



UNIVERSITY OF PIRAEUS

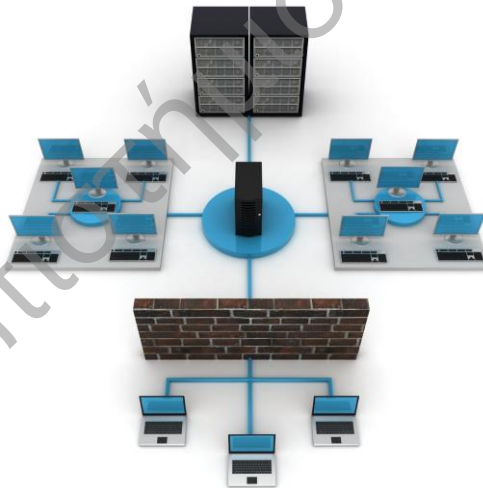
**DEPARTMENT OF
DIGITAL SYSTEMS**

POSTGRADUATE PROGRAM

DIGITAL SYSTEMS SECURITY

THESIS

**“Protecting with Network Security Strategies a
Medium Size Enterprise and Implementing Scenarios
Attacks and Countermeasures on Cisco Equipment”**



SUPERVISOR: DR. CHRISTOS XENAKIS

STUDENT: IOANNIS BAXEVANOS MTE/1118

**PIRAEUS
NOVEMBER 2014**

Contents

1. Introduction.....	1
2. Network Foundation Protection	2
2.1 The Importance of the Network Infrastructure	2
2.2 The Network Foundation Protection Framework	2
2.3 Implementing Network Foundation Protection.....	3
2.4 Best Practices for Securing the Management Plane	4
2.5 Best Practices for Securing the Control Plane.....	5
2.6 Best Practices for Protecting the Data Plane	6
3. Securing the Management Plane on Cisco IOS Devices	8
4. Securing Layer 2 – Data Link Layer	12
4.1.1 CAM Table Overflow Attacks	13
4.1.2 Mitigating CAM Table Overflow Attacks	14
4.2.1 VLAN Hopping.....	14
4.2.2 Mitigating VLAN Hopping Attacks.....	15
4.3.1 Spanning Tree Protocol Manipulation Attacks	16
4.3.2 Preventing STP Manipulation Attacks.....	16
4.4 Rogue Switch - Spanning Tree Modification	17
4.5.1 DHCP Consumption/Starvation Attacks.....	17
4.5.2 Mitigating DHCP Consumption/Starvation Attacks.....	18
4.6.1 Rogue DHCP Server.....	19
4.6.2 Mitigating Rogue DHCP Server	20
4.7.1 MAC Address Spoofing - Man-in-the-Middle Attacks	20
4.7.2 Mitigating MAC Address Spoofing Attacks	21
4.8.1 Private VLAN Vulnerabilities	21
4.8.2 Defending Private VLANs.....	22
4.9.1 IEEE 802.1x EAP Attacks.....	22
4.9.2 IEEE 802.1x EAP Attacks Mitigation	23
5. Securing Layer 3 – Network Layer	24
5.1 Access Control Lists for Packet Filtering	25
5.2 Configuration of Access Lists is Compulsory.....	25
5.3 Stopping Malicious Traffic with an Access List.....	25
5.4 Protection Against Layer 3 Attack Types	27

5.4.1 Traffic Characterization	27
5.4.2 Bogon Blocking and Spoofing	27
5.4.3 DoS and Distributed DoS Attacks	28
5.4.4 Disabling Directed Broadcasts	31
5.4.5 Filtering Directed Broadcasts	31
5.4.6 Simple Reconnaissance Attacks	31
5.4.7 Reflexive Access Lists	33
5.5 Context-Based Access Control (CBAC)	34
5.5.1 Filtering Traffic	34
5.5.2 Inspecting Traffic	34
5.5.3 Detecting Intrusions	35
5.5.4 Generating Alerts and Audits	35
5.5.5 CBAC Enhancements over ACLs	35
5.6 Zone-Based Firewalls (ZBF)	37
5.6.1 How Zone-Based Firewall Operates	37
5.6.2 Specific Features of Zone-Based Firewalls	38
5.6.3 Zones and Why We Need Pairs of Them	38
5.7 Identifying Malicious Traffic on the Network Using Sensors	40
5.7.1 Signature-Based IPS/IDS	40
5.7.2 Policy-Based IPS/IDS	41
5.7.3 Anomaly-Based IPS/IDS	41
5.7.4 Reputation-Based IPS/IDS	41
5.8 Intrusion Prevention System (IPS)	41
5.8.1 Cisco IOS Based IPS	42
5.8.2 Actions That May Be Taken	43
5.8.3 Best Practices When Tuning IPS	43
6. Network Attacks Implementation and Countermeasures	45
6.1 CDP Flooding – DoS Attack	45
6.2 DHCP Starvation/Consumption Attack	49
6.3 VLAN Hopping - DTP and 802.1q Attack	55
6.4 MAC/ARP Spoofing Attack using Ettercap and Xplico Tools	60
6.5 ARP Spoofing Attack using Nmap, Arpspoof, Driftnet, Urlsnarf and Wireshark Tools ..	70
Resources	80

1. Introduction

Security has been important for a long time, with an increasing focus on it over the years. When LANs connecting personal computers began to emerge back in the early 1980s, security was not goal number one, and maybe not even in the top two or three when implementing a network. It was more of an afterthought. Today, however, security for corporate networks is at or near the top of the list.

One challenge to network security is that the threats to a network constantly change. So the security network engineer has to design the network with the best practices for security, and then monitor and vigilantly update it.

This thesis presents the tactics and methods of protecting the local network of a small-medium size company by implementing security measures, protecting the communications and data according to CIA (Confidentiality, Integrity, Availability) and the network devices from crashing, maintaining them functional. More specifically, initially the Network Foundation Protection (NFP) takes place breaking the infrastructure down into smaller components, and then systematically focusing on how to secure each of those components. There is a strategic approach of hardening the network so that we can manage it and allow it to correctly maintain the routing tables, and most important, so that the network stays functional and can forward traffic. Subsequently, there is a detailed reference to the concept of the management plane, which is a collection of protocols and access methods we use to configure, manage, and maintain a network device, and of course examines how to protect it.

The next chapter has to do with the security of Layer 2 technologies. This describes Layer 2 security steps and security features on switches available to combat network security threats. These threats result from weaknesses in Layer 2 of the OSI model - the Data-Link Layer. Switches act as arbiters to forward and control all the data flowing across the network. The current trend is for network security to be solidified through the support of switch security features that build feature-rich, high-performance, and optimized networks. It also examines the integrated security features available on Cisco Catalyst switches to mitigate threats and configures these features in order to build robust networks.

The following chapter highlights some of the most common mitigation techniques available on Cisco platforms and commonly applied on specific Layer 3 devices, such as routers or Layer 3 switches. There are references to the types of Access Control Lists (ACLs), the Firewall and Intrusion Prevention System (IPS) features of the Cisco routers IOS, in order to enhance the security of a network. These options help us to control the incoming and outgoing network traffic by analyzing the data packets and determining whether they should be allowed through or not, based on applied rule sets and policies.

At the last chapter, there are presented five types of network attacks and countermeasures. These attacks are conducted at a simulated environment of the Graphical Network Simulator – GNS3 using virtual network devices loaded with the proper Cisco IOS. Virtual machines are also used as hosts in the network which run the Ubuntu, Windows-XP and Backtrack operating systems. At the role of attacker is the host with Backtrack OS and the role of victims or testing results are the hosts with Ubuntu and Windows-XP OS.

2. Network Foundation Protection

The network infrastructure primarily consists of routers and switches and their interconnecting cables. The infrastructure has to be healthy and functional if we want to be able to deliver network services.

If we break a big problem down into smaller pieces, such as security and what an attacker might do, we can then focus on individual components and parts. By doing this, the work of implementing security becomes less daunting. That is what Network Foundation Protection (NFP) is all about, breaking the infrastructure down into smaller components, and then systematically focusing on how to secure each of those components.

2.1 The Importance of the Network Infrastructure

This section covers a strategic approach to hardening the network so that we can manage it and allow it to correctly maintain the routing tables, and most important, so that the network stays “healthy” and can forward traffic.

Many pieces and parts make up a network, and even one simple component that is not working can cause a failure of the network. If a network does not work, revenue and productivity suffer. In a nutshell, if we have vulnerabilities such as weak passwords (or no passwords), software vulnerabilities, or misconfigured devices, that leaves the door open to attackers. The impact of a down network is huge. It normally affects the work force and other systems and customers that rely on that network. The NFP framework is designed to assist us to logically group functions that occur on the network and then focus on specific security measures that we can take with each of these functions.

2.2 The Network Foundation Protection Framework

For Cisco IOS routers and switches, the NFP framework is broken down into three basic planes (sections):

- **Management plane:** This includes the protocols and traffic that an administrator uses between his workstation and the routers or switches itself. An example is using a remote management protocol such as Secure Shell (SSH) to monitor or configure the router or switch. The management plane is listed here first because until the device is configured (which occurs in the management plane), the device will not be too functional in a network. If a failure occurs in the management plane, it may result in losing the ability to manage a network device.
- **Control plane:** This includes protocols and traffic that the network devices use on their own without direct interaction from an administrator. An example could be a routing protocol. A routing protocol can dynamically learn and share routing information that the router can then use to maintain an updated routing table. If a failure occurs in the control plane, a router may lose the ability to share or correctly learn dynamic routing information, and

as a result not have the routing intelligence to be able to route for the network.

- **Data plane:** This includes traffic that is being forwarded through the network (sometimes called transit traffic). An example is a user on one part of the network who is accessing a server. The data plane represents the traffic that is either being switched or forwarded by the network devices between the client and server. A failure of some component in the data plane results in the customer's traffic not being able to be forwarded. Other times, based on policy, we might want to deny specific types of traffic on the data plane.

Some interdependence exists between these three planes. For example, if the control plane fails, and the router does not know how to forward traffic, this scenario impacts the data plane because the user's traffic cannot be forwarded. Another example is a failure in the management plane that might allow an attacker to configure devices and as a result could cause both a control plane and data plane failure.

2.3 Implementing Network Foundation Protection

The following table describes security measures that we can use to protect each of the three planes.

Plane	Security Measures	Protection Objectives
Management plane	<ul style="list-style-type: none"> -Authentication, Authorization, Accounting (AAA) -Authenticated Network Time Protocol (NTP) -Secure Shell (SSH) -Secure Sockets Layer/Transport Layer Security (SSL/TLS) -Protected syslog -Simple Network Management Protocol Version 3 (SNMPv3) -Parser views 	<p>Authenticate and authorize any administrators. Protect time synchronization by using authenticated NTP. Use only encrypted remote-access protocols such as SSH for CLI and SSL/TLS for GUI tools and use secure versions of SNMP. If plaintext tools are used (such as syslog or Telnet), they should be protected by encryption protocols such as IPsec or should be used out of band (a separate network just for management traffic). A parser "view" is a way to limit what a specific individual, based on his role, can do on the router.</p>
Control plane	<ul style="list-style-type: none"> -Control plane policing (CoPP) -Control plane protection (CPPr) -Authenticated routing protocol updates. 	<p>The control plane tools can be implemented to limit the damage an attacker can attempt to implement directly at the router's IP address (traffic addressed directly to the router, which the router must spend CPU resources to process). Routing protocol updates should be authenticated to remove the possibility of an attacker manipulating routing tables by putting a rogue router running the same routing protocol on the network. The attacker could be doing reconnaissance to learn the routes, or the attacker</p>

		could be attempting to manipulate the resulting data plane by changing the routing on the network.
Data plane	-Access control lists (ACLs) -Layer 2 controls, such as private VLANs, <i>Spanning Tree Protocol (STP)</i> guards -IOS IPS, -Context-Based-Access-Control, Zone-Based Firewall	ACLs, when applied as filters on interfaces, can control which traffic (transit traffic) is allowed on the data plane. At Layer 2, by protecting the infrastructure there, we can avoid a rogue switch from becoming the root of our spanning tree, which would affect the data plane at Layer 2. Firewall filtering and services can also control exactly what traffic is flowing through our network. An example is using an IOS Zone-Based Firewall to implement policy about the data plane and what is allowed.

Table 1 - Components of a Threat Control and Mitigation Strategy.

2.4 Best Practices for Securing the Management Plane

To secure the management plane, the following are the best practices:

- Enforcing password policy, including features such as maximum number of login attempts and minimum password length.
- Implementing *role-based access control (RBAC)*. This concept has been around for a long time in relation to groups. By creating a group that has specific rights, and then placing users in that group, we can more easily manage and allocate administrators. With RBAC, we can create a role (like a group) and assign that role to the users who will be acting in that role. With the role comes the permissions and access. Ways to implement RBACs include using *Access Control Server (ACS)* and CLI parser views (which restrict the commands that can be issued in the specific view the administrator is in). Custom privilege level assignments are also an option to restrict what a specific user may do while operating at that custom privilege level.
- Using AAA services, and centrally manages those services on an ACS server. With AAA, a network router or switch can interact with a centralized server before allowing any access, before allowing any command to be entered, and while keeping an audit trail that identifies who has logged in and what commands they executed while there. Our policies about who can do what can be configured on the central server, and then we can configure the routers and switches to act as clients to the server as they make their requests asking whether it is okay for a specific user to log in or if it is okay for a specific user to issue a specific command.
- Keeping accurate time across all network devices using secure NTP.
- Using encrypted and authenticated versions of SNMP, which includes Version 3.
- Controlling which IP addresses are allowed to initiate management sessions with the network device.

- Locking down syslog. Syslog is sent in plain text. On the infrastructure of our network, only permit this type of traffic between the network device's IP address and the destinations that the network device is configured to send the syslog messages to. In practice, not too many people are going to encrypt syslog data, although it is better to do so. Short of doing encryption, we could use an *out-of-band (OOB)* method to communicate management traffic between our network devices and the management stations. An example is a separate VLAN that user traffic never goes on to, and using that separate VLAN just for the management traffic. If management traffic is sent in-band, which means the management traffic is using the same networks (same VLANs, for instance), all management traffic needs to have encryption, either built in or have it protected by encryption (such as using IPsec).

2.5 Best Practices for Securing the Control Plane

There are three ways to secure the Control Plane, which are the following:

Control plane policing (CoPP): We can configure this as a filter for any traffic destined to an IP address on the router itself. For example, we can specify that management traffic, such as SSH/HTTPS/SSL and so on, can be rate-limited (policed) down to a specific level. This way, if an attack occurs that involves an excessive amount of this traffic, the excess traffic above the threshold set could simply be ignored and not have to be processed directly by the CPU. Another way to think of this is as applying *quality of service (QoS)* to the valid management traffic and policing to the bogus management traffic. This is applied to a logical control plane interface (not directly to any Layer 3 interface) so that the policy can be applied globally to the router.

Control plane protection (CPPr): This allows for a more detailed classification of traffic (more than CoPP) that is going to use the CPU for handling. The three specific subcategories that can be classified are traffic to one of the physical or logical interfaces of the router, certain data plane traffic that requires CPU intervention before forwarding (such as IP options), and *Cisco Express Forwarding (CEF)* exceptions (traffic related to network operations, such as keep-alives or packets with *Time-To-Live (TTL)* mechanisms that are expiring) that have to involve the CPU.

The benefit of CPPr is that we can rate-limit and filter this type of traffic with a more fine-toothed comb than CoPP. This is also applied to a logical control plane interface, so that regardless of the logical or physical interface the packets come in on, the router processor can still have the protection.

Routing protocol authentication: The most routing protocols support authentication. If we use authentication, a rogue router on the network will not be believed by the authorized network devices (routers). The attacker may have intended to route all the traffic through his device, or perhaps at least learn details about the routing tables and networks.

Concluding, using CoPP or CPPr, we can specify which types of management traffic are acceptable at which levels. For example, we could decide and configure the router to believe that SSH is acceptable at 100 packets per second, syslog is acceptable at 200 packets per second, and so on. Traffic that exceeds the

thresholds can be safely dropped if it is not from one of our specific management stations, we can specify all those details in the policy.

2.6 Best Practices for Protecting the Data Plane

The data plane concerns traffic that is going through our network device rather than to a network device. This is traffic from a user going to a server, and the router is just acting as a forwarding device. This is the data plane. Some of the prevalent ways to control the data plane (which may be implemented on an IOS router) are the following:

ACLs used for filtering: There are many types of ACLs and many ways to apply them for filtering. An ACL can be used as a classification mechanism used in other features, such as an IOS firewall, identifying traffic for control plane protection, identifying who is allowed to connect to a vty line, where SNMP is allowed, and so on. In the discussion of protecting the data plane, we focus primarily on ACLs applied directly to interfaces for the purpose of filtering.

IOS firewall support: The firewall features on an IOS router have grown over the years. The older technology for implementing a firewall on IOS routers was called *context-based access control (CBAC)*. CBAC has been replaced with the more current Zone-Based Firewall on the IOS.

IOS IPS: IOS IPS is a software implementation of an *intrusion prevention system (IPS)* that is overlaid on top of the existing routing platform, to provide additional security. IOS IPS uses signature matches to look for malicious traffic. When an alert goes off because of a signature match, the router can prevent the packet from being forwarded, thus preventing the attack from reaching the final destination.

TCP Intercept: This tool allows the router to look at the number of half-formed sessions that are in place and intervene on behalf of the destination device. This can protect against a destination device from a SYN-flood attack that is occurring on the network. The Zone-Based Firewall on an IOS router includes this feature.

Unicast Reverse Path Forwarding (uRPF): This can mitigate spoofed IP packets. When this feature is enabled on an interface, as packets enter that interface the router spends an extra moment considering the source address of the packet. It then considers its own routing table, and if the routing table does not agree that the interface that just received this packet is also the best egress interface to use for forwarding to the source address of the packet, it then denies the packet. This is a good way to limit IP spoofing.

To secure the data plane, the following are the best practices:

- Blocking unwanted traffic at the router. We can implement ACLs inbound or outbound on any Layer 3 interface on the router. With extended ACLs, which can match based on the source and or destination address, placing the ACL closer to the source saves resources because it denies the packet before it consumes network bandwidth and before route lookups are done on a router

that is filtering inbound rather than outbound. Filtering on protocols or traffic types known to be malicious is a good idea.

- Reducing the chance of *denial-of-service (DoS)* attacks. Techniques such as TCP Intercept and firewall services can reduce the risk of SYN-flood attacks.
- Reducing spoofing attacks. For example, we can filter (deny) packets trying to enter our network (from the outside) that claim to have a source IP address that is from our internal network.
- Providing bandwidth management. Implementing rate-limiting on certain types of traffic can also reduce the risk of an attack (*Internet Control Message Protocol [ICMP]*, for example, which would normally be used in small quantities for legitimate traffic).
- IPS. When possible, we could use an IPS to inhibit the entry of malicious traffic into the network.

Normally, for data plane protection we think of Layer 3 and routers. Obviously, if traffic is going through a switch, a Layer 2 function is involved, as well. Layer 2 mechanisms that we can use to help protect the data plane include the following:

- Port security to protect against MAC addresses flooding and *CAM(content-addressable memory)-overflow* attacks. When a switch has no more room in its tables for dynamically learned MAC addresses, there is the possibility of the switch not knowing the destination Layer 2 address (for the user's frames) and forwarding a frame to all devices in the same VLAN. This might give the attacker the opportunity to eavesdrop.
- *Dynamic Host Configuration Protocol (DHCP)* snooping to prevent a rogue DHCP server from handing out incorrect default gateway information and to protect a DHCP server from a starvation attack (where an attacker requests all the IP addresses available from the DHCP server so that none are available for clients who really need them).
- *Dynamic ARP inspection (DAI)* can protect against *Address Resolution Protocol (ARP)* spoofing, ARP poisoning (which is advertising incorrect IP-to-MAC address mapping information), and resulting Layer 2 man-in-the-middle attacks.
- IP source guard, when implemented on a switch, verifies that IP spoofing is not occurring by devices on that switch.

For the last above mentioned, we will refer in much more detailed information at the "Securing Layer 2 Technologies" chapter. [1]

3. Securing the Management Plane on Cisco IOS Devices

Accessing and configuring Cisco devices is a common occurrence for an administrator. Malicious router management traffic from an unauthorized source can pose a security threat. For example, an attacker could compromise router security by intercepting login credentials such as the username and password. This chapter presents the concept of the management plane, which is a collection of protocols and access methods we use to configure, manage, and maintain a network device, and examines how to protect it.

By default, when we connect to a console port we are not prompted for a username or any kind of password. By requiring a username or password, we are taking the first steps toward improving what is called the management plane on this router or switch.

