rays.com/products/ida/index.shtml>.
2486. Kismet, from <http://www.kismetwireless.net/download.shtml>.