The management plane includes not only configuration of a system, but also who may access a system and what they are allowed to do while they are logged in. The management plane also includes messages to or from a Cisco router or switch that is used to maintain or report on the current status of the device, such as a management protocol like Simple Network Management Protocol (SNMP).

When we want to connect a device remotely, we are connecting over IP. This might be possible for an unauthorized person to also connect remotely. The management plane would enable us to control who may connect to manage the device, when they may connect, what they may do, and report on anything that they did. At the same time, we want to ensure that all the packets that go between the device being managed and the computer where the administrator is sitting are encrypted so that anyone who potentially may capture the individual packets while going through the network could not interpret the contents of the packets which might contain sensitive information about the configuration or passwords used for access.

When implementing a network, it is essential to use the following best practices that improve the security posture of the management plane.

Strong passwords: The passwords should be complex and difficult to guess. An attacker can break a password in several ways, including a dictionary or a brute force attack. A dictionary attack automates the process of attempting to log in as the user, running through a long list of words (potential passwords), when one attempt fails, the attack just tries the next one and so on. A brute-force attack doesn't use a list of words, but rather tries thousands or millions of possible character strings trying to find a password match (modifying its guesses progressively if it incorrectly guesses the password or stops before it reaches the boundary set by the attacker regarding how many characters to guess, with every possible character combination being tried). A tough password takes longer to break than a simple password.

User authentication and AAA: Require administrators to authenticate using usernames and passwords. This is much better than just requiring a password and not knowing exactly who the user is. To require authentication using usernames and passwords, we can use a method authentication, authorization, and accounting (AAA). Using this, we can control which administrators are allowed to connect to which devices and what they can do while they are there, and we can create an audit trail (accounting records) to document what they actually did while they were logged in.

Role-based access control (RBAC): Not every administrator needs full access to every device, and we can control this through AAA and custom privilege levels/parser views. For example, if there are junior administrators, we might want to create a group that has limited permissions. We could assign users who are junior administrators to that group, they then inherit just those permissions. This is one example of using RBAC. Another example of RBAC is creating a custom privilege level and assigning user accounts to that level. Regardless of how much access an administrator has, a change management plan for approving, communicating, and tracking configuration changes should be in place and used before changes are made.

Encrypted management protocols: When using in-band or out-of-band management, encrypted communications should be used, such as Secure Shell (SSH) or Hypertext Transfer Protocol Secure (HTTPS). Out-of-band (OOB) management implies that there is a completely separate network just for management protocols and a different network for end users and their traffic. In-band management is when the packets used by our management protocols may intermingle with the user packets (considered less secure than OOB). Whether in-band or out-of-band, if a plaintext management protocol must be used, such as Telnet or HTTP, we can use in combination with a virtual private network (VPN) tunnel that can encrypt and protect the contents of the packets being used for management.

Logging: Logging is a way to create an audit trail. Logging includes not only what administrators have changed or done, but also system events that are generated by the router or switch because of some problem that has occurred or some threshold that has been reached. Determine the most important information to log, and identify logging levels to use. A logging level simply specifies how much detail to include in logging messages, and may also indicate that some less-serious logging messages do not need to be logged. Because the log messages may include sensitive information, the storage of the logs and the transmission of the logs should be protected to prevent tampering or damage. Allocate sufficient storage capacity for anticipated logging demands. Logging may be done in many different ways, and our logging information may originate from many different sources, including messages that are automatically generated by the router or switch and sent to a syslog server. A syslog server is a computer that is set up to receive and store syslog messages generated from network devices.

Simple Network Management Protocol (SNMP): It is preferable to use SNMP Version 3 for security purposes, because of its authentication and encryption

capabilities. We can use SNMP to change information on a router or switch, and we can also use it to retrieve information from the router or switch. An SNMP trap is a message generated by the router or switch to alert the manager or management station of some event.

Unfortunately, the ability to get information from or send configuration information to a managed device poses a potential security vulnerability. Specifically, if an attacker introduces a rogue NMS into the network, the attacker's NMS might be able to gather information about network resources by polling the Management Information Bases (MIBs) of managed devices. In addition, the attacker might launch an attack against the network by manipulating the configuration of managed devices by sending a series of SNMP SET messages.

The security weaknesses of SNMPv1 and SNMPv2c are addressed in SNMPv3. SNMPv3 uses the concept of a security model and a security level as described to the following table.

Security Model	Security Level	Authentication Strategy	Encryption Type
SNMPv1	noAuthNoPriv	Community string	None
SNMPv2c	noAuthNoPriv	Community string	None
SNMPv3	noAuthNoPriv	Username	None
	authNoPriv	MD5 or SHA	None
	authPriv	MD5 or SHA	CBC-DES (DES-56)

Table 2 - SNMP Security Models and Security Levels Supported by Cisco IOS.

Through the use of the security algorithms, as shown above, SNMPv3 dramatically increases the security of network management traffic as compared to SNMPv1 and SNMPv2c. Specifically, SNMPv3 offers three primary security enhancements:

- **Integrity:** Using hashing algorithms, SNMPv3 can ensure that an SNMP message was not modified in transit.
- **Authentication:** Hashing allows SNMPv3 to validate the source of an SNMP message.
- **Encryption:** Using the CBC-DES (DES-56) encryption algorithm, SNMPv3 provides privacy for SNMP messages, making them unreadable by an attacker who might capture an SNMP packet.

Network Time Protocol (NTP): Using NTP to synchronize the clocks on network devices so that any logging that includes time stamps may be easily correlated. Preferably, using NTP Version 3 leverages its ability to provide authentication for time updates. This becomes very important to correlate logs between devices in case there is ever a breach and we need to reconstruct (or prove in a court of law) what occurred.

To configure the NTP, we first need to know what the IP address is of the NTP server we will be working with, and we also want to know what the authentication key is and the key ID. NTP authentication is not required to function, but is preferable to ensure that the time is not modified because of a rogue NTP server sending inaccurate NTP messages using a spoofed source IP address.

Secure system files: Making it difficult to delete, whether accidentally or on purpose, the startup configuration files and the IOS images that are on the file systems of the local routers and switches. We can do so by using built-in IOS.

More specifically, if a router has been compromised, and the flash file system and NVRAM have been deleted, there could be significant downtime as the files are put back in place before restoring normal router functionality. The Cisco Resilient Configuration feature is intended to improve the recovery time by making a secure working copy of the IOS image and startup configuration files (which are referred to as the *primary bootset*) that cannot be deleted by a remote user.

Thus, we enabled and saved the primary boot-set to a secure archive in persistent storage. After any legitimate modification to the running configuration, we could also enable the creation of a secure configuration archive, saving this to the flash memory of the router. In case we would like to undo this feature, we must be connected via the console. This prevents remote users from disabling the feature. [1]

4. Securing Layer 2 – Data Link Layer

In this section, there will be presented many different security threats that focus on Layer 2 technologies and how to address them, implementing countermeasures. Everything at Layer 3 and higher is encapsulated into some type of Layer 2 frame. If the attacker can interrupt, copy, redirect, or confuse the Layer 2 forwarding of data, that same attacker can also disrupt any type of upper-layer protocols that are being used. Switches act as arbiters to forward and control all the data flowing across the network. The current trend is for network security to be solidified through the support of switch security features that build feature-rich, high-performance, and optimized networks.

With the rapid growth of IP networks in the past years, high-end switching has played one of the most fundamental and essential roles in moving data reliably, efficiently, and securely across networks. The Data-Link layer provides the functional and procedural means to transfer data between network entities with interoperability and interconnectivity to other layers, but from a security perspective, the data-link layer presents its own challenges. Network security is only as strong as the weakest link, and Data-Link layer is no exception. Applying first-class security measures to the upper layers (Layers 3 and higher) does not benefit the network if Layer 2 is compromised. Cisco switches offer a wide range of security features at Layer 2 to protect the network traffic flow and the devices themselves. These integrated security features will be reported in this chapter in order to mitigate threats that result from the weaknesses in Layer 2 of the OSI model.

Unlike hubs, switches can regulate the flow of data between their ports by creating almost “instant” networks that contain only the two end devices communicating with each other. Data frames are sent by end systems, and their source and destination addresses are not changed throughout the switched domain. Switches maintain content-addressable memory (CAM) lookup tables to track the source addresses located on the switch ports. These lookup tables are populated by an address-learning process on the switch. In case of the destination address of a frame is not known or the frame received by the switch is destined for a broadcast address, the switch forwards the frame out all ports. With their ability to isolate traffic and create the “instant” networks, switches can be used to divide a physical network into multiple logical or VLANs through the use of Layer 2 traffic segmentation. VLANs enable network administrators to divide physical networks into a set of smaller logical networks. Like their physical counterparts, each VLAN consists of a single broadcast domain isolated from other VLANs and work by tagging packets with an identification header and then restricting the ports that the tagged packets can be received on to those that are part of the VLAN. The two most prevalent VLAN tagging techniques are the IEEE 802.1q tag and the Cisco Inter-Switch Link (ISL) tag.

From this point till the end of this chapter we will present types of Layer 2 attacks and the countermeasures for each case.

4.1.1 CAM Table Overflow Attacks

A Context Addressable Memory (CAM) table is a switch's table with great capacity but not infinite, which remembers the entire MAC addresses which learned dynamically. At a topology with more than one switch, MAC addresses are learning dynamically from one to another.

When a Layer 2 switch receives a frame, the switch looks in the CAM table for the destination MAC address. If an entry exists for that MAC address, the switch forwards the frame to the port identified in the CAM table for that MAC address. If the MAC address is not in the CAM table, the switch forwards the frame out all ports on the switch. If the switch sees a response as a result of the forwarded frame, it updates the CAM table with the port on which the communication was received. In a typical LAN environment where there are multiple switches connected on the network, all the switches receive the unknown destination frame.

As previously mentioned, the CAM table has a limited size. Cisco Catalyst switches use the 63 bits of source (MAC, VLAN, and so on) and create a 14-bit hash value. If the value is the same, there are eight buckets in which to place CAM entries. These entries expire after a certain inactivity period. The default on the Cisco Catalyst switch is 5 minutes. If enough MAC addresses are flooded to a switch before existing entries expire, the CAM table fills up, and new entries are not accepted. When the CAM table is full, the switch starts flooding the packets out all ports. This incident is called a "CAM table overflow".

In a CAM table overflow attack, an attacker sends thousands of bogus MAC addresses from one port, which looks like valid hosts' communication, to the switch. One of the more popular tools used for launching this type of attack is called Macof, which was written using PERL code, ported to C language, and bundled into the Dsniff suite. Dsniff is a collection of tools for network auditing and penetration testing. Macof can generate 155,000 MAC entries on a switch per minute. The goal is to flood the switch with traffic by filling the CAM table with false entries. Once this table is flooded with MAC addresses and the device starts to connect to the network, the switch does not have enough memory to remember where everyone locates. This gives the attacker the ability to do eavesdropping attack. So if MAC table is flooded and PC1 is talking to PC2, where normally this could be a private conversation, because the switch forgot where the MAC address of PC2 is, because of all the offending MAC addresses which have come to the system and there is no enough room for another. Now the switch with an unknown frame of PC2 will forward - flood this frame to all ports. Thus everything that is coming from PC1 destined to PC2 will be recorded from the attacker. [2]

4.1.2 Mitigating CAM Table Overflow Attacks

We can mitigate CAM table overflow attacks in several ways. One of the primary ways is to configure port security on the switch. We can apply port security in three ways:

- **Static secure MAC addresses** - A switch port may be manually configured with the specific MAC address of the device that connects to it.
- **Dynamic secure MAC addresses** - The maximum number of MAC addresses that will be learned on a single switch port is specified. These MAC addresses are dynamically learned, stored only in the address table, and removed when the switch restarts.
- **Sticky secure MAC addresses** - The maximum number of MAC addresses on a given port may be dynamically learned or manually configured. The manual configuration is not a recommended method because of the high administrative overhead. The sticky addresses will be stored in the address table and added to the running configuration. If the addresses are saved in the configuration file, the interface does not need to dynamically relearn them when the switch restarts.

The type of action taken when a port security violation occurs falls into the following three categories:

- **Protect** - If the number of secure MAC addresses reaches the limit allowed on the port, packets with unknown source addresses are dropped until a number of MAC addresses are removed or the number of allowable addresses is increased. We receive no notification of the security violation in this type of instance.
- **Restrict** - If the number of secure MAC addresses reaches the limit allowed on the port, packets with unknown source addresses are dropped until some number of secure MAC addresses is removed or the maximum allowable addresses is increased. In this mode, a security notification is sent to the Simple Network Management Protocol (SNMP) server (if configured) and a syslog message is logged. The violation counter is also incremented.
- **Shutdown** - If a port security violation occurs, the interface changes to error-disabled and the LED is turned off. It sends an SNMP trap, logs to a syslog message, and increments the violation counter. [2]

4.2.1 VLAN Hopping

VLANs are a simple way to segment the network within an enterprise, to improve performance and simplify maintenance. Each VLAN consists of a single broadcast domain. VLANs work by tagging packets with an identification header. Ports are restricted to receiving only packets that are part of the VLAN. The VLAN information may be carried between switches in a LAN using trunk ports. Trunk ports have access to all VLANs by default. They route traffic for multiple

VLANs across the same physical link. Two types of trunks are used: 802.1q and ISL. The trunking mode on a switch port may be sensed using Dynamic Trunk Protocol (DTP), which automatically senses whether the adjacent device to the port may be capable of trunking. If so, it synchronizes the trunking mode on the two ends. The DTP state on a trunk port may be set to auto, on, off, desirable, or no negotiate. The DTP default on most switches is auto.

One of the areas of concern with Layer 2 security is the variety of mechanisms by which packets that are sent from one VLAN may be intercepted or redirected to another VLAN, which is called VLAN hopping. VLAN hopping attacks are designed to allow attackers to bypass a Layer 3 device when communicating from one VLAN to another. The attack works by taking advantage of an incorrectly configured trunk port.

It is important to note that this type of attack does not work on a single switch because the frame will never be forwarded to the destination. But in a multi-switch environment, a trunk link could be exploited to transmit the packet. There are two different types of VLAN hopping attacks:

- **Switch spoofing** - The network attacker configures a system to spoof itself as a switch by emulating either ISL or 802.1q, and DTP signaling. This makes the attacker appear to be a switch with a trunk port and therefore a member of all VLANs.
- **Double tagging** - Another variation of the VLAN hopping attack involves tagging the transmitted frames with two 802.1q headers. Most switches today perform only one level of decapsulation. So when the first switch sees the double-tagged frame, it strips the first tag off the frame and then forwards with the inner 802.1q tag to all switch ports in the attacker's VLAN as well as to all trunk ports. The second switch forwards the packet based on the VLAN ID in the second 802.1q header. This type of attack works even if the trunk ports are set to off. [2]

4.2.2 Mitigating VLAN Hopping Attacks

Mitigating VLAN hopping attacks requires the following configuration modifications:

- Always using dedicated VLAN IDs for all trunk ports.
- Disabling all unused ports and placing them in an unused VLAN.
- Setting all user ports to non-trunking mode by disabling DTP.
- For backbone switch-to-switch connections, explicitly configuring trunking.
- Not using the user native VLAN as the trunk port native VLAN.
- Not using VLAN 1 as the switch management VLAN. [2]

4.3.1 Spanning Tree Protocol Manipulation Attacks

Spanning Tree Protocol (STP) prevents bridging loops in a redundant switched network environment. By avoiding loops, we can ensure that broadcast traffic does not become a traffic storm.

STP is a hierarchical tree-like topology with a “root” switch at the top. A switch is elected as root based on the lowest configured priority of any switch (0 through 65,535). When a switch boots up, it begins a process of identifying other switches and determining the root bridge. After a root bridge is elected, the topology is established from its perspective of the connectivity. The switches determine the path to the root bridge, and all redundant paths are blocked. STP sends configuration and topology change notifications and acknowledgments (TCN/TCA) using Bridge Protocol Data Unit (BPDU). An STP attack involves an attacker spoofing the root bridge in the topology. The attacker broadcasts out an STP configuration/topology change BPDU in an attempt to force an STP recalculation. The BPDU sent out announces that the attacker’s system has a lower bridge priority. The attacker can then see a variety of frames forwarded from other switches to it. STP recalculation may also cause a denial-of-service (DoS) condition on the network by causing an interruption of 30 to 45 seconds each time the root bridge changes. [3]

4.3.2 Preventing STP Manipulation Attacks

In order to mitigate STP manipulation, we can use the Root Guard and BPDU Guard features from the Cisco IOS Software. These features enforce the placement of the root bridge and the STP domain borders.

More specifically, the STP Root Guard feature is designed to allow the placement of the root bridge in the network. One of the configured switches might be connected to other switches that we do not manage. To avoid this case, we should have already taken precaution steps preventing configured local switches from learning about a new root switch through one of its local ports by configuring root guard on those ports. This will help in preventing tampering of the existing STP topology.

Regarding STP BPDU Guard, it is used to keep all active network topology predictable. By enabling BPDU guard, if a BPDU is seen inbound on a port, the switch port that was forwarding stops and disables the port. A user should never be generating legitimate BPDUs. This configuration, applied to ports that should only be access ports to end stations, helps to prevent another switch (that is sending BPDUs) from being connected to the network. This could prevent manipulation of the current STP topology. A port that has been disabled because of a violation shows a status of err-disabled. We could also configure the switch to automatically bring an interface out of err-disable, based on the reason it was placed there and how much time has passed before bringing the interface back to up. [1]

4.4 Rogue Switch - Spanning Tree Modification

In case of a user wants to plug a new switch replacing his PC, he could change the topology, harming its functionality. A new switch that supports STP, has an auto-index setup so automatically can communicate to the other switch Switch1. The problem here is that the port where the replacement did, is misconfigured, so it allows spanning-tree messages of being sent and received to the rest of the switched network. A possible case is the new switch to become the root of the Spanning-Tree.

Unfortunately, CDP (Cisco Discovery Protocol) is allowed to run on all ports, so the attacker could learn that he is connected on port FastEthernet1/1 of the Switch1. If the attacker finds access to the Switch2 on port FastEthernet1/1 and he connects the cable to the new switch, things would be much worst. If the new switch becomes the root of spanning-tree, the two parallel links (that connects Switch1 with Switch2) will be blocked both of them. Thus, all the traffic between Switch1 and Switch2 will pass through the new-rogue switch. This can be a Man-in-the-Middle attack from the attacker.

In order to avoid this kind of attack, we could enable the feature of BPDU-Guard. This feature can be enabled globally, which affects any port that is configured as an access port, or per interface. His duty is to prevent any inbound BPDU to these ports, shutting them down, because on these ports are connected PCs and no other kind of consoles, such as the rogue-switch in this case.

Furthermore, there are BPDU-Filter and Root-Guard features, we could use. BPDU-Filter doesn't shut the port down but it doesn't allow BPDUs to flow. As far as Root-Guard is concerned, if a switch is connected to other legitimate switches and we don't want the spanning-tree topology to change, we will specify the certain connected ports in a way that these should never be the root ports.

Often, we could implement global configuration (to all access ports) to the switch as far as BPDU-Guard and portfast are concerned, and then we could test these with Yersinia tool. In order to protect the switch, we can enable any portfast interface that will have the benefit of BPDU-Guard through his access ports. Enabling also the error-disable recovery feature, because of the case of BPDU-Guard violation, this feature will bring up the port after a specific predetermined amount of seconds.

4.5.1 DHCP Consumption/Starvation Attacks

The intent of the DHCP Consumption Attack is for the attacker to prevent hosts from gaining access to the network by denying them an IP address by consuming all of the available IP addresses in the DHCP pool.

All Cisco Catalyst switch models use a CAM (Content Addressable Memory) table for Layer 2 switching. As frames arrive on switch ports, the source MAC addresses are learned and recorded in the CAM table. The port of arrival and the VLAN are both recorded in the table, along with a time stamp. If a MAC address learned on one switch port has moved to a different port, the MAC address and time stamp are recorded for the most recent arrival port. Then, the previous

entry is deleted. If a MAC address is found already present in the table for the correct arrival port, only its time stamp is updated.

There could be two scenarios, once with the IP address coming from the DHCP server and once using a static IP address from the same subnet as the IP address in the DHCP server's pool. Since the outcome of the DHCP consumption attack will be the same no matter if an IP address is used from the DHCP Server or a static IP address is used, because there could be two DHCP Consumption scenarios. The one could be a Cisco Catalyst Switch that is acting as the DHCP Server and the other could be a Cisco Router that is acting as the external DHCP Server. [4]

The DHCP server dynamically assigns IP addresses to hosts on a network. The administrator creates pools of addresses available for assignment. A lease time is associated with the addresses.

A DHCP starvation attack works by broadcasting DHCP requests with spoofed MAC addresses. This scenario is achieved with attack tools such as Gobbler, which looks at the entire DHCP scope and tries to lease all the DHCP addresses available in the DHCP scope. This is a simple resource starvation attack, similar to a SYN flood attack. The attacker can then set up a rogue DHCP server and respond to new DHCP requests from clients on the network. This might result in a "man-in-the-middle" attack.

4.5.2 Mitigating DHCP Consumption/Starvation Attacks

The methods used to mitigate a MAC address spoofing attack may also prevent DHCP starvation by using the DHCP snooping feature. Implementation of RFC 3118, Authentication for DHCP Message, will also assist in mitigating this type of attack. We could also limit the number of MAC addresses on a switch port, a mitigation strategy for CAM table flooding, to mitigate DHCP starvation attacks.

Other features on the Cisco Catalyst switch, such as IP Source Guard, may also provide additional defense against attacks. IP Source Guard initially blocks all IP traffic except from DHCP packets captured by DHCP snooping process. When a client receives a valid IP address from the DHCP server, an ACL is applied to the port. This ACL restricts the traffic from the client to those source IP addresses configured in the binding. [2]

The two security features that help the mitigation of this attack the most are DHCP Snooping and Dynamic ARP Inspection (DAI).

More specifically, DHCP Snooping is a security feature capable of intercepting DHCP messages crossing a switch and blocking bogus DHCP offers. DHCP Snooping uses the concept of trusted and untrusted ports. Typically, the trusted ports are used to reach DHCP servers or relay agents, while untrusted ports are used to connect to clients. All DHCP messages are allowed on trusted ports, while only DHCP client messages are accepted on untrusted ports. As neither servers nor relay agents are connected to untrusted ports, server messages like DHCP OFFER, DHCP ACK, and DHCP NAK are dropped on untrusted ports. In addition, DHCP Snooping builds and maintains a MAC-to-IP binding table that is used to validate DHCP packets received from untrusted ports. DHCP Snooping

discards all untrusted DHCP packets not consistent with the information in the binding table. For DHCP snooping to function properly, all DHCP servers must be connected to the switch through trusted interfaces. The DHCP Snooping binding table contains the MAC address, IP address, lease time in seconds, and VLAN port information for the DHCP clients on the untrusted ports of a switch. The information that is contained in a DHCP-Snooping binding table is removed from the binding table when its lease expires or DHCP Snooping is disabled in the VLAN. [4]

Furthermore, Dynamic ARP Inspection (DAI) is a security feature that helps prevent ARP poisoning and other ARP-based attacks by intercepting all ARP requests and responses, and by verifying their authenticity before updating the switch's local ARP cache or forwarding the packets to the intended destinations. The DAI verification consists primarily of intercepting each ARP packet and comparing its MAC address and IP address information against the MAC-IP bindings contained in a trusted binding table. DAI discards any ARP packets that are inconsistent with the information contained in the binding table. The trusted binding table is dynamically populated by DHCP snooping when this feature is enabled. In addition, DAI allows the configuration of static ARP ACLs to support systems that use statically configured IP addresses and that do not rely on DHCP. DAI can also be configured to drop ARP packets with invalid IP addresses, such as 0.0.0.0 or 255.255.255.255, and ARP packets containing MAC addresses in their payloads that do not match the addresses specified in the Ethernet headers.

Another important feature of DAI is that it implements a configurable rate-limit function that controls the number of incoming ARP packets. This function is particularly important because all validation checks are performed by the CPU, and without a rate-limiter, there could be a DoS condition.

DAI also associates a trust state with each interface on the system, similar to DHCP Snooping. Packets arriving on trusted interfaces bypass all DAI validation checks, while those arriving on untrusted interfaces go through the DAI validation process. In a typical network configuration for DAI, all ports connected to host ports are configured as untrusted, while all ports connected to switches are configured as trusted. With this configuration, all ARP packets entering the network from a given switch will have passed the security check.

By default, DAI is disabled on all VLANs, and all ports are configured as untrusted. As discussed earlier, DAI populates its database of valid MAC address to IP address bindings through DHCP snooping. It also validates ARP packets against statically configured ARP ACLs. It is important to note that ARP ACLs have precedence over entries in the DHCP snooping database. ARP packets are first compared to user-configured ARP ACLs. If the ARP ACL denies the ARP packet, then the packet will be denied even if a valid binding exists in the database populated by DHCP snooping. [4]

4.6.1 Rogue DHCP Server

Let's suppose that two PCs are searching for IP addresses in two different switches using the same VLAN. So the router must play the role of a DHCP Server in order to give IP address when it receives a request. In this case, a client issues

a Discover searching for a DHCP Server to take an IP address. The router with enabled the function of DHCP Server will answer with an Offer. The client receives this and answers with a Request. And finally the Server answers with an Acknowledgement.

If an attacker wants to compromise the network and tries to do a Man-in-the-Middle (MitM) attack, he could try to become the DHCP Server, installing some software or having a boot up CD. Supposedly, the attacker has the PC1 with 10.0.0.50/24 IP address and PC2 does a Discover belonging to this network. If the rogue DHCP server at PC1 makes an Offer and the PC2 takes it, PC2 could acquire an 10.0.0.51/24 IP address but it will belong to the gateway of 10.0.0.50/24 network. So every time that a local PC wants to get out of local network, it would do ARP with the address of the gateway that belongs to the attacker, forwarding frames to this address 10.0.0.50/24 and this will forward the frames to the default gateway which may belong to the local router. But in this way, the attacker achieves his goal to eavesdrop anything of this local network and specifically from the PC2 in our case, which may belong to a sensitive section of the company such as human resources, doing DHCP spoofing attack.

4.6.2 Mitigating Rogue DHCP Server

In order to protect from this kind of attack, we enable the function of DHCP snooping prevention that is supported by Cisco switches. Firstly, we enable this function on the switches and secondly, we specify the VLAN (for example VLAN10) which PCs should be protected. This action makes the switch to pay attention to those DHCP messages that go across. Switches already have MAC address tables, but in this case they have to create a DHCP Snooping table to keep track of Layer 2 and Layer 3 addresses that a new PC acquires via a DHCP, for the new hosts that are connecting, and they also have to assume that whoever connects to a port or any other new port of the switches in VLAN10, it is specifying as untrusted. Thus any port that belongs to VLAN10 is not trusted by DHCP snooping feature, which means that any server's packet that belongs to the Offer and Acknowledgement steps of handshake between a new client and DHCP server, the switches will not let them come in on an untrusted port. So, in this case, the malicious user on PC1 tries to do an Offer or an Acknowledgement or any DHCP server related traffic, the port that is connected PC2 will not going to let the frame in by default, with enabled DHCP snooping feature. The only that we have to do is to specify which port is trusted, in which the original DHCP server is connected.

4.7.1 MAC Address Spoofing - Man-in-the-Middle Attacks

MAC spoofing involves the use of a known MAC address of another host that is authorized to access the network. The attacker attempts to make the target switch forward frames destined for the actual host to the attacker's device instead. This is done by sending a frame with the other host's source Ethernet address (MAC) with the objective to overwrite the CAM table entry. After the

CAM is overwritten, all the packets destined for the actual host will be diverted to the attacker. If the original host sends out traffic, the CAM table will be rewritten again, moving the traffic back to the original host port.

Another method of spoofing MAC addresses is to use Address Resolution Protocol (ARP), which is used to map IP addressing to MAC addresses residing on one LAN segment. When a host sends out a broadcast ARP request to find a MAC address of a particular host, an ARP response comes from the host whose address matches the request. The ARP response is cached by the requesting host. ARP protocol also has another method of identifying host IP-to-MAC associations, which is called Gratuitous ARP (GARP), which is a broadcast packet used by hosts to announce their IP address to the LAN to avoid duplicate IP addresses on the network. GARP can be exploited maliciously by an attacker to spoof the identity of an IP address on a LAN segment. This is typically used to spoof the identity between two hosts or all traffic to and from the default gateway. One of the tools used to spoof ARP entries is called Arpspoof and is part of a collection of tools known as Dsniff. [3]

4.7.2 Mitigating MAC Address Spoofing Attacks

We can use the port-security command described in the “Mitigating CAM Table Overflow Attacks” section to specify MAC addresses connected to particular ports. However, this type of configuration has a high administrative overhead and is prone to mistakes. There are other mechanisms, such as hold-down timers, that we can use to mitigate ARP spoofing attacks by setting the length of time an entry will stay in the ARP cache. Hold-down timers by themselves are insufficient to mitigate attacks. It is possible to combine this with the modifications to the ARP cache expiration time for all the hosts, but this is also unmanageable. One recommended alternative is to use private VLANs to mitigate these types of network attacks that we will describe later. A couple of other features of Cisco IOS Software that provide serious protection from this type of attack are the DHCP Snooping and Dynamic ARP Inspection (DAI) which are described in detail previously. [2]

4.8.1 Private VLAN Vulnerabilities

The PVLAN feature prevents interhost communications providing port-based security among adjacent ports within a VLAN across one or more switches. PVLAN provides Layer 2 isolation to quarantine hosts from one another among ports within the same PVLAN. Access ports in a PVLAN are allowed to communicate only with the certain designated router ports. In most cases, this is the default gateway IP address. Private VLANs and normal VLANs can coexist on the same switch. The PVLAN feature allows segregating traffic at Layer 2, thereby transforming a broadcast segment into a nonbroadcast multi-access-like segment. To prevent interhost and interserver communication, PVLAN can be used efficiently because the number of subnets or VLANs is greatly reduced, although the segmented approach within a single network segment is still

achieved. The number is reduced because there is no need to create extra subnet/VLANs.

There are three types of private VLAN ports:

- **Promiscuous:** A promiscuous port can communicate with all interfaces, including the isolated and community ports within a PVLAN. The function of the promiscuous port is to move traffic between ports in community or isolated VLANs. It can use access lists to identify which traffic can pass between these VLANs. Only one promiscuous port is allowed per single PVLAN, and it serves all the community and isolated VLANs in the Private VLAN.
- **Isolated:** An isolated PVLAN port has complete Layer 2 segregation from all the other ports within the same PVLAN, but not from the promiscuous ports. Traffic from the isolated port is forwarded only to the promiscuous ports and none other.
- **Community:** Community ports are logically combined groups of ports in a common community and can pass traffic among themselves and with promiscuous ports. Ports are separated at Layer 2 from all other interfaces in other communities or isolated ports within their PVLAN.

A network vulnerability of private VLANs involves the use of a proxy to bypass access restrictions of the private VLAN. In a proxy attack, frames are forwarded to a host on the network connected to a promiscuous port, such as a router. The network attacker sends a packet with its source IP and MAC address and a destination IP address of the target system but a destination MAC address of the router. The switch forwards the frame to the router. The router routes the traffic, rewrites the destination MAC address as that of the target, and sends the packet out. Because the router is authorized to communicate with the private VLANs, the packet is forwarded to the target system. This type of attack allows for unidirectional traffic only because the private VLAN filter blocks the target's attempts to respond.

4.8.2 Defending Private VLANs

We can configure ACLs on the router port to mitigate private VLAN attacks. We can also use virtual ACLs on the Cisco Catalyst Layer 3 switch platforms to help mitigate the effects of private VLAN attacks. [5]

4.9.1 IEEE 802.1x EAP Attacks

The IEEE 802.1x is a framework for passing the Extensible Authentication Protocol (EAP) messages over a wired or wireless network. EAP over LAN (EAPoL) offers a framework for authentication and control of user traffic to a protected network. A critical flaw in the EAPoL protocol was identified that can be exploited by an intruder to hijack an existing session and thereby gain access to a wireless network resulting in a MITM-type of attack.

The IEEE 802.1x is a device authentication standard originally targeted for use in an Ethernet LAN but that later gained widespread uptake in wireless networks when the vulnerabilities of WEP in the IEEE 802.11 standard were identified. The 802.1x framework defines the guidelines for packaging EAP messages by using EAPoL protocol (Ethernet frames using the EAP encapsulation over LANs). The basic framework of 802.1x has three components. A user or client as a supplicant that requires authentication on the wireless LAN, an authentication server typically a RADIUS server, and the authenticator which is a device that sits between the supplicant and the authentication server (such as wireless access point).

Two critical vulnerabilities were discovered in the EAPoL 802.1x protocol. The first vulnerability is the injection of a forged (spoof) EAP-Success message toward the end of the EAPoL authentication sequence, resulting in an MITM attack. The EAP-Success message is sent from the authenticator to the supplicant, and this message does not have any integrity check to preserve the information before the authenticator and the supplicant transition to the next state in the authentication sequence—that is, the authenticated state. The attacker can send an unsolicited forged EAP-Success message to the supplicant that appears to come from the authenticator, allowing the intruder to passively establish itself in the network path between the supplicant and the authenticator. The second vulnerability exists when an intruder can hijack an existing session that is already established. After the supplicant has successfully authenticated with the authentication server, the authenticator and the supplicant state both move to the authenticated state. An intruder can send a maliciously crafted dissociate frame to the supplicant spoofing the authenticator source MAC address. This causes the supplicant to believe that the message comes from the authenticator instructing it to terminate the session and dissociate from the wireless network. Note that the authenticator is still maintaining an authenticated and associated state for this session. The attacker then forges the source MAC address of the dissociated system to assume its identity, gaining successful access to the network impersonating the victim. The authenticator has no way to reconfirm this, and therefore the session is hijacked. [5]

4.9.2 IEEE 802.1x EAP Attacks Mitigation

There is no integrity mechanism available in the EAPoL protocol that can mitigate an 802.1x attack in a wireless network. The recommended workaround is to use the Protected EAP (PEAP) protocol instead, and thereby deploy 802.1x on wireless access points. The PEAP authentication protocol was developed to address these and other concerns about 802.1x, in particular its use in a wireless network. The PEAP structure offers integrity by implementing the authentication sequence in two parts:

- A TLS session is established between the supplicant and the PEAP authentication server.
- EAP exchange is carried out over the TLS session to authenticate the supplicant that is using a defined EAP authentication protocol.

As of this writing, there are no known vulnerabilities identified in PEAP. [5]

5. Securing Layer 3 – Network Layer

This section highlights some of the most common mitigation techniques available on Cisco platforms and commonly applied on specific Layer 3 devices, such as routers or Layer 3 switches. More specifically, we are referring to the types of Access Control Lists (ACLs) and the Firewall features of the Cisco routers IOS, in order to enhance the security of a network. These options help us to control the incoming and outgoing network traffic by analyzing the data packets and determining whether they should be allowed through or not, based on applied rule sets and policies. The use purpose of a firewall is to establish a barrier between a trusted, secure internal network and the Internet, the insecure and untrusted network.

More detailed, the roles of firewall technics using access lists include the following:

- *Packet Filtering.* ACLs are implemented inbound or outbound on various interfaces of the router and they are used to control, what transit traffic is forwarded by the router.
- *Stateful Filtering.* This technic has to do with the router, which remembers the session of an internal user (from the inside network), which established going out to the Internet. The router knows the user's source and destination IP addresses and ports numbers. It also knows what steps of handshake is being held, and monitors all these to the stateful session table. The purpose of this is when an external device (from the Internet) responds to a user's request, the arriving traffic to the router doesn't drop, like in other cases, because of the stateful filtering technic of remembering the accurate elements of the user.
- *Reflexive ACLs.* This technic is implementing at the outside interface of the router. Every time that an internal user goes out of the local network the key-name "reflect" dynamically creates an inverted ACL entry. It seems to be a mirror image in order to have the flexibility to use it for the return traffic.
- *Context Based Access Control (CBAC).* This technic does stateful filtering, filtering traffic, inspecting traffic, detecting intrusions and generating alerts and audits.
- *Zone Based Firewall (ZBF).* This technic is the most complete and recent from the previous ones. ZBF includes the features of identifying zones, traffic (class maps), actions (policy maps) and specifying the policy to use on the zone pair.

At the outside interface of the boarder router we could deny everything that is coming in and we inspect traffic that is going out from the internal network, we could also use CBAC rule which inspect the traffic that is going out, so the reply traffic can dynamically bypass the ACL. We could also create zones such as a zone called "outside" implementing this on the outside interface of our perimeter router, because of facing the untrusted network Internet. A second zone called "inside" could be implemented on the inside interface which belongs to the trusted internal network. What stands out in ZBF technic is that the routed traffic between zones is denied by default. Thus we have to specify the traffic rows

between different zones. All the above mentioned will be presented in more detail below.

5.1 Access Control Lists for Packet Filtering

Access control lists (ACLs) (also referred to as access lists) filter network traffic by controlling whether routed packets are forwarded or blocked at a router's interfaces. A router examines each packet to determine whether to forward or drop the packet, on the basis of the criteria we have specified within the access lists.

Access list criteria could be the source address of the traffic, the destination address of the traffic, the upper-layer protocol, or other information. Notable are the cases that sophisticated users can sometimes successfully evade or fool basic access lists because no authentication is required.

5.2 Configuration of Access Lists is Compulsory

There are many reasons to configure access lists, such as for contents restriction of routing updates or for providence of traffic flow control. One of the most important reasons to configure access lists is to provide security for the network. We should use access lists to provide a basic level of security for accessing the local network. If we do not configure access lists on routers and mainly on the border router, all packets passing through them could be allowed onto all parts of the local network.

Access lists offer a great flexibility in order to implement security policies depending on ours network needs. We can allow one host to access a part of our network and prevent another host from accessing the same area. We can also create users profiles in which we specify the permitted-accepted attitude of them, inside our network.

5.3 Stopping Malicious Traffic with an Access List

In reality, the packet-filtering access lists (meaning an access list applied to an interface with the intention of filtering some of the traffic) is not just used to filter malicious traffic. It can be applied to an interface to enforce a policy.

To begin with, we make a few assumptions about a network topology. First, the routers have routing tables that provides the knowledge of how to forward to any of the networks shown. If a router does not have a route, it will not matter what type of access lists we apply because the router will be unable to forward packets out of any interface because it does not know which interface to use.

So, with routing in place and the functional working router, we will identify some types of malicious traffic the router could filter by leveraging a well-written access list either as a filter or used as a classifier along with another feature.

Attacks that could be mitigated include the following:

■ **IP address spoofing:** An example of this is a device sending a packet but lying about its source IP address. Let's suppose that all the networks in the bottom left of a local network are under our control, meaning we are responsible for them. If we know that all those networks that begin with 44.44, live off of the gig 1/0 interface, an access list could be created and applied outbound on that interface that denies any packets claiming to come from one of those 44.44 networks. This would stop packets that come in on the g3/0 or g2/0 that were claiming to be coming from 44.44.x.x, because the ACL would deny the packet outbound on g1/0. This technique has lots of variations, but the concept is the same: filter the bogus, spoofed traffic, and not forward it. Instead of applying the access list that denies source IP addresses from the 44.44 range outbound on gig 1/0, we also could have applied that same access list inbound on gig 3/0 and gig 2/0. By applying the access list closer to the source of the attacker, we are saving some resources on the router, because the router does not have to do a route lookup, decide which interface to forward out of, only to then deny that packet because of an access list on that outbound interface. Ideally, we must place access lists as close to the source as possible as long as the placement of that access list is implementing the policy that the administrator wants to happen, because of avoiding forward a packet all the way through the network when we know we are going to kill it anyway.

■ **TCP SYN-flood attacks:** This type of an attack is a denial-of-service (DoS) attack. These are designed to take down a device, often a server. It is done by sending the initial sequence for a three-way handshake and either spoofing the source address so that the server is trying to reply to yet another victim or having the attacker intentionally fail to reply in an attempt to tie up resources on the server. Firewall techniques, such as the IOS Zone-Based Firewall, could be used to mitigate that type of attack, but there are other features such as TCP Intercept that the router can use (in conjunction with an access list to identify the servers to protect) to mitigate these types of attacks, as well.

■ **Reconnaissance attacks:** By configuring a filtering ACL and applying it to an interface, we could deny certain types of Internet Control Message Protocol (ICMP) or User Datagram Protocol (UDP) traffic that may be used for an attacker to learn additional details about the networks behind the router. Traffic that could be denied may include the protocols used by traceroute and ping, among other traffic that might not be critical for the network and that might be safely denied without any impact to the production network.

■ **General vulnerabilities:** By implementing an ACL that uses the attitude of least permission, which means that nothing is allowed except for what we explicitly permit, we can remove a whole range of potential vulnerabilities and weaknesses in our network from the eyes and potential access of the attacker. If the ports and protocols are not available to go through the router, the attacker cannot leverage those ports and protocols against a system on the other side of the router. [1]

5.4 Protection Against Layer 3 Attack Types

In this section, we focus on some important filtering rules that we should implement on the boarder router of a network, in order to protect against various types of Layer 3 attacks.

5.4.1 Traffic Characterization

The first and most essential step in the attack mitigation process is gathering relevant information about the characteristics of an attack to determine the type of attack and to devise a relevant threat-mitigation strategy based on attack vectors.

The Access Control List (ACL) is the most commonly adopted technique to classify the packets into various attack streams, and it is valuable for characterizing both known and unknown attacks and for tracing packet streams back to their point of origin. Other features such as debugging, logging, and IP accounting can also be used. However, with recent versions of Cisco IOS Software, access lists and access list logging are predominant in characterization and tracing network attacks.

An ACL with a series of permit statements is used to characterize traffic flows of interest. ACL extends the capability of checking packets based on various options in the packet header as more sophisticated attacks emerge. ACL counters are further used to determine which flows and protocols are potential threats because of their unexpected high volume. After the suspect flows are identified, a logging option can be used to capture additional packet characteristics. [5]

5.4.2 Bogon Blocking and Spoofing

Bogon filtering is the practice of filtering bogons, which are bogus IP addresses. Bogon is also an informal name for an IP packet on the public Internet that claims to be from an area of the IP address space reserved, but not yet allocated or delegated by the Internet Assigned Numbers Authority (IANA) or a delegated Regional Internet Registry (RIR). The areas of unallocated address space are called the bogon space. Bogons are not the same as reserved private address and link-local address ranges, such as those in 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16, and 169.254.0.0/16, which are reserved for private networks. Many types of bogon addresses exist, including the addresses that should be used only internally, such as RFC 1918 addresses, loopback addresses (127.0.0.0/8), reserved IANA addresses, multicast and research addresses (224.0.0.0/4), DHCP local address (169.254.0.0/16) and the documentation/test network (192.0.2.0/24).

What kind of advantage does bogon filtering provide? Statistics from security experts has found that 66 percent of DDoS attacks use bogons. Of this 66 percent, more than half of these attacks use source addresses from a Class D or E address space. Therefore, blocking these addresses will stop more than 60 percent of DDoS attacks right at our front door - the perimeter router. Also, bogon filtering typically is used to prevent spoofing attacks, where the hacker tries to hide his

source IP address. Bogon filtering filters out the addresses that are basically bogus addresses or impossible addresses.

So, in order to block bogus traffic coming into our network we implement access lists which block the IASA unassigned addresses, RFC 1918 private addresses, multicast addresses (224.0.0.0/4), research addresses (240.0.0.0/4), network 0, DHCP local addresses (169.254.0.0/16), loopback addresses (127.0.0.0/8) and documentation/test network (192.0.2.0/24). Furthermore, it is of highly importance to block internal network addresses, for preventing source addresses coming into the network that have the same addresses that we are using.

It is a fact that most security threats (more than 60 percent), originate from inside networks. Therefore, we need to take precautions about restricting traffic leaving the network as well. In this case, we are concerned about two things: what address is in the source field and what address is in the destination field. To restrict traffic leaving the network, we could use a filter similar to the one we created for ingress filtering on the Cisco IOS perimeter router. However, it would be much simpler to create an ACL that specifies exactly what source addresses are allowed out and lets the implicit deny drop everything else.

One problem arises with the previous egress filter. If a hacker can compromise one of the internal devices and gain unauthorized access to it, he might implement a DoS attack from it, attempting to send traffic to bogon network numbers. Some of this, we already are preventing because internal router will not forward certain kinds of traffic, such as multicasts, unless we have configured a multicast routing protocol. However, many network administrators set up a default route to route traffic to destinations that routers do not have in their routing tables. We definitely do not want a default route routing traffic to unassigned addresses. Therefore, we either can filter this in the egress filter, denying the bogon destination addresses and placing these statements before our permit statement, or we can use a static route for these bogon networks and point it to a null interface, commonly called black hole routing.

So, we implement an ACL to prevent traffic from being sent to bogon addresses for the egress filter, as well as to prevent spoofing attacks with bogon addresses as the source IP address in the packet header. [6]

5.4.3 DoS and Distributed DoS Attacks

Denial-of-service (DoS) attacks attempt to limit the operation of a device, possibly even causing it to crash. DoS attacks come in many flavors and can use different protocols, such as ICMP, TCP, and UDP. This section presents some of the most common types of DoS attacks and what we can do with ACLs to prevent or, at a minimum, limit our exposure. [6]

5.4.3.1 Using an ACL to Characterize SYN Flood Attacks

There are many variations of SYN flood attacks, with the most common being a situation in which a target machine is flooded with TCP SYN connection requests. In most cases, the source addresses and source TCP ports of the connection

request packets are randomized and spoofed. The objective is to force the target host to maintain TCP state information for a large number of incomplete connections (half-open connections), also called embryonic connections.

SYN flood attacks are sometimes easy to identify because the target host (such as the HTTP or SMTP server) becomes extremely slow, crashes, or hangs. SYN floods are not the only vector, several other vectors exist that are aimed in a similar flooding attack. Most people focus on SYN floods as a critical security attack vector. In reality, some SYN flood mitigation paths open the door for other TCP-based attack vectors.

There are two major types of SYN-flood attacks:

- *Nonspoofed source addresses*: Easy to trace, usually launched from compromised hosts (user workstations and servers).
- *Spoofed source addresses*: Difficult to trace, when spoofing invalid addresses from Bogon space (unallocated address range) or valid addresses from someone else's address blocks SYN Round Trip Time (RTT) is the interval between the sending of SYN+ACK and reception of the corresponding ACK from the other host (receiver). A successful SYN flood occurs when the number of simultaneous SYNs exceeds the capacity of the victim's TCP Listen queue and the rate of SYNs exceeds the victim's ability to clear the SYN_RCVDs in an interval less than the SYN+ACK RTT. The objective of the attack is to crowd out valid SYN_RCVDs before the client's ACK has a chance to get to the server. If an ACK is received, and there is no available SYN_RCVD waiting, the connection fails and the DoS is successful.

Many features are available that we can use to reduce the impact of SYN floods. The effectiveness of these features depends on the environment, therefore, we should carefully examine these solutions. Some techniques available to prevent or minimize the impact of SYN flood attacks include the following:

- Rate-limiting (CAR).
- Context-Based Access Control (CBAC).
- TCP Intercept.
- On security appliances such as PIX firewalls, static and NAT commands provide an option to monitor and control half-open embryonic connections.
- Anti-spoofing: Do not allow traffic claiming to be sourced from customer IP blocks to ingress from the uplink or Internet.
- Anti-bogon: Do not allow traffic claiming to be sourced from reserved addresses or from an IPv4 block that has yet to be allocated by the Internet Assigned Numbers Authority (IANA).
- A source-based remote triggered black hole (RTBH) filtering technique can also be used as a SYN flood mitigation tool. This feature provides real-time defense against DDoS attacks by using a combination of IP routing features. [5]

5.4.3.2 Using an ACL to Characterize ICMP Flood or Smurf Attack

The Smurf attack, also commonly known as ICMP flooding, has two victims: a target victim and a reflector or amplifier. The attacker sends a large number of ICMP echo requests (pings) to the broadcast address of the reflector subnet, but he uses up his entire bandwidth in order to amplify the attack results. The source addresses of these packets are forged (spoofed) to be the address of the target victim (a server). For each packet sent by the attacker, hosts on the reflector subnet respond to the target victim, thereby flooding the victim network and causing congestion that results in a Denial of Service.

A similar attack called Fraggle uses directed broadcasts in the same technique, sending UDP echo requests instead of ICMP echo requests. Fraggle usually achieves a smaller amplification factor than Smurf and is much less popular.

There are several ways to distinguish the Smurf attack from the simple ping flood:

- Smurf packets are sent to a directed broadcast address, rather than to a unicast address, whereas ordinary ping floods almost always use unicast. This can be checked in the addresses with the log-input keyword on the appropriate access list entry.
- When experiencing a Smurf reflector attack, a disproportionate number of output broadcasts in the show interface counters is displayed, and usually a disproportionate number of broadcasts are sent in the show ip traffic display. A standard ping flood does not increase the background broadcast traffic.
- When experiencing a Smurf reflector attack, there is more outbound traffic toward the uplink, as compared to the inbound traffic from the uplink. In general, there are more output packets than input packets on the suspected interface.

When a Smurf reflector is closer to the intruder than the ultimate target, it is much easier to trace the attack. ISPs need to be closely involved in tracing such attacks. However, in other situations, the reflector may not be closer to the attacker than the target. The target could be on a subnet, with the reflector on the other side of the network.

To stop the routers from being reflectors in such attacks, we disable directed broadcast feature. This should be configured on each interface of all routers. The command that disables this feature drops any packets on the router that are sent to a directed broadcast address that causes multiple hosts to respond to the ICMP echo request.

Given the huge impact that both of these two attacks have, we should take steps to prevent these kinds of massive attacks, which are the a. shutting down networks with amplifiers, b. disabling directed broadcast addresses, c. performing ingress and egress filtering of directed broadcast addresses, d. performing rate limiting through CAR (this does not prevent the attack, but it does limit the amount of bandwidth that the ICMP and UDP echos (or replies)

can use), and e. using IP unicast reverse-path forwarding (uRPF) verification to prevent IP spoofing. [5]

5.4.4 Disabling Directed Broadcasts

Initially, we disable directed broadcast processing on our routers. This is done by going into each router and, on each interface, disabling directed broadcasts. If we miss a router or an interface on a router, directed broadcasts typically will be forwarded by other routers. Thus, we might accidentally make ourselves an amplifier. Some Cisco routers have the feature of directed broadcasts to be disabled as default configuration on router's interfaces. [6]

5.4.5 Filtering Directed Broadcasts

The second filtering solution that we have to implement is ingress and egress filtering. The deny statements drop all ICMP echos and echo requests, and also drop all UDP echo requests. The problem with deny statements for echo is that this means we cannot get back any echo replies for testing. Therefore, if it is needed we can tweak this configuration by allowing echo replies only to certain internal management devices.

In the second solution with the usage of an extended ACL to filter we block all the directed broadcast addresses. This can be a cumbersome task, especially if you have many subnets in your network.

We could also filter directed broadcasts in both directions on the public interface of the border router. The first statement in the first ACL could prevent spoofing of internal addresses by an Internet hacker. The second and third statements could drop any traffic destined to the two internal broadcast addresses. In the second ACL, there could be two sets of commands, in which the first two prevent the internal devices from sending directed broadcasts to the Internet - basically, from becoming the victim (reflector) of an amplification attack. The second two statements could prevent the devices from generating packets with source-directed broadcast addresses, just in case the hacker is inside the local network and wants to trick someone on the outside into sending stuff back into the local network.

With ACLs we can inspect the number of matches with the appropriate show commands. If we observe one of these deny statements incrementing extremely fast, we are under attack. If we see the ICMP echos incrementing, someone is trying to use us as an amplifier. If we see the ICMP echo replies incrementing, we are the intended victim. By adding the log-input parameter to the deny ACL statements from the last section, we can determine from which interface the packets are coming from. [6]

5.4.6 Simple Reconnaissance Attacks

Many network threats begin with a reconnaissance attack. In a reconnaissance attack, the hacker tries to illicit information about a network, including what IP

addresses are in use. Two very basic tools that are useful in this situation are ping and traceroute. Ping, which uses ICMP messages, is useful in testing connectivity, but it also can help a hacker learn what devices are reachable in the targeted network. Therefore, we should limit what types of ICMP messages are allowed in and out of a local network.

5.4.6.1 Ingress Filtering of ICMP Traffic

In order to confront ingress filtering of ICMP traffic, we apply an access list about ICMP. The first ACL statement blocks ICMP echos, which is also useful in preventing DoS attacks such as Smurf. The second ACL statement blocks ICMP redirects. This also can be done with the appropriate (“no ip redirects”) command. One issue with this command is that it needs to be configured on all interfaces of all routers on which redirect attacks are possible. Some hackers use redirect messages to corrupt routing information. The third deny statement prevents subnet mask requests. Hackers can use this message to learn what subnets we have, including the directed broadcasts of these addresses. When this is known, the hacker can implement a DoS Smurf or Fraggle attack. Except from deny statements, we could also do a permit statement in this ACL that allows echo reply messages to be sent only to the administrator’s PC which is necessary so that the administrator can test connectivity to devices on the Internet, but no one else must can. By doing this, we are dropping traffic from amplifiers in a Smurf attack. Following this, the second permit statement allows remaining ICMP traffic into the internal network.

5.4.6.2 Egress Filtering of ICMP Traffic

Besides filtering traffic in the ingress direction, we should restrict egress ICMP traffic, especially to prevent someone inside the local network from executing an attack against an outside destination. Thus, we could apply an access list about ICMP for egress traffic, which its statements should permit certain ICMP message types so that connections can be established between certain devices. The first permit statement could allow the management station to send echo messages, through ping, to test connectivity to devices on the Internet. The second permit statement could allow internal devices to send problems with packet headers to devices on the Internet sending traffic inbound. The packet-too-big parameter in the permit statement allows MTU discovery along the path between the source and destination, and the source quench statement allows devices to adapt to congestion. All other ICMP messages are dropped.

5.4.6.3 Traceroute

Traceroute traces the path of routers along the way to the destination. It is very useful in troubleshooting routing problems. However, hackers can use this to find what routers exist in a network. To prevent traceroute into a local network, we need to block UDP ports 33,400 through 34,400 with the use of an ACL. Because we cannot be sure what specific port traceroute uses, we must filter the

range of ports. If we are concerned about internal devices performing traceroute, we can use the same deny command in an egress filter, however, we need to allow the external replies to the traceroute queries. This can pose a problem because the source port used is a random one. Unfortunately, when using normal extended ACLs, we must open a large range of UDP ports to allow the returning traffic back into the local network in the ingress filter. Therefore, in case we want to deny external users from performing traceroute but want to allow a few internal users, a more suitable a stateful firewall solution, such as reflexive ACLs or CBAC. [6]

5.4.7 Reflexive Access Lists

Unlike standard IP ACLs that can filter on Layer 3 information, and extended IP ACLs that can filter on Layers 3 and 4 information, Reflexive ACLs (RACLs) can filter on Layers 3, 4, and 5 (session layer). In this section we present the usage of RACLs, which implements a stateful firewall function on routers.

5.4.7.1 How RACLs Handle Returning Traffic

Unlike extended IP ACLs, RACLs are aware of state information concerning a connection - at least, to a certain degree. In other words, a reflexive ACL can detect when a user initiates a connection to the outside world and can allow only returning traffic for that user's connection. Unlike extended IP ACLs that can do this somewhat for TCP connections (with the established parameter), RACLs can do this with all IP protocols.

RACLs perform a function similar to that of a true stateful firewall, like CBAC. Only when a session is initiated on the inside of a network is returning traffic allowed back in. RACLs accomplish this feat by using temporary ACL statements that are inserted into an extended ACL filter, which is applied on router's external interface. When the session ends or the temporary entry times out, it is removed from the external interface's ACL configuration. This reduces the exposure to DoS attacks by a hacker. However, RACLs do have some limitations, but at least they provide a solution that is much better than one that uses only extended ACLs.

Reflexive ACLs, however, do look at the source (origin) of the traffic. Only when a valid source originates traffic is a temporary ACL entry created, allowing the returning traffic only for this connection. In other words, the Cisco IOS looks at who establishes a session and uses this information to allow returning traffic for that specific connection. This gives us much tighter control over the type of traffic allowed back into the local network, presenting a much smaller window in our filter that a hacker can take advantage of. Plus, RACLs actually build on the ACL technology of the Cisco IOS. This means they are an extension of extended ACLs, where we can mix and match the two features to provide a more secure perimeter defense. By combining these two filtering features, we are much less susceptible to network attacks such as IP spoofing, DoS, and access attacks. [6]

5.5 Context-Based Access Control (CBAC)

The Cisco IOS Firewall feature set is a Cisco IOS add-on that provides enhanced security functions for Cisco IOS devices. It provides more features than a standard stateful firewall, because it enhances the router's security by including features in the Cisco IOS that allow it to perform the functions of an enterprise firewall, such as the Cisco PIX. One of the key features included in the Cisco IOS Firewall feature set that we could utilize for a network is the Context-based Access Control (CBAC), which has many more features and fewer limitations than RACLs. CBAC provides stateful application layer filtering, including support for unorthodox protocols and multimedia applications. It can examine supported connections for embedded NAT and PAT information and perform the necessary translations. In addition, it can open additional stateful connections for supported applications, such as FTP and H.323.

CBAC provides four main functions:

- Filtering traffic
- Inspecting traffic
- Detecting intrusions
- Generating alerts and audits

5.5.1 Filtering Traffic

One of the main functions of CBAC is to filter traffic intelligently, specifically for TCP, UDP and ICMP connections. As with RACLs, one of its functions is to allow returning traffic into a local network. Also, it can be used to filter traffic that originates on either side of the router - internal or external.

Unlike extended ACLs, which can filter only on Layers 3 and 4, and RACLs, which can filter on Layer 5 (session layer) information, CBAC supports application inspection, meaning that it can examine the contents of certain kinds of packets when making its filtering decision, such as, it can examine SMTP commands in an SMTP connection. It also can examine a connection's messages to determine the state of a connection. For instance, FTP uses two connections, a control and a data connection. CBAC can examine the control connection, determine that a data connection is being created, and add this connection to its state table. CBAC supports many multimedia, as well as other applications, which perform this function. Likewise, CBAC can examine HTTP connections for Java applets and filter them, if so desired.

5.5.2 Inspecting Traffic

CBAC can inspect application layer information and use this to maintain its stateful firewall function, even for applications that open multiple connections or embed NATed addressing or port information in applications.

This inspection process not only allows returning traffic back into the local network, but it also can be used to prevent TCP SYN flood attacks. CBAC can examine the rate at which connections are being made to a service and can shut down these connections if a specified threshold is reached. It also can examine TCP connections to make sure that sequence numbers fall within a certain range, dropping any suspicious packets. Besides examining TCP connections, it can examine traffic for DoS fragment attacks.

5.5.3 Detecting Intrusions

Not only CBAC can inspect traffic to implement a stateful firewall, but it also can use this feature to detect certain kinds of DoS attacks. CBAC even can provide protection against SMTP e-mail attacks, limiting the type of SMTP commands that can be sent to an internal e-mail servers. All of these kinds of attacks can cause CBAC to generate logging information about the attack, as well as optionally resetting TCP connections or dropping malicious packets.

5.5.4 Generating Alerts and Audits

CBAC can generate real-time alerts of problems and detected attacks, as well as provide a detailed audit trail of connection requests. There is the opportunity of logging all network connection requests, including the IP addresses of the source and destination, the ports used in the connection, the number of bytes sent, and at what time the connection started and ended. [6]

5.5.5 CBAC Enhancements over RACLs

Unlike with RACLs, however, many additional things can occur while the Cisco IOS, using CBAC, is inspecting the traffic on the connection or connections in the state table. This section covers some other enhancements that CBAC has over RACLs.

5.5.5.1 TCP Traffic

TCP is handled in the same manner with CBAC as is done with RACLs. With TCP, CBAC inspects the connection and examines the control bits in the TCP segment header. If it sees a teardown process (FIN), the Cisco IOS waits 5 seconds for the connection to be torn down gracefully, and then the dynamic ACL entry is removed from the external ACL and the corresponding entry in the state table is removed. If a TCP session is idle longer than 1 hour, the Cisco IOS removes the entry. One unique thing about CBAC, versus RACLs, is that CBAC also monitors the setup of a connection. With CBAC, the Cisco IOS expects the connection to be set up within 30 seconds (user-configurable) of seeing the first SYN segment. If the connection is not established during this time period, the Cisco IOS removes the entry from its state table and ACL.

Another difference is that CBAC examines the sequence numbers in TCP connections to make sure that they fall within an expected range. If they do not fall within an expected range, CBAC drops these packets and assumes that a spoofing or DoS attack is occurring.

5.5.5.2 UDP Traffic

With TCP, it is easy to determine the state of the connection by examining the control bits in the TCP segment header. However, UDP is connectionless, making the inspection process more difficult. As with ACLs, CBAC approximates the life of a UDP connection. It assumes that if no traffic is seen on a UDP connection for more than 30 seconds (user-configurable), the connection must have completed, therefore, the Cisco IOS removes the entry in the state table (and the dynamic ACL entry). This is similar to ACLs, with the exception of the default timeout. However, CBAC has one UDP enhancement over ACLs, it also inspects DNS queries and replies. With this feature, CBAC expects that when an internal device generates a DNS query, the remote DNS server will respond with a DNS reply within 5 seconds (user-configurable). If a reply is not seen in 5 seconds, the DNS connection entry is removed, to prevent spoofing. Likewise, when the DNS reply is seen from the DNS server, the Cisco IOS immediately removes the entry from its state table (and the dynamic ACL entry). These two enhancements are used to prevent DNS spoofing and DoS attacks.

5.5.5.3 ICMP Traffic

Inspection of ICMP traffic was introduced in Cisco IOS 12.2(11). Before this, CBAC could inspect only TCP and UDP traffic. With ICMP inspection, CBAC can inspect common ICMP message types, including echo request, echo reply, destination unreachable, time exceeded, timestamp request, and timestamp reply messages. CBAC does not inspect other ICMP message types. When inspecting ICMP traffic, the Cisco IOS expects replies to the supported ICMP message types within 10 seconds. If none is seen, the ICMP connection is removed from the state table and the dynamic ACL entry is removed. However, if a response is seen, only the supported message types are allowed (based on the request), other message types are dropped.

5.5.5.4 Extra Connections

Certain applications, such as FTP and multimedia, open additional connections to transfer information. As mentioned, ACLs either do not handle these connections very well or do not handle them at all. CBAC, on the other hand, supports many of these types of connections. CBAC can inspect the control connection for these applications to determine whether a data connection or other connection is being opened. When CBAC notices the addition of a new connection, it automatically adds this information to its state table to permit the connection through the router.

5.5.5.5 DoS Detection and Prevention

CBAC can detect certain kinds of DoS flood attacks. When an attack occurs, the Cisco IOS can take any of the following three actions:

- Block the offending packets
- Protect the internal resource from becoming overloaded with fake connections
- Generate an alert message

To detect DoS attacks, CBAC uses timeout and threshold values to inspect the setup of TCP connections. When TCP connections are being established, they usually do not take more than a second or two. Therefore, if a lot of TCP SYNs are seen from a single source, a threshold can be set to restrict the number of these sessions. In addition, if the connections are not completed within a specific time frame (30 seconds, by default), the Cisco IOS removes this information from its state table and notifies both the source and destination with a TCP RST message on the connection. This is used, especially for the internal resource, to free up these half-open (commonly called embryonic) connections. There is a definition of three different thresholds to limit the number of half-open connections:

- Total number of half-open TCP or incomplete UDP sessions
- Total number of half-open TCP or incomplete sessions over a period of time
- Total number of half-open TCP sessions per host

When these thresholds are reached, the Cisco IOS can start dropping incomplete connections that have not been deleted, notify the source and destination that the connections have been torn down (TCP RST), generate an alert, and/or block TCP traffic from the offending devices.

5.6 Zone-Based Firewalls (ZBF)

Cisco has implemented some form of firewall protection in their IOS for many years. Thus, there is the ability to implement packet filtering with access lists applied to interfaces. The older firewall feature set named Context-Based Access Control (CBAC) provided stateful filtering. However, that CBAC has been replaced by a newer method for stateful filtering and application inspection called the Zone-Based Firewall (ZBF).

5.6.1 How Zone-Based Firewall Operates

With ZBFs, interfaces are placed into zones. Zones are created by the network administrator, using any naming convention that makes sense, such as inside, outside, and DMZ (Demilitarized Zone) are quite common. Then policies are specified as to what transit (user) traffic is allowed to be initiated, (for example,

from users on the inside destined to resources on the outside) and what action the firewall should take, such as inspection (which means to do stateful inspection of the traffic). After traffic is inspected, the reply traffic is allowed back through the firewall because of the stateful filtering feature. The policies are implemented in a single direction (for example, inside to outside). If we want to allow initial traffic in both directions, we create two unidirectional policies for traffic to be allowed and inspected from the inside to the outside, and also from the outside to the inside. We implement two separate policies because the policies themselves are unidirectional.

One benefit of this modular approach is that after policies are in place, if we add additional interfaces, all we need to do is add those interfaces to existing zones, and our policies will automatically be in place.

5.6.2 Specific Features of Zone-Based Firewalls

The ZBF major features include the following:

- Stateful inspection
- Application inspection
- Packet filtering
- URL filtering
- Transparent firewall (implementation method)
- Support for virtual routing and forwarding (VRF)
- Access control lists (ACL) are not required as a filtering method to implement the policy

Many of these features we have mentioned previously and some of the concepts are new. Let's consider a few not yet discussed. URL filtering refers to the ability to control what traffic is permitted or denied (mostly denied) based on the URL that is trying to be accessed by the client. VRFs are virtual routing tables on a Cisco router that can be used to compartmentalize the routing tables on the router instead of keeping all the routes in the global (primary) routing tables. A transparent firewall is implemented at Layer 2 but can still perform analysis of traffic at Layer 3 and higher.

5.6.3 Zones and Why We Need Pairs of Them

A zone is first created by the administrator, and then interfaces can be assigned to zones. A zone can have one or more interfaces assigned to it. Any given interface can belong to only a single zone. There is a default zone, called the self-zone, which is a logical zone. For any packets directed to the router directly (the destination IP represents the packet is for the router), the router automatically considers that traffic to be entering the self-zone. In addition, any traffic initiated by the router is considered as leaving the self-zone. By default, any traffic to or from the self-zone is allowed, but this policy can change.

For the rest of the administrator-created zones, no traffic is allowed between interfaces in different zones. For interfaces that are members of the same zone,

all traffic is permitted by default. If we want to allow traffic between two zones, such as between the inside zone (using interfaces facing the inside network) and the outside zone (interfaces facing the internet), we must create a policy for traffic between the two zones, and that is where a zone pair comes into play.

A zone pair, which is just a configuration on the router, is created identifying traffic sourced from a device in one zone and destined for a device in the second zone. The administrator then associates a set of rules (the policy) for this unidirectional zone pair, such as to inspect the traffic, and then applies that policy to the zone pair.

A small company, with users on the inside network, with the only other connection being the Internet, may want to create two zones, one for the inside and one for the outside. Then they would assign the inside interface to the inside zone, and the outside interface to the outside zone. Then, a policy could be created that specifies that traffic that is initiated from the inside users and going out to the Internet should be inspected and that information should be placed in the stateful database. A zone pair identifying traffic from the inside to the outside would have the policy applied to it, letting it know that the stateful inspection should be done.

A larger company that has a public facing server may have three interfaces and three zones. The zones may be inside, outside, and DMZ. Compared to the small company, this medium-sized company creates an additional zone pair (from outside to DMZ), and then applies a policy to that zone pair to allow outside users to access the servers on the DMZ.

Cisco uses a language called the *Cisco Common Classification Policy Language (C3PL)* for the implementation of the policy. This process has three primary components:

- **Class maps:** These are used to identify traffic, such as traffic that should be inspected. Traffic can be matched based on Layer 3 through Layer 7 of the OSI model, including application-based matching. Class maps can also refer to *access control lists (ACL)* for the purpose of identifying traffic or even call upon other class maps. Class maps can have multiple match statements. A class map can specify that all match statements have to match (which is a *match-all condition*) or can specify that matching any of the entries is considered a match (which is a *match-any condition*). A system defined class map named *class-default* can be used that represents all traffic not matched in a more specific (administratively configured) class map.

- **Policy maps:** These are the actions that should be taken on the traffic. Policy maps call on the class maps for the classification of traffic. Policy maps with multiple sections are processed in order. The primary actions that can be implemented by the policy map are *inspect* (which means that stateful inspection should happen), *permit* (which means that traffic is permitted but not inspected), *drop*, or *log*.

- **Service policies:** This is where you apply the policies, identified from a policy map, to a zone pair. This step actually implements the policy. If a policy map contains multiple actions, based on different class map-identified traffic, the

policy map is processed from top to bottom, applying the actions as traffic matches the class maps. If a specific section of a policy map matches, the action is taken. If traffic does not match, the packet is compared against the next section of the policy map. If none of the sections match the traffic, the default behavior action is taken. The default policy for traffic that is trying to be initiated between two zones (starting in one zone and going to a device in another zone) is an implicit deny. The exception to this default deny is traffic to or from the built-in “self” zone, which is allowed by default. [1]

5.7 Identifying Malicious Traffic on the Network Using Sensors

Sensors can identify malicious traffic in many different ways. This section examines some of the techniques used by IPS and IDS sensors. When the sensor is analyzing traffic, it looks for malicious traffic based on the rules that are currently in place on that sensor. There are several different methods that sensors can be configured to use to identify malicious traffic, including the following:

- Signature-based IPS/IDS
- Policy-based IPS/IDS
- Anomaly-based IPS/IDS
- Reputation-based IPS/IDS

5.7.1 Signature-Based IPS/IDS

A signature is just a set of rules looking for some specific pattern or characteristic in either a single packet or a stream of packets. A new sensor may have thousands of default signatures provided by Cisco. Not all the signatures are enabled, but administrator can enable, disable, customize, and create new signatures to meet the needs of the current network where the sensor is operating. An example is a known attack, such as a cross-site scripting attack, where an HTTP variable contains a quote or bracket character to terminate some HTML syntax, followed by a <SCRIPT> tag. Cisco has written a signature (set of rules) looking for exactly that. Cisco creates additional signatures as new and significant attacks are discovered, which the administrator can implement by updating the signatures on the sensor. Signature-based IPS is the most significant method used on sensors today. For tuning purposes, if a default signature keeps triggering based on traffic that is normal for our network, that is considered a false positive for our specific network, and we could tune the sensor by either disabling that signature or setting the filter so that that signature does not generate an alert when it sees a match from specific IP addresses.

5.7.2 Policy-Based IPS/IDS

This type of traffic matching can be implemented based on the policy for a network. For example, if a company has a policy that states that no Telnet traffic should be used (for security reasons) on specific areas of our network, we can create a custom rule that states that if TCP traffic is seen destined to port 23 (which is port for Telnet) the IPS can generate an alert and drop the packet. In comparison with IDS, it could simply generate an alert but cannot drop the packet on its own because IDS is in promiscuous mode, and not inline. The sneaky part of this one is that policy-based IPS/IDS is most likely implemented by creating a custom signature that causes this policy to be enforced (which would also make this signature based).

5.7.3 Anomaly-Based IPS/IDS

An example of anomaly-based IPS/IDS is creating a baseline of how many TCP sender requests are generated on average each minute that do not get a response. This is an example of a half-formed session. If a system creates a baseline of this (let's suppose the baseline is an average of 30 half-formed sessions per minute), and then notices the half-formed sessions have increased to more than 100 per minute, and then acts based on that and generates an alert or begins to deny packets, this is an example of anomaly-based IPS/IDS. The Cisco IPS/IDS appliances have this ability (called anomaly detection), and it is used to identify worms that may be propagating through the network.

5.7.4 Reputation-Based IPS/IDS

If some type of global attack is creeping across the networks of the world but has not hit our network yet, wouldn't it be nice to know about it so that we can filter that traffic before it enters our network? The answer is yes, obviously. Reputation-based IPS collects input from systems all over the planet that are participating in global correlation, so what other sensors have learned collectively, our local sensor can use locally. Reputation-based IPS/IDS may include descriptors such as blocks of IP addresses, URLs, DNS domains, and so on as indicators of the sources for these attacks. Global correlation services are managed by Cisco as a cloud service. [1]

5.8 Intrusion Prevention System (IPS)

An IOS-based IPS is a software-implemented Intrusion Prevention System (IPS). Many companies purchase appliances that are dedicated to IPS. For companies with limited budgets, however, an IPS implemented on the perimeter router can provide attack visibility and help thwart attacks that it recognizes. A company may have appliances at their corporate headquarters and can still leverage the IOS-based IPS at branch offices.

5.8.1 Cisco IOS Based IPS

The router operates in IPS mode because it is already in-line with the customer's traffic, it is a router after all. Thus we do not need to redesign the topology of the network to put this IPS in place. We just need to have a license that supports the feature, and configure the software. IOS-based IPS supports the following detection technologies:

- Profile based
- Signature based
- Protocol analysis based

One option that is missing from this list that a sensor appliance does have but the IOS IPS does not, is an anomaly-based detection.

The IOS IPS provides the following benefits:

- The ability to do dynamic updates of signatures
- Integrates easily into an existing network
- Compatible to work alongside of other security features, such as Zone-Based Firewalls (ZBF), virtual private network (VPN) termination, access control lists (ACL), authentication, authorization, and accounting (AAA), and many others on the same router as long as there is enough memory and CPU to support all the features
- Can be managed by Cisco Configuration Professional (CCP), IPS Manager Express (IME) and Cisco Security Manager (CSM), and via the command-line interface (CLI)
- Supports attack signatures from the same signature database that is used by the IPS appliances

Several of the features commonly found in Cisco IPS products and included in the IOS IPS Signature Features are the following:

Regular expression string pattern matching.

This feature enables the creation of string patterns using variables. An example of a regular expression is hot|cold, which is part of the signature that would look for a match on the word hot or cold. Using regular expressions can allow a single string to be used to match several possible combinations of that string inside of a packet.

Response actions

This enables the sensor to take action in response to a triggered event, such as denying a packet, creating an alert, resetting the TCP connection, or denying the attacker's future packets for a period of time.

Alarm summarization

This helps prevent resource exhaustion by summarizing events that are all the same or at least the same signature. Under heavy attack, a summary may show the attack happened 15,000 times as opposed to producing 15,000 individual alerts. Summarization is tunable on both the IOS IPS and the appliances.

Threshold configuration

Threshold configuration identifies thresholds, which if exceeded may trigger events. For example, a specific string of text can be identified in a signature. That same signature can specify that an alert will be generated only after that string of text has been seen five times within a 60-second window.

Anti-evasive techniques

Similar to the appliances, the intelligence in the IOS IPS is designed to correctly interpret the actual data regardless if it is fragmented or using a combination of character sets, such as Unicode.

Risk ratings

A calculated number between 0–100 associated with an alert. The higher the number, the more risk is presumed. Identifying our critical servers/hosts enables the system to generate higher risk ratings when signatures to those devices are matched. Those higher risk ratings can then trigger countermeasures against the attacks.

5.8.2 Actions That May Be Taken

The possible actions that we can choose in IOS IPS are as follows:

- Deny attacker inline
- Deny connection inline
- Deny packet inline
- Produce alert (the default action for most signatures)
- Reset TCP connection (effective only against TCP-based attacks)

5.8.3 Best Practices When Tuning IPS

The following are best practices when tuning IPS:

- Beginning with the basic signature category, and seeing how much memory and CPU utilization this takes in the production network, before moving to the advanced signature category which will take significantly more CPU and resources from the router.
- Scheduling downtime for the installation and updates.

- Retiring signatures that are irrelevant to our network to save resources on the router.
- Monitoring free memory to ensure that we do not cause harm to our router by loading too many additional services.
- There are options available that can tell the IOS router to not forward any traffic through an IPS-protected interface if some type of problem causes the signature not to compile. The term for this is *fail closed*. The other option, which indicates that if a problem with the IPS signatures not compiling occurs the router should still forward traffic, is called *fail open* . Based on the security policy, we want to choose the option that meets the needs for the company. A fail close could cause a failure of the network due to a failure of IPS, but it is more secure than fail open.
- For performance reasons, we should be very careful before unretiring and enabling the All category of signatures. [1]

Πανεπιστήμιο Πειραιώς

6. Network Attacks Implementation and Countermeasures

At this last chapter, there are presented five types of network attacks and countermeasures. These are the CDP Flooding Attack which results in Denial-of-Service, DHCP Starvation Attack consuming of all the available IP addresses in a subnet leading to DoS, DTP and 802.1q Attacks which have to do with unauthorized conversion of switch ports from access to trunk (802.1q encapsulation) taking place through the DTP ending up to compromised network functionality regarding VLANs, MAC/ARP Spoofing Attack using Ettercap and Xplico tools and ARP/MAC Spoofing Attack using Nmap, Arpspoof, Driftnet, Urlsnarf and Wireshark tools leading to Man-In-The-Middle attack. These attacks are conducted at a simulated environment of the Graphical Network Simulator – GNS3 using virtual network devices interconnected on different kind of topologies and loaded with the proper Cisco IOS. Virtual machines are also used as hosts in these network topologies which are loaded with the Ubuntu, Windows-XP and Backtrack operating systems. At the role of attacker is used the host with Backtrack OS and the role of victims or testing results are used the hosts with Ubuntu and Windows-XP OS.

6.1 CDP Flooding – DoS Attack

Cisco Discovery Protocol (CDP) is a data link layer proprietary protocol that is enabled on Cisco routers and switches and can be used to leverage connections and to discover information about neighboring Cisco devices. Through CDP, we can collect information about network layer addresses, the Cisco IOS Software version, and the platform type of neighboring Cisco devices.

The following example shows the output from the *show cdp neighbors detail* command, which is executed on a Cisco router to reveal information about a neighboring Cisco device.

```
Device ID: RemoteRouter
Entry address(es):
IP address: 192.168.12.5
Platform: cisco 3725R, Capabilities: Router
Interface: Ethernet0, Port ID (outgoing port): Ethernet0
Holdtime: 114 sec
```

```
Version:
Cisco Internetwork Operating System Software
IOS (tm) 3725 Software (C3725-sy-mz), Version 12.2(3)
Copyright 1986-2002 by Cisco Systems, Inc.
```

```
advertisement version: 2
Duplex: full
```

From this command output, we can determine a whole bunch of information such as the IP address (192.168.12.5), platform (3725), and Cisco IOS version (12.2(3)) of a neighboring device. CDP is not encrypted, and it does not have mechanisms for authentication between devices. A hacker or penetration tester can connect a rogue router or switch and discover information about devices on the network.

CDP also can be used for denial of service (DoS) attacks by flooding a system with CDP messages-advertisements. In case of a router, a malicious user who has a PC connected to this could start sending tens of thousands of CDP messages, advertising different systems with random names. This action could increase the CPU utilization of the network device in unprecedented levels, causing the loss of production traffic, leading in dropping frames. This incident is an effective DoS attack, causing a great damage to a production environment.

Firstly, we clear the router from CDP counters and table. After that, we type the commands *show cdp traffic* and *show cdp neighbors* in order to verify that CDP traffic and neighbors are clean-empty. In a production environment we will have sixty seconds until a new CDP message show up. We can also observe the average of CPU utilization with the command *show process cpu sorted*, which is low this time because of clearing CDP messages commands.

Secondly, launching the Yersinia tool on Backtrack and select the "Launch Attack" button from which a popup window appears selecting the second choice "flooding CDP table" from the first tab "CDP" and clicking OK. The tool launches the attack, sending hundreds of CDP messages. At the main screen of the Yersinia tool is appearing a list of random names (bogus) of devices (DevID), the local interface which these messages are sending from, TTL and date -time.

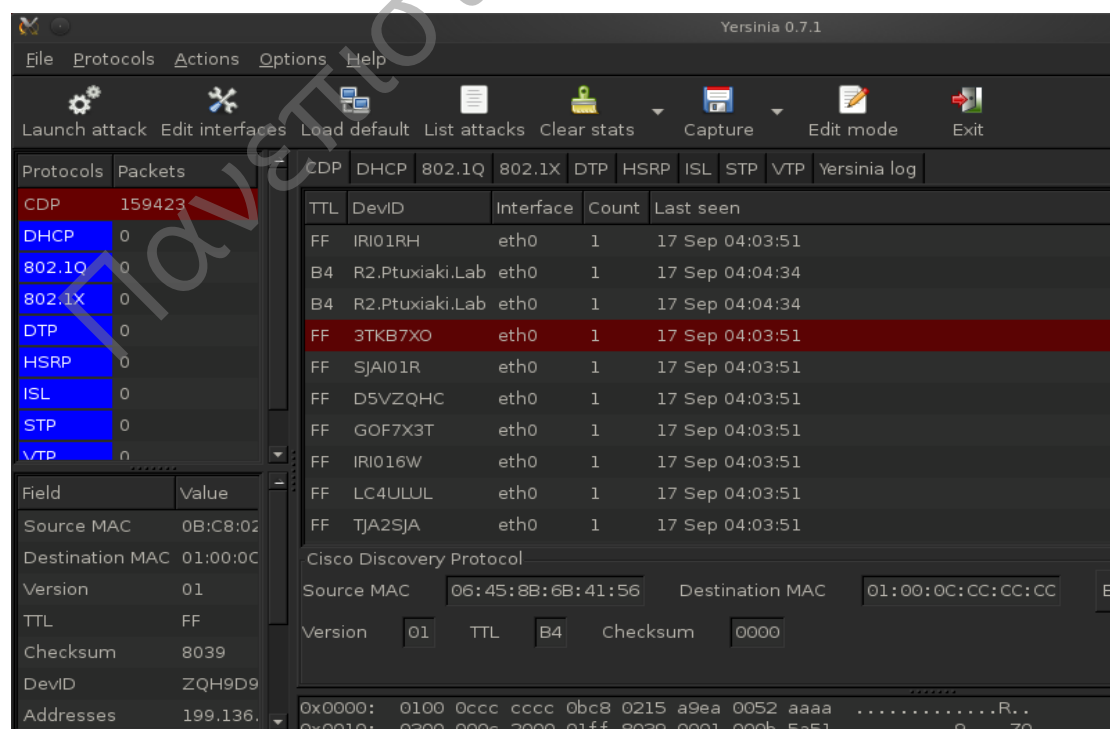


Figure 1 – Running the CDP Flooding Attack.

To verify this process we are going back to the router's CLI. By typing the command `show cdp traffic` we can see how many messages have been received (Input: 12002 and later 24001) from the device, constantly increasing as the attack continues to run.

```
R2#sh cdp traffic
CDP counters :
  Total packets output: 46, Input: 12041
  Hdr syntax: 0, Chksum error: 0, Encaps failed: 0
  No memory: 0, Invalid packet: 0, Fragmented: 0
  CDP version 1 advertisements output: 5, Input: 12002
  CDP version 2 advertisements output: 41, Input: 39
R2#
```

Figure 2 – CDP traffic results as the attack is running.

```
R2#sh cdp traffic
CDP counters :
  Total packets output: 70, Input: 24058
  Hdr syntax: 0, Chksum error: 0, Encaps failed: 0
  No memory: 0, Invalid packet: 0, Fragmented: 0
  CDP version 1 advertisements output: 8, Input: 24001
  CDP version 2 advertisements output: 62, Input: 57
R2#
```

Figure 3 – Increasing values as the attack is running after few seconds.

From the command `show process cpu sorted | include CPU|PID Runtime|CDP Protocol` the average of CPU usage is getting higher. More specific, from the following screenshots we can observe that as the runtime increases, the average of CPU utilization is ranging between 85-90% for five seconds, in contrast with one minute and five minute average which are increasing constantly. This situation has as a result to become the device operationally sluggish and unproductive.

```
R2#show process cpu sorted | include CPU|PID Runtime|CDP Protocol
CPU utilization for five seconds: 89%/100%; one minute: 36%; five minutes: 11%
PID Runtime(ms)  Invoked      uSecs   5Sec    1Min    5Min  TTY Process
 82             39036      26378   1479  57.61% 21.63%  5.61%  0 CDP Protocol

R2#show process cpu sorted | include CPU|PID Runtime|CDP Protocol
CPU utilization for five seconds: 84%/100%; one minute: 65%; five minutes: 23%
PID Runtime(ms)  Invoked      uSecs   5Sec    1Min    5Min  TTY Process
 82             74792      54899   1362  56.34% 40.97% 13.75%  0 CDP Protocol

R2#show process cpu sorted | include CPU|PID Runtime|CDP Protocol
CPU utilization for five seconds: 89%/100%; one minute: 80%; five minutes: 37%
PID Runtime(ms)  Invoked      uSecs   5Sec    1Min    5Min  TTY Process
 82            127416     94101   1354  61.02% 51.32% 22.39%  0 CDP Protocol
```

Figure 4 - The average of CPU utilization is increasing constantly.

From the command `show cdp neighbors` we can observe the random names that were appeared in the main screen of the Yersinia tool (Device ID), the local interface from which the messages arrived, the holdtime and the capabilities of each message.

```
R2#sh cdp neighbors
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID           Local Intrfce      Holdtme    Capability   Platform  Port ID
R2.Ptuxiaki.Lab    Fas 0/1            86         R S I        3725      Fas 0/1
R3.Ptuxiaki.Lab    Ser 1/1            145        R S I        3725      Ser 1/1
R1.Ptuxiaki.Lab    Ser 1/0            150        R S I        3725      Ser 1/0
BEMD5VM            Fas 0/1            220        r            yersinia  Eth 0
KPGB3TY            Fas 0/1            220        R T B H r    yersinia  Eth 0
BW5VMRI            Fas 0/1            220        R T H I r    yersinia  Eth 0
D59ZCH9            Fas 0/1            220        R S H        yersinia  Eth 0
BKBG3TK            Fas 0/1            220        R T I        yersinia  Eth 0
Z9ZQGLC            Fas 0/1            220        R T S H      yersinia  Eth 0
6WMDI01            Fas 0/1            220        R T H I      yersinia  Eth 0
9ZQH9Z9            Fas 0/1            219        R B          yersinia  Eth 0
NEA6WNE            Fas 0/1            219        R T I        yersinia  Eth 0
VMQHC4U            Fas 0/1            219        R I r        yersinia  Eth 0
QH9ZQHC            Fas 0/1            219        R B H I r    yersinia  Eth 0
GLC4UKB            Fas 0/1            219        S H I r      yersinia  Eth 0
A2SJAJA            Fas 0/1            219        B            yersinia  Eth 0
WNE6WNS            Fas 0/1            219        T S I        yersinia  Eth 0
QHQH9H9            Fas 0/1            219        T S r        yersinia  Eth 0
OF7FOXO            Fas 0/1            219        R T H        yersinia  Eth 0
YPGKB3T            Fas 0/1            219        S            yersinia  Eth 0
--More--
```

Figure 5 - The output shows random bogus names are coming from the attacker's device running the Yersinia tool.

A second way to send multiple CDP frames is with the usage of Linux-based CDP Sender tool from the Phenoelit IRPAS package. The syntax to flood a device with CDP frames is as follows:

```
Linux#./cdp -i eth0 -n 100000 -l 1480 -r -v
```

These are the options in this command:

- -i - The interface on our computer that we want to send CDP frames out of. Typically, this is eth0.
- -n - The number of CDP frames we want to send. In this example, 100,000 frames are being sent.
- -l - The MTU size. For Ethernet networks, this should be set to 1480.
- -r - Randomize the device ID. Without this option, the router sees the same device identifier and ignores any subsequent frames after it receives the first frame.
- -v - This enables verbose output. This is optional.

In our case, we know the Layer 3 addresses, platforms, and IOS version of neighboring devices, and there is no need to collect these elements. Thus we can safely disable CDP on our routers and switches. The two commands we can use to disable CDP on our router are *no cdp run* which disables CDP globally on all interfaces of the router or *no cdp enable* which disables CDP on a particular interface.

Just in case, we are using CDP internally, then at a minimum we should disable it on the outbound interface. If we do not require CDP internally, we can safely disable it globally.

6.2 DHCP Starvation/Consumption Attack

The Dynamic Host Configuration Protocol (DHCP) is a network protocol used to configure devices that are connected to a network so they can communicate on that network using IPs. The protocol is implemented in a client-server model, in which DHCP clients request configuration data, such as an IP address, a default route, and one or more DNS server addresses from a DHCP server.

An example of use of the protocol is in a residential local area network (LAN). In this case, a DHCP server is contained in the router while the clients are hosts, e.g., personal computers, smart phones, or printers on the local network. The router itself is a client within the network of the Internet service provider (ISP) and receives its configuration information upstream from the ISP's DHCP server.

A DHCP server maintains a database of available IP addresses and configuration information. When the server receives a request from a client, the DHCP server determines the network to which the DHCP client is connected, and then allocates an IP address or prefix that is appropriate for the client, and sends configuration information appropriate for that client. DHCP servers typically grant IP addresses to clients only for a limited interval. DHCP clients are responsible for renewing their IP address before that interval has expired, and must stop using the address once the interval has expired, if they have not been able to renew it.

In case of a malicious user who tries to put his own DHCP Server into the local network for purposes of committing other types of attacks like Man in the Middle Attack, he must compromise the real DHCP Server and an effective way to do that is with the method of DHCP Starvation Attack.

Thus a DHCP client demands an IP address from the DHCP Server in order to operate as a part of the whole network where he belongs. His first step is to make a Discover looking for a DHCP Server. The point here is to prevent the real DHCP Server to respond and to do the next three steps of Offer, Request and Acknowledgement, with his system which intervenes to the handshake.

Below is depicted our topology in which the DHCP Starvation Attack was conducted.

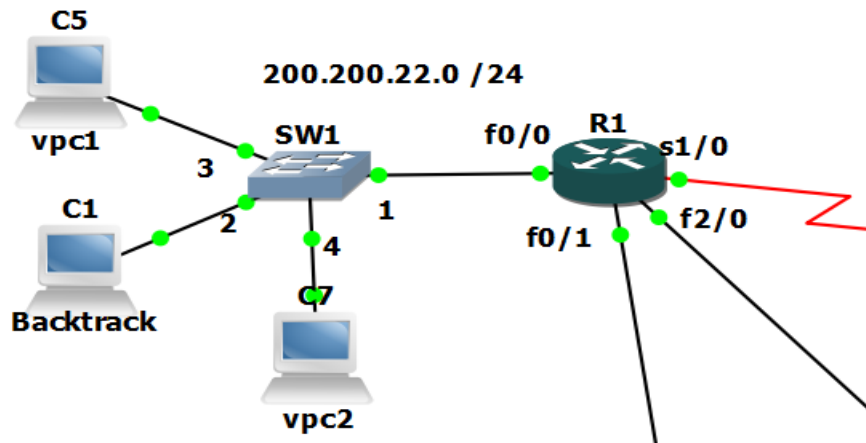


Figure 6 - The topology which used for DHCP Starvation Attack.

Initially, the attacker's pc which will launch the attack got the IP address of 200.200.22.2 from the DHCP pool2 of the router as shows the following figure.

```

root@bt:~# dhclient eth0
Internet Systems Consortium DHCP Client V3.1.3
Copyright 2004-2009 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/eth0/08:00:27:66:6d:c1
Sending on LPF/eth0/08:00:27:66:6d:c1
Sending on Socket/fallback
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 3
DHCPOFFER of 200.200.22.2 from 200.200.22.1
DHCPREQUEST of 200.200.22.2 on eth0 to 255.255.255.255 port 67
DHCPACK of 200.200.22.2 from 200.200.22.1
bound to 200.200.22.2 -- renewal in 40938 seconds.

```

Figure 7 – Requesting an IP address from the DHCP pool2.

In a local network the most common subnets we meet is the /24 which means 253 free IP address the very most, are available to be given. With the help of a Backtrack pc, we could issue as many Discovers are needed in order to deplete the available IP addresses of the real DHCP Server. Thus it couldn't answer to DHCP client requests.

We open the CLI on Backtrack system verifying some information about the Cisco router which acts also as a DHCP server in the local network. The command `show ip dhcp binding` displays address bindings on the Cisco IOS DHCP Server, which in the moment there is only one IP address leased out to the vpc1.

```

R1_DHCPserver#show ip dhcp binding
Bindings from all pools not associated with VRF:
IP address      Client-ID/      Lease expiration      Type
Hardware address/
User name
200.200.22.2    0800.2766.6dc1  Mar 02 2002 12:15 AM  Automatic

```

Figure 8 – The IP address that was given to the network device in the DHCP binding table.

The *show ip dhcp server statistics* command displays DHCP counters, where all of them are cumulative. The counters will be initialized, or set to zero, with the *clear ip dhcp server statistics* command.

```
R1_DHCPserver#show ip dhcp server statistics
Memory usage          24934
Address pools         2
Database agents       0
Automatic bindings    1
Manual bindings       0
Expired bindings      0
Malformed messages    0
Secure arp entries    0

Message                Received
BOOTREQUEST            0
DHCPDISCOVER           10
DHCPPREQUEST           3
DHCPCDECLINE           0
DHCPCRELEASE           0
DHCPCINFORM            0

Message                Sent
BOOTREPLY              0
DHCPOFFER              2
DHCPCACK               2
DHCPCNAK               0
```

Figure 9 – DHCP Server counters.

And if we want to see the content of the pools and the range that has been set, we could type the command *show ip dhcp pool*. The outcome at the following figure shows that the IP address range is 200.200.22.1 – 200.200.22.254 and the leased action is one.

```
R1_DHCPserver#show ip dhcp pool

Pool pool2 :
Utilization mark (high/low) : 100 / 0
Subnet size (first/next)    : 0 / 0
Total addresses              : 254
Leased addresses             : 1
Pending event                : none
1 subnet is currently in the pool :
Current index      IP address range      Leased addresses
200.200.22.3      200.200.22.1 - 200.200.22.254      1
```

Figure 10 – Information about DHCP pool2

Opening the command prompt on Backtrack, we type Yersinia -G to open the graphical interface of this tool. Then we choose the second tab DHCP and from the appearing window we select the choice “sending DISCOVER packet” from the second tab.

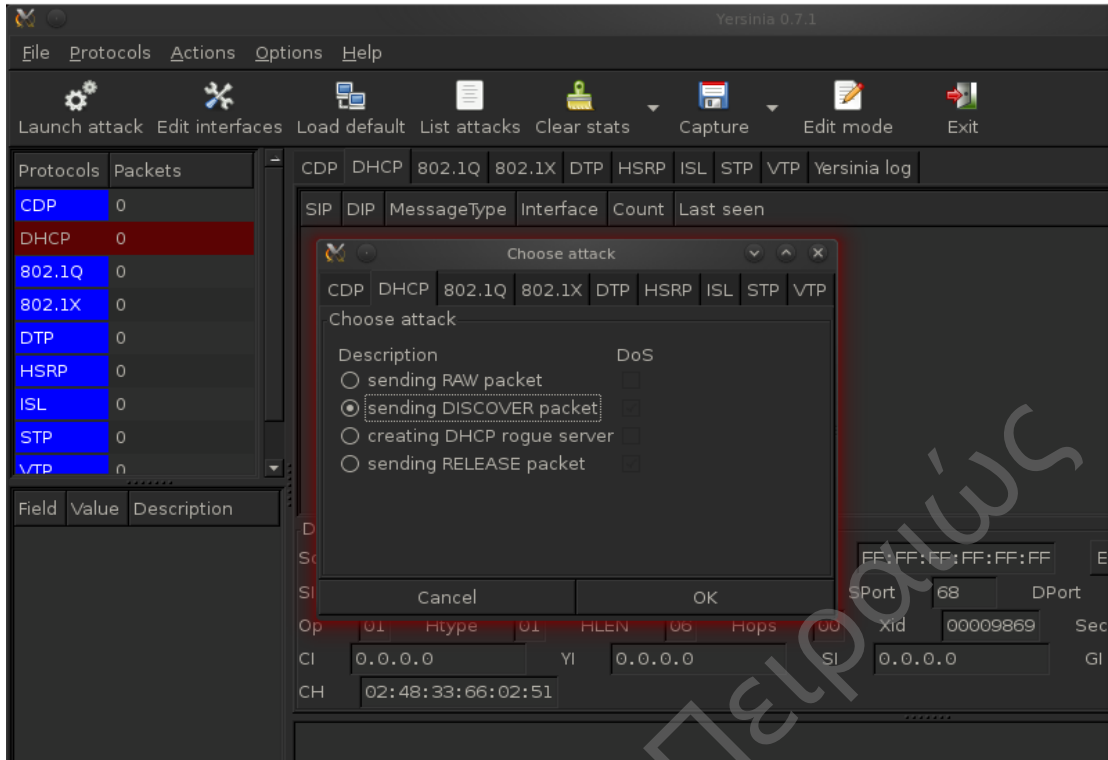


Figure 11 – Launching the attack with tool Yersinia.

Selecting OK, the attack starts.

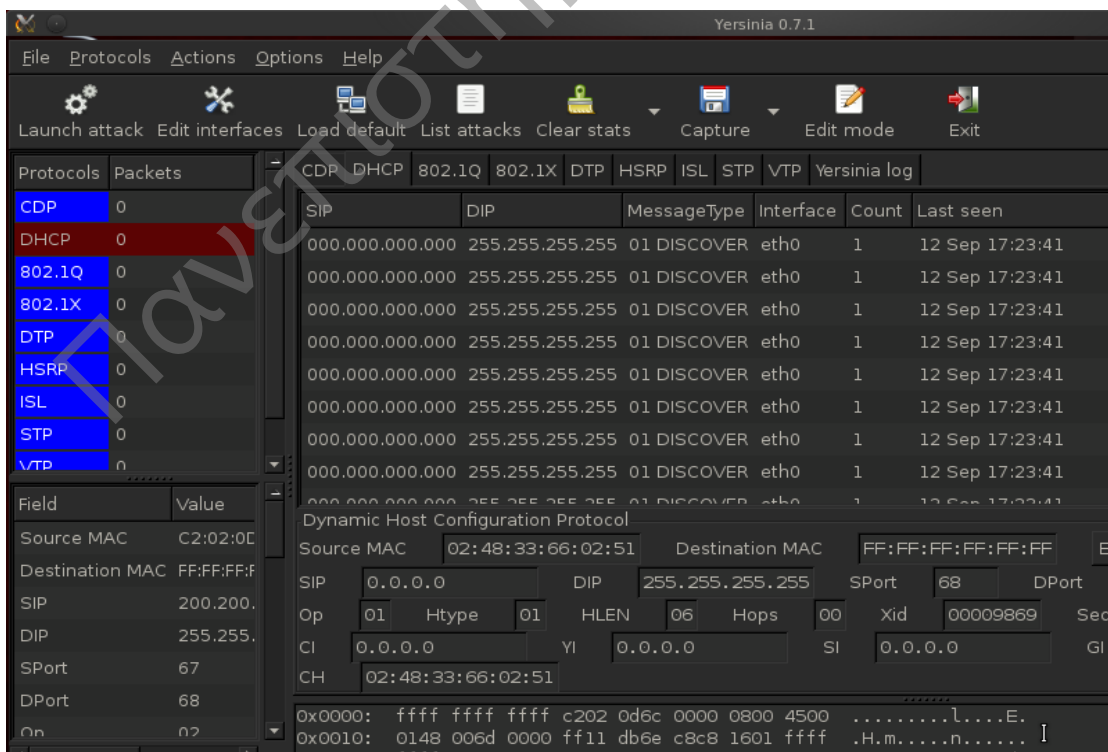


Figure 12 – The attack is running.

With the following results:

```
R1_DHCPserver#show ip dhcp binding
Bindings from all pools not associated with VRF:
IP address      Client-ID/
                Hardware address/
                User name
200.200.22.2    0800.2766.6dc1    Mar 02 2002 01:00 AM    Automatic
200.200.22.3    0f16.4343.04f4    Mar 01 2002 01:12 AM    Automatic
200.200.22.4    a5e8.da39.5c41    Mar 01 2002 01:13 AM    Automatic
200.200.22.5    47c2.6d11.da8b    Mar 01 2002 01:13 AM    Automatic
200.200.22.6    9c8c.2220.c9ca    Mar 01 2002 01:13 AM    Automatic
200.200.22.7    f2ed.7d0c.8dc0    Mar 01 2002 01:13 AM    Automatic
200.200.22.8    21b4.8a76.716a    Mar 01 2002 01:13 AM    Automatic
200.200.22.9    a5af.780d.8eb4    Mar 01 2002 01:13 AM    Automatic
200.200.22.10   d460.cc26.25cb    Mar 01 2002 01:13 AM    Automatic
200.200.22.11   d20e.8a45.8cb0    Mar 01 2002 01:13 AM    Automatic
200.200.22.12   8fff.4673.8aff    Mar 01 2002 01:13 AM    Automatic
200.200.22.13   e7aa.d702.d36c    Mar 01 2002 01:13 AM    Automatic
200.200.22.14   576f.e63d.605b    Mar 01 2002 01:13 AM    Automatic
200.200.22.15   c228.c82e.ec3f    Mar 01 2002 01:13 AM    Automatic
200.200.22.16   cfdd.480e.4004    Mar 01 2002 01:13 AM    Automatic
200.200.22.17   da20.6f46.1a84    Mar 01 2002 01:13 AM    Automatic
200.200.22.18   aa19.fd06.9c40    Mar 01 2002 01:13 AM    Automatic
200.200.22.19   1713.0865.2743    Mar 01 2002 01:13 AM    Automatic
200.200.22.20   ef4d.0934.d44b    Mar 01 2002 01:13 AM    Automatic
--More--
```

Figure 13 – The first IP addresses that were given for unreal users.

...

```
200.200.22.231  8e44.8969.7185    Mar 01 2002 01:20 AM    Automatic
200.200.22.232  e176.1352.6f49    Mar 01 2002 01:20 AM    Automatic
200.200.22.233  492d.7239.be04    Mar 01 2002 01:20 AM    Automatic
200.200.22.234  9e08.853b.8615    Mar 01 2002 01:20 AM    Automatic
200.200.22.235  a991.9067.c22e    Mar 01 2002 01:20 AM    Automatic
200.200.22.236  f95e.e613.ecdf    Mar 01 2002 01:20 AM    Automatic
200.200.22.237  1a72.f727.d3b4    Mar 01 2002 01:20 AM    Automatic
200.200.22.238  93c3.c318.66fd    Mar 01 2002 01:21 AM    Automatic
200.200.22.239  78e0.252c.6ab4    Mar 01 2002 01:21 AM    Automatic
200.200.22.240  0d8f.bf27.c3b3    Mar 01 2002 01:21 AM    Automatic
200.200.22.241  e105.b917.ee63    Mar 01 2002 01:21 AM    Automatic
200.200.22.242  a213.ad00.b23d    Mar 01 2002 01:21 AM    Automatic
200.200.22.243  e2a9.ea0f.f81c    Mar 01 2002 01:21 AM    Automatic
200.200.22.244  93ae.1824.e41a    Mar 01 2002 01:21 AM    Automatic
200.200.22.245  44f5.af73.85b2    Mar 01 2002 01:21 AM    Automatic
200.200.22.246  0ccc.5250.08e6    Mar 01 2002 01:21 AM    Automatic
200.200.22.247  07c4.6e15.3025    Mar 01 2002 01:21 AM    Automatic
200.200.22.248  eec4.5f17.4ba6    Mar 01 2002 01:21 AM    Automatic
200.200.22.249  c6cc.e85e.23f5    Mar 01 2002 01:21 AM    Automatic
200.200.22.250  cd66.9b7f.a0b0    Mar 01 2002 01:21 AM    Automatic
200.200.22.251  4c03.3964.64b3    Mar 01 2002 01:21 AM    Automatic
200.200.22.252  496a.5a11.0280    Mar 01 2002 01:21 AM    Automatic
200.200.22.253  a2a6.833f.0dfd    Mar 01 2002 01:21 AM    Automatic
200.200.22.254  74c2.8c70.3859    Mar 01 2002 01:21 AM    Automatic
R1_DHCPserver#
```

Figure 14 – The last IP addresses that were given for unreal users.

The DHCP pool of R1_DHCPserver filled up with IP and MAC addresses of bogus-unreal users from the fake DISCOVER requests that were launched by the Backtrack system.

```
R1_DHCPserver#show ip dhcp server statistics
Memory usage          29774
Address pools         2
Database agents       0
Automatic bindings   21
Manual bindings       0
Expired bindings      340
Malformed messages   0
Secure arp entries    0

Message               Received
BOOTREQUEST           0
DHCPDISCOVER          61583
DHCPREQUEST           1
DHCPDECLINE           0
DHCPRELEASE           0
DHCPIFORM             1

Message               Sent
BOOTREPLY              0
DHCPPOFFER            360
DHCPACK                1
DHCPNAK                0
```

Figure 15- DHCP Server counters were changed after the attack.

```
R1_DHCPserver#show ip dhcp pool pool2

Pool pool2 :
  Utilization mark (high/low)    : 100 / 0
  Subnet size (first/next)        : 0 / 0
  Total addresses                  : 254
  Leased addresses                 : 250
  Pending event                   : none
  1 subnet is currently in the pool :
  Current index      IP address range      Leased addresses
  200.200.22.19     200.200.22.1 - 200.200.22.254    250
R1_DHCPserver#
```

Figure 16 - Information about DHCP pool2 were changed after the attack.

Where the number of leased addresses is 250 and not 254 because of the default-gateway, the Backtrack system and the two other host-pcs in the subnet.

Clients of the victim network are then starved of the DHCP resources, thus DHCP Starvation can be classified as a Denial of Service attack. The attacker can then set up a Rogue DHCP Server on the network and perform man in the middle attacks, or simply set their machine as the default gateway and sniff packets.

Countermeasures

In order to protect the router from this kind of attack, we should implement restrictive rules on the Layer 2 device – SW1, which is in the middle of the router and clients.

As the topology shows the router R1-DHCPserver is connected to the port 1 on the switch, the other users are connected to ports 3 and 4, and the malicious user on port 2.

Thus the best way to confront this attack is to enable the features of Port-Security and DHCP Snooping from the Cisco IOS in every port of the switch SW1. So we can impose a strong restriction with Port-Security regarding the usage of MAC addresses from each client, setting them to three IP addresses available per port. In case of a violation action, restriction takes part in, and if this happens, then this port forces to turn in inactive mode for a specified time of five minutes. And DHCP Snooping security feature that uses the concept of trusted and untrusted ports and is capable of intercepting DHCP messages crossing a switch and blocking bogus DHCP Offers from clients. These security features described in more detail previously at the chapter of “Securing Layer 2 Technologies”.

6.3 VLAN Hopping - DTP and 802.1q Attack

In this scenario, conversion of unauthorized switch access ports to trunk (802.1q encapsulation) ports takes place through the Dynamic Trunking Protocol (DTP).

Let's suppose that we've got VLAN 10 and 20 which are associated with each own IP subnets and users. That's very likely what we're seeing in most networks is one VLAN per one subnet address. As so we have two groups of users, and we also have another VLAN 30 for a server or servers. We separate them into separate networks because of the big number of devices but a more likely scenario of why we divided them up is for security reasons.

If we have users on VLAN 10 and VLAN 20, and we want to control what ports are available to go between those VLANs and the server VLANs, we can use an access control list on a router. On router R1, we have configured sub-interfaces, one for each VLAN, in order to route between them. So when traffic needs to go between VLAN 10 and VLAN 30, the traffic would go over to its default gateway. The router would re-encapsulate the Layer 2 frames and send them over. At the same time, R1 has the opportunity to check access control lists to verify whether or not packets, based on Layer 4 protocols, or ports involved, or source and destination IP addresses, whether or not that traffic is allowed. So the router here is the security enforcer regarding what transit traffic is going to make it between the various VLANs.

Perhaps a malicious user at VLAN 10 wants to bypass the ACL on the router in order to reach the servers subnet. If the administrator hasn't secured the switches, and if the switches are in an attitude of using DTP to dynamically negotiate the port, whether it's an access port or a trunk port, he can run the Yersinia and modprobe802.1q tools from BackTrack, and establish a trunk mode connection between the host and the switch.

If it happened, his connection will be able to carry VLAN 20 and VLAN 30 traffic and then all he need to do on his BackTrack system is to load up the actual support for 802.1q encapsulation. That's the mechanism for tagging frames, for doing trunking, and creating some sub-interfaces on our BackTrack system. Thus, he can actually create interfaces for each VLAN 20 and VLAN 30, logical interfaces, and for every single VLAN that he wants. Before the unauthorized malicious action, once we have an interface on VLAN 30, we can then go ahead and communicate directly with the other devices on that VLAN and not have to go through the router. This is a really serious, serious problem, if somebody on our network is doing this, because they've just bypassed all of our local access control list security.

As far as the hardware is concerned we have catalyst of 3560 G switches, two hosts and one of them at VLAN 10 is running the BackTrack system, which has ethernet0 connected to port G0/3. We have also got a router R1 connected to the outside untrusted network (Internet) and one server at VLAN 30 on the right side of our topology as it is depicted below.

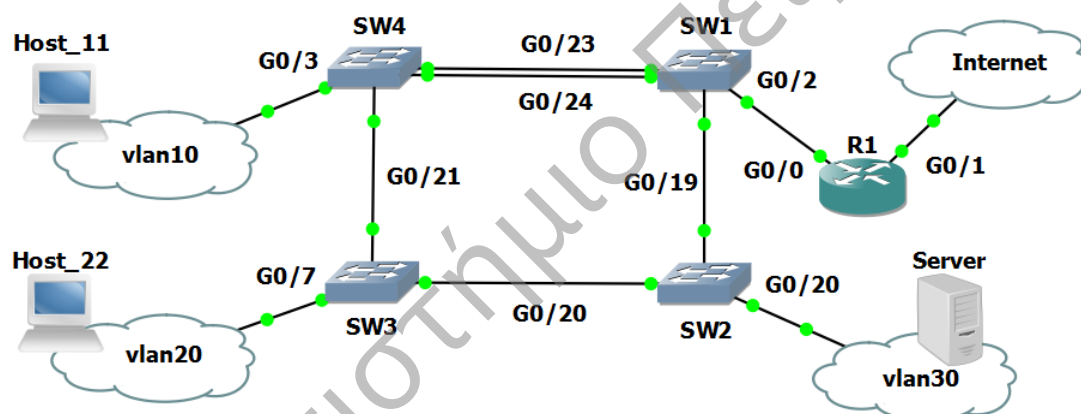


Figure 17 - The topology for DTP and 802.1q attack.

So the switch interface where is connected the malicious user is administratively configured as dynamic, meaning it will be willing to negotiate trunking. Dynamic has two options of desirable which means willing to be a trunk, and auto which means it will be a trunk in case somebody asks for this. It really doesn't matter if they're Desirable or Auto, because we're going to send DTP messages converting the interface to trunk. This is a mistake! We should never leave a port being willing on its own to determine if it will be a trunk, or an access port. We should control that and lock it down hard. In this case, it's not administratively locked down. It's set up as desirable, which a lot of switches are by default, which is a huge problem as well. And it's currently operating in access port mode. So it's an access port associated with VLAN 10, and that's currently how it is. However, if we send in a DTP frame, it will flip over and become a trunk port. Totally, we have got an IP address on VLAN 1, and an IP address each interface of VLAN 10,

20, and 30. Now, an attacker would not have direct access to this. However, it would be fairly simple to determine what those addresses are.

On the switch, we enable debugging about DTP events for confirmation. If we launch an attack, and then we can all see the direct reaction on the switch, that's just another confirmation that it's actually working. Again, if we're an attacker, we're not initially going to have access to the command line interface of the switch until we compromise it with the above mentioned tools.

At the Yersinia's GUI tool, we can observe that it's showing us what the attitude is of the switch and how many packets the DTP has got in. As the figure 18 below shows, it is currently set up as an access port, but he's Desirable, meaning he's willing to negotiate a trunk with us. All we need to do is to send down a DTP packet, and he'll flip over to trunk status.

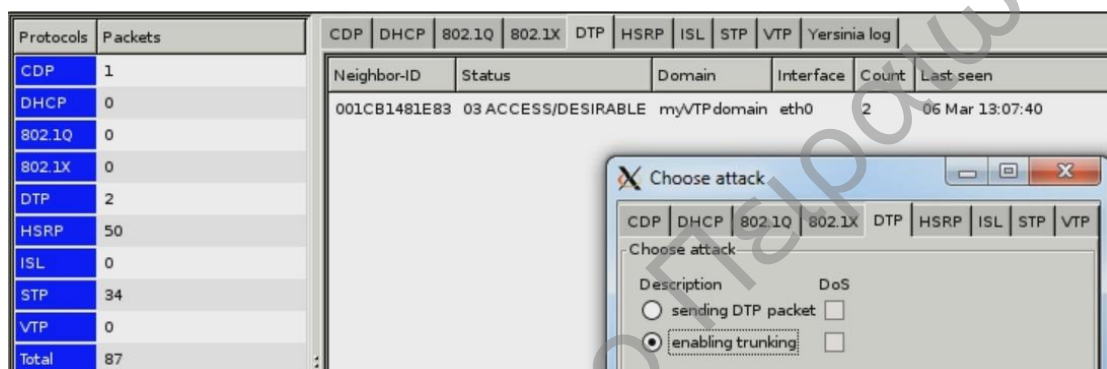


Figure 18 – The current status is Access/Desirable at the switch port and after that we launch the attack.

Thus we launch the DTP enable trunking attack, sending DTP packets to the G0/3 port of the switch.

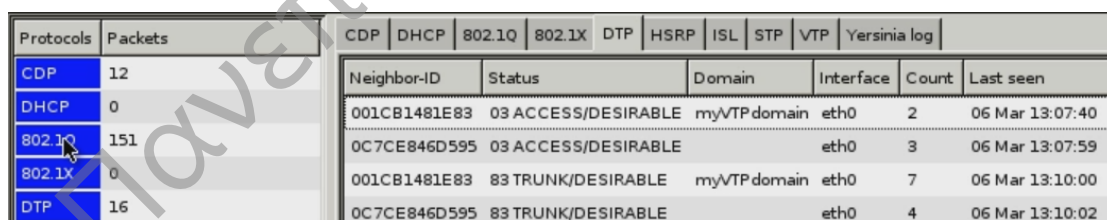


Figure 18 – After the attack the status of switch port has changed to Trunk/Desirable.

Due to debugging the switch console outputs some details regarding these messages being sent. We've just sent DTP and this switch has just converted that interface over to a trunk, mentioning also that G0/3 interface is a trunk using 802.1q encapsulation for packets and all VLANs (1,10,20,30) traffic are allowed to come through. Furthermore observing the Yersinia GUI tool we notice that our 802.1q traffic is now increasing. That's happening because every frame it sees that now has the 802.1q tag, it's processing. So by selecting to 802.1q tab, as the

picture below depicts, this shows what it knows about VLAN 10, VLAN 30, VLAN 20 and that's how we could know what VLANs are out there.

Protocols	Packets
CDP	13
DHCP	0
802.1Q	166
802.1X	0
DTP	16
HSRP	51
ISL	0
STP	106
VTP	1
Total	353

VLAN	L2Proto1	Src IP	Dst IP	IP Prot	Interface	Count	Last seen
0010	0800 IP	10.0.0.161	224.0.0.2	17 UKN	eth0	37	06 Mar 13:10:32
0010	0800 IP	10.0.0.1	224.0.0.2	17 UKN	eth0	1	06 Mar 13:09:11
0030	0800 IP	30.0.0.161	224.0.0.2	17 UKN	eth0	37	06 Mar 13:10:31
0030	0806 ARP	30.0.0.50		UKN	eth0	2	06 Mar 13:08:57
0010	0800 IP	10.0.0.1	224.0.0.2	17 UKN	eth0	40	06 Mar 13:10:30
0020	0806 ARP	20.0.0.50		UKN	eth0	2	06 Mar 13:08:57
0010	0800 IP	10.0.0.1	224.0.0.2	17 UKN	eth0	2	06 Mar 13:10:32
0020	0800 IP	20.0.0.161	224.0.0.2	17 UKN	eth0	37	06 Mar 13:10:31
0010	0800 IP	10.0.0.1	224.0.0.2	17 UKN	eth0	1	06 Mar 13:10:06
0020	0806 ARP	20.0.0.50		UKN	eth0	1	06 Mar 13:08:54

Figure 19 – We notice that our 802.1q traffic is now increasing.

So we have the source IP addresses of devices which are the switch's and router's addresses 10.0.0.161 and 10.0.0.1 respectively and virtual address (30.0.0.50, 20.0.0.50, 10.0.0.50) advertised by HSRP.

Now our challenge is how do we get ourselves onto any VLAN that we want to? All we have to do is create logical sub-interfaces and set it to do trunking on those interfaces. And we can logically be on any one of those VLANs, and assign ourselves an appropriate IP address based on the subnet information. So for VLAN 20, they're using the 20 network, for VLAN 30, they're using the 30 network.

At the BackTrack system we have to turn on support for 802.1q tagging by issuing the command "modprobe 8021q". We also have to create new logical sub-interfaces that are responding and appropriate for each of those VLANs. That can happen by issuing the command "vconfig add eth0 20" to create a logical interface of ethernet 0, and 20 the number of VLAN the 802.1q tag of 20, and then the command "ifconfig eth0.20 20.0.0.99/24" to bring it up assigning an IP address. If we wanted to find out everybody who's currently on that network, one approach of doing that is doing a broadcast ping which will ping everybody on the 20.0.0.0 network. Anybody who responds, we know they're on that network segment. We also follow exact the same process for VLAN 30, doing broadcast ping of the 30 network and we have 30.0.0.161 that's our multilayer switch also responding to that.

Another option, if we don't want to create sub-interfaces and generate traffic with a trunk up, we can still hear quite a bit of traffic for each of those networks. Any broadcasts, for example, or unknowns on any of those VLANs are going to be

forwarded through the trunk. So if we wanted to capture that, there's two ways of doing it. The first, we could just start Wireshark and that would show us everything on that interface. Or the second in Yersinia we could use the Capture - All Protocols, option and that's yet another way to capture that information. At Wireshark we captured all that is coming in on the trunked interface, of the BackTrack system. But if we look at any one of these packets and open it up, we will see that it has the 802.1q header in it, and it's tagged. Let's take a look at one from VLAN 10, and indeed, it is tagged with VLAN 10, as the figure 19 shows below.

No.	Source	Destination	Protocol	Info
1	Cisco_48:1e:83	Spanning-tree-(for-bridge	STP	RST. Root = 32768/1/00:1c:b1:48:1e:80 Cost =
2	30.0.0.161	224.0.0.2	HSRP	Hello (state Speak)
3	30.0.0.161	224.0.0.2	HSRP	Hello (state Standby)
4	10.0.0.161	224.0.0.2	HSRP	Hello (state Active)
5	20.0.0.161	224.0.0.2	HSRP	Hello (state Standby)
6	Cisco_48:1e:83	Spanning-tree-(for-bridge	STP	RST. Root = 32768/1/00:1c:b1:48:1e:80 Cost =
7	30.0.0.161	224.0.0.2	HSRP	Advertise (state Active)
8	30.0.0.161	224.0.0.2	HSRP	Hello (state Active)
9	All-HSRP-routers_00	Broadcast	ARP	Gratuitous ARP for 30.0.0.50 (Reply)
10	All-HSRP-routers_00	STP-InlinkFast	ARP	Gratuitous ARP for 30.0.0.50 (Reply)


```

Frame 4: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
Ethernet II, Src: All-HSRP-routers_00 (00:00:0c:07:ac:00), Dst: IPv4mcast_00:00:02 (01:00:5e:00:00:02)
802.1Q Virtual LAN, PRI: 6, CFI: 0, ID: 10
  110. .... = Priority: Voice, 10ms latency and jitter (6)
  ...0 .... = CFI: Canonical (0)
  .... 0000 0000 1010 = ID: 10
  Type: IP (0x0800)
Internet Protocol Version 4, Src: 10.0.0.161 (10.0.0.161), Dst: 224.0.0.2 (224.0.0.2)
  
```

Figure 19 – Wireshark capturing trunked traffic.

So effectively, by creating our sub-interfaces, we can join anyone of those VLANs and logically be directly connected to each and every one that shows up on that trunk.

In this scenario, we presented at how we can trick an access port on a switch to become a trunk, simply by manipulating dynamic trunking protocol messages. And then once we have a trunk established, we can then go ahead and create sub-interfaces and have direct connectivity to any of the VLANs on that trunk.

Countermeasures

In order to prevent or mitigate this type of attack we should apply some basic switching security features. First of all, we should not let our ports negotiate at all. If they're going to be an access port, we have to hard code them. If they're going to be a trunk port, we have to hard code them and to turn off DTP. Turning off the whole negotiation process no matter what DTP shows up, the switch don't pay attention. Another good common security practice for switches is to take all the ports that are currently not in use, and do two things. Initially, we must shut them down and after that we must assign them to an unused VLAN, like VLAN 999, something that's not used in our environment. If somebody does get physical access to ports on the switch, they are not going to lead to anything useful for them.

6.4 MAC/ARP Spoofing Attack using Ettercap and Xplico Tools

In this scenario, we will implement a Man-in-the-Middle attack using the tools Ettercap and Xplico from Backtrack. This type of attack has to do with a malicious user or a hacker who puts his machine in the logical way between two machines communicating together. Having this position, he could launch a lot of different dangerous attacks because he is in the way between two normal machines.

Ettercap is a tool for Man-in-the-Middle attacks on a LAN. It is able to perform attacks against the ARP protocol by positioning itself as "man in the middle" and, once positioned as this, it is able to infect, replace, and delete data in a connection, discover passwords for protocols such as FTP, HTTP, POP, SSHv1, and provide fake SSL certificates in HTTPS sections to the victims.

There are several kinds of attacks to become "man in the middle" based on the ARP protocol.

The ARP protocol is a layer 3 protocol used to translate IP addresses to physical network card addresses or MAC addresses. When a device tries to access a network resource, it will first send requests to other devices asking for the MAC address associated with the IP it wants to reach. The caller will keep the IP - MAC association in its cache, the ARP cache, to speed up new connections to the same IP address.

The attack comes when a machine asks the other ones to find the MAC address associated with an IP address. The attacker will answer to the caller with fake packets saying that the IP address is associated to its own MAC address and in this way, will "short-cut" the real IP - MAC association answer coming from another host. This attack is referred as ARP poisoning or ARP spoofing and is possible only if the attacker and the victims are inside the same broadcast domain which is defined on the host by an IP address and a Subnet mask, for example: 192.168.10.1 255.255.255.0.

In our scenario, we will use the case study below where a machine with IP 192.168.1.2 reaches Internet resources from a local network. After the ARP poisoning attack, the PC which runs Ettercap tool with IP 192.168.1.100 is set as "man in the middle".

About Ettercap's PC behavior we must mention that every time Ettercap starts, it disables IP forwarding in the kernel and begins to forward packets itself. It usually slows down the network performances between the two hosts because of the packets' machine process time. Ettercap also needs root privileges to open the Link Layer sockets. After the initialization phase, the root privileges are not needed anymore, so Ettercap drops them to UID = 65535 (nobody). Since Ettercap has to create log files, it must be executed in a directory with the right permissions.

Xplico is a network forensics analysis tool (NFAT), which is a software that reconstructs the contents of acquisitions performed with a packet sniffer such as Wireshark, Ettercap, tcpdump and Netsniff-ng. Unlike the protocol analyzer, whose main characteristic is not the reconstruction of the data carried by the protocols, Xplico born expressly with the aim to reconstruct the protocol's

application data and it is able to recognize the protocols with a technique named Port Independent Protocol Identification (PIPI). To clarify what Xplico does, we can imagine to have the raw data (Ethernet or PPP) of a web navigation (HTTP protocol), which in this case Xplico is able to extract and reconstruct all the Web pages and contents (images, files, cookies, and so on). Similarly, Xplico is able to reconstruct the e-mail exchanged with the IMAP, POP and SMTP protocols. Among the protocols that Xplico identifies and reconstructs there are VoIP, MSN, IRC, HTTP, IMAP, POP, SMTP and FTP.

The Xplico's architecture provides an input module to handle data input (from probes or packet sniffer), an output module to organize the decoded data presenting them to the end user, and a set of decoding modules, called protocol dissector for the decoding of the individual network protocol. With the output, module Xplico can have different user interfaces, in fact it can be used from command line and from a web user interface called "Xplico Interface". The protocol dissector is the modules for the decoding of the individual protocol, each protocol dissector can reconstruct and extract the data of the protocol. All modules are plug-in and, through the configuration file, they can be loaded or not during execution of the program. This allows to focus the decoding, that is, if we want to decode only VoIP calls but not the Web traffic then we configure Xplico to load only the RTP and SIP modules excluding the HTTP module. [7]

Another feature of Xplico is its ability to process (reconstruct) huge amounts of data, it is able to manage pcap (packet capture) files of many Gbs and also Tbs and from multiple capture probes simultaneously, this thanks to the use of various types of "input modules". The pcap files can be uploaded in many ways, directly from the Xplico Web user interface or with a SFTP or with a transmission channel called PCAP-over-IP. For this features Xplico is used in contexts of Lawful interception and in Network Forensics. [8] Xplico and also its specific version called pcap2wav is able to decode VoIP calls based on the RTP protocol (SIP, H323, MGCP, SKINNY) and supports the decode of audio codecs G711ulaw, G711alaw, G722, G729, G723, G726 and MSRTA (Microsoft's Real-time audio). [9]

Topology

At the depicted left part of the network topology below there are four PCs which belong to the subnet of 10.0.10.0 /24. At this chunk, we will implement a Man-in-the-Middle attack with the attacker at Backtrack-PC (IP: 10.0.10.7), between the default gateway of the router (IP: 10.0.10.1) and the victim at Ubuntu-PC (IP: 10.0.10.9).

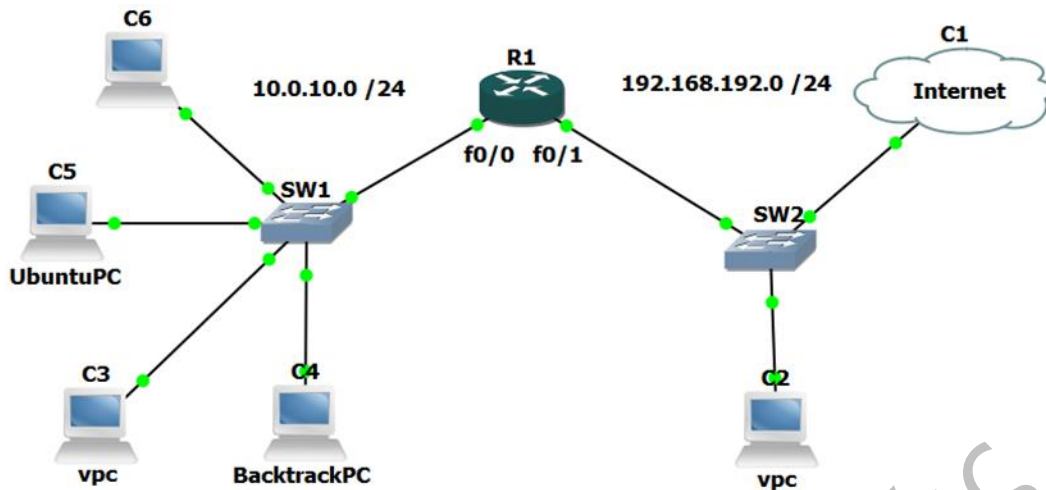


Figure 20 – Network Topology for the MitM scenario.

Initially, the attacker will do an ARP poison for the two victims with the help of Ettercap program, and then we will launch the Xplico program in order to capture the transferring data which will pass from the attacker's PC. Launching Ettercap we select the option "Unified Sniffing" from the category "Sniff", and from the appearing window we specify from what network interface the Backtrack-PC will sniff, and in our case is "eth0". After that we must scan all the hosts who belong to this part of the network. Thus we select the option "Scan for Hosts" from the category "Hosts". After the loading we select the option "Host list" from the category "Hosts" again, in order to give us the list.

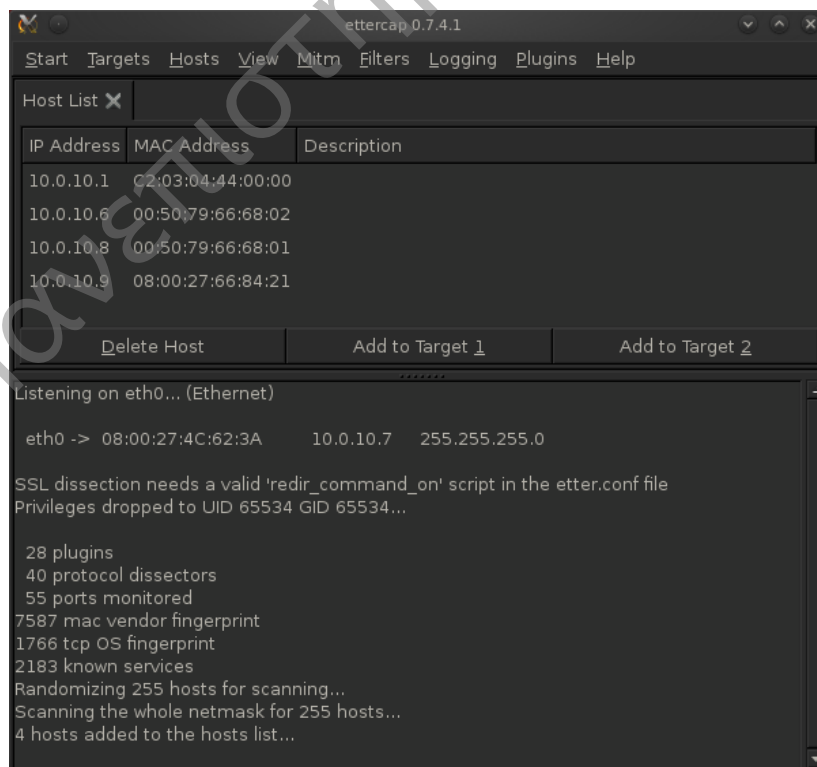


Figure 21 – Ettercap's results for hosts in the network.

Having the list, we select the first line of 10.0.10.1 C2:00:0D:D0:00:00 (router's default gateway IP and MAC addresses) as the first target with the button "Add to Target 1". We also select the second line of 10.0.10.9 08:00:27:66:84:21 (Ubuntu-PC's IP and MAC addresses) as the second target with the button "Add to Target 2".

After the definition of targets and the launch of the attack what will happen is ARP spoofs, poisoning the ARP caches for both of two devices. The ARP poisoning will provide to victims that the MAC address of the other one is the MAC address of the Backtrack-PC.

To turn the attack on we select the "Arp poisoning" from the category "Mitm" on toolbar the appearing window we click OK.

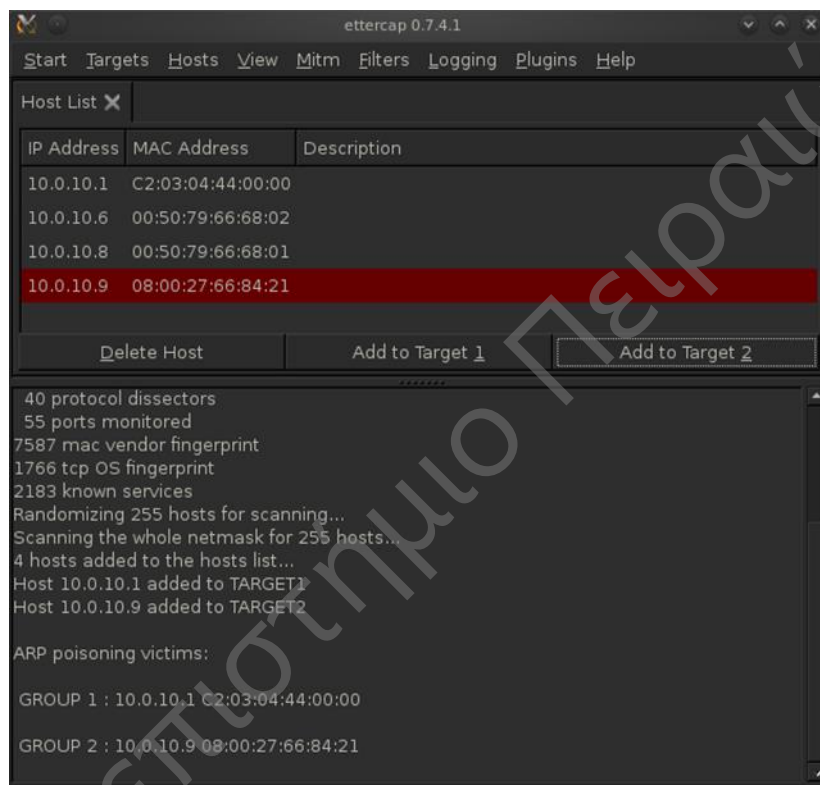


Figure 22 – Activating the ARP spoofing attack.

To verify that we check the MAC addresses at the CLI on any involved device below.

```
root@bt:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:4c:62:3a
          inet addr:10.0.10.7  Bcast:10.0.10.255  Mask:255.255.255.0
```

Figure 23 - Checking the MAC address of BackTrack-PC.


```
jb@jb-VirtualBox:~$ arp -a
? (10.0.10.1) at 08:00:27:4c:62:3a [ether] on eth0
```

Figure 24 - Checking the MAC address of Ubuntu-PC.

```
R1(config)#do show arp
Protocol  Address          Age (min)  Hardware Addr  Type   Interface
Internet  10.0.10.1        -          c203.0444.0000 ARPA   FastEthernet0/0
Internet  10.0.10.6        14         0050.7966.6802 ARPA   FastEthernet0/0
Internet  10.0.10.7        6          0800.274c.623a ARPA   FastEthernet0/0
Internet  10.0.10.8        13         0050.7966.6801 ARPA   FastEthernet0/0
Internet  10.0.10.9        0          0800.274c.623a ARPA   FastEthernet0/0
Internet  192.168.192.2    10         0050.56f4.ef8f ARPA   FastEthernet0/1
Internet  192.168.192.130  1          020c.29c2.115c ARPA   FastEthernet0/1
Internet  192.168.192.136  13         0050.7966.6800 ARPA   FastEthernet0/1
Internet  192.168.192.141 -          c203.0444.0001 ARPA   FastEthernet0/1
Internet  192.168.192.254 207        0050.56e5.b157 ARPA   FastEthernet0/1
```

Figure 25 - Checking the MAC addresses at the router's CLI.

From attacker's Backtrack-PC we launch the tool Xplico from the path:

 - Backtrack - Forensics - Network Forensics - xplico web gui.

It launches a web server, starting Apache2 server.

```
root: sh <2>
File Edit View Bookmarks Settings Help
Module php5 already enabled
Enabling module rewrite.
Run '/etc/init.d/apache2 restart' to activate new configuration!
Enabling site xplico.
Run '/etc/init.d/apache2 reload' to activate new configuration!
httpd not running, trying to start
 * Enabling additional executable binary formats binfmt-support [ OK ]
root
 * Starting Xplico offline mode
Modifying priority to -1 [ OK ]

----- XPLICO GUI -----

WARNING: Apache2 server started:
You will have to stop it manually.

XPLICO WEB GUI:
http://localhost:9876/

----- XPLICO GUI -----
root@bt:~#
```

Figure 26 - Starting Xplico tool.

In order to start the Xplico web graphical user interface, we should open a browser typing the address <http://localhost:9876/>. We choose to open Firefox web browser and at the first appeared page, we log in with the same username and password, "xplico". After login, we can capture the data, which are passing through the interface eth0 of Backtrack-PC, with many different ways.

Firstly, we create a new case, selecting the option "Live Acquisition", naming this as "MaliciousAction" and then clicking "Create". Thus we created a new case, as the following screenshot shows.

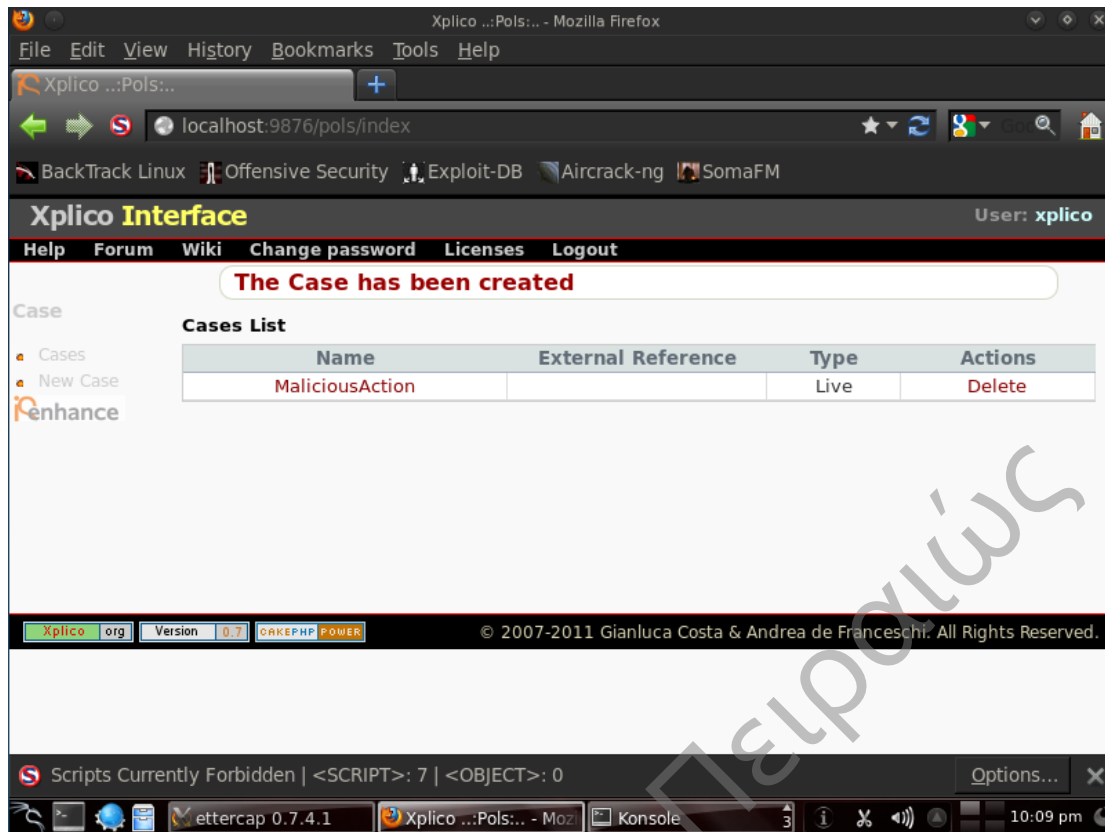


Figure 27 – Creating a new case.

Secondly, we click the name “MaliciousAction” and after that we select “New Session” in that case, naming it “Session1” and clicking “Create”. Now, we have the following screenshot.



Figure 28 – Creating a new session.

Clicking on “Session1”, the following page full of options and functions appears.

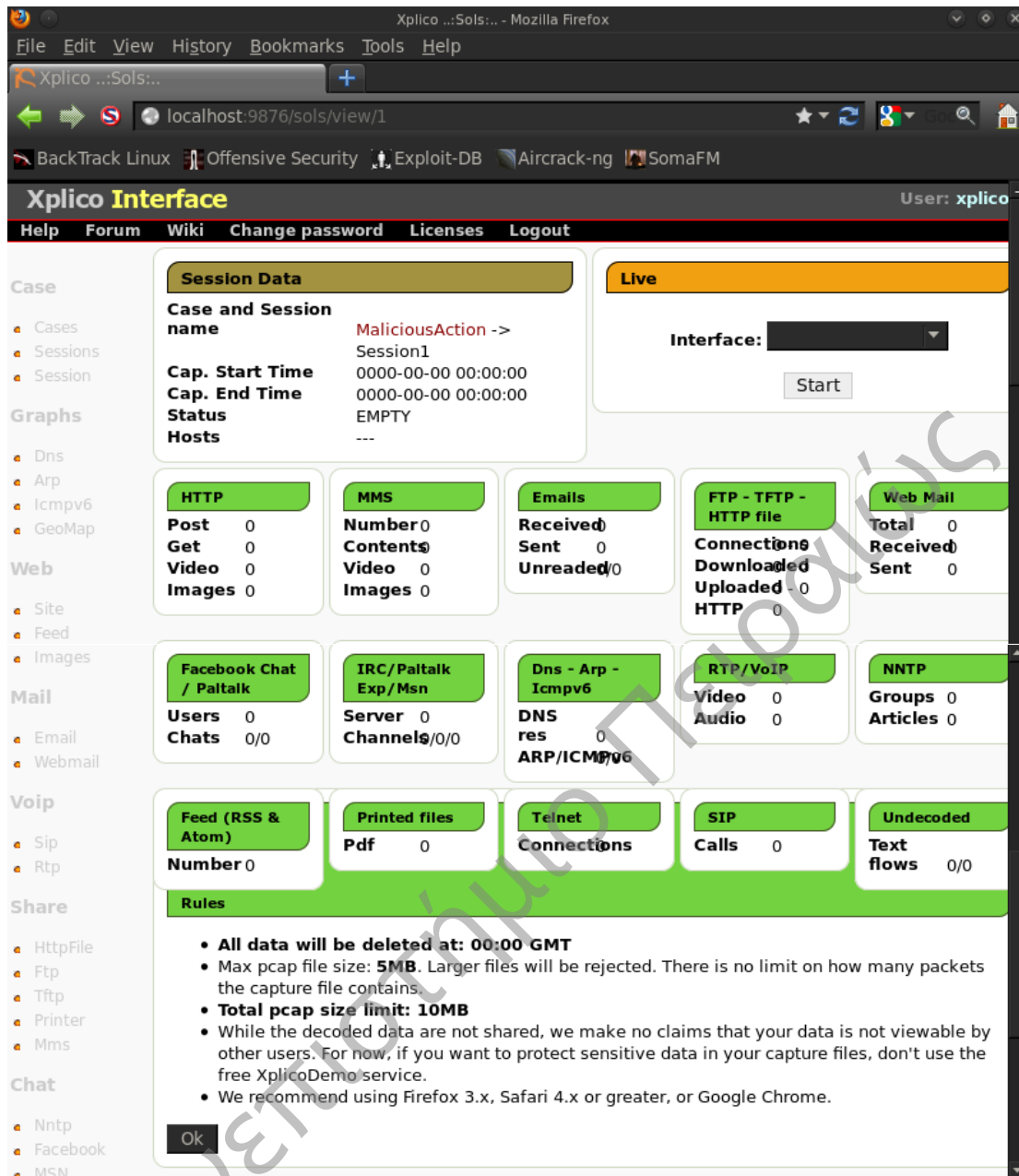


Figure 29 – Options and functions of the Session1.

Selecting interface eth0 and clicking “Start”, Xplico starts the capture everything passing through that interface.

From Ubuntu-PC we do an ordinary search for something like “computer network technology”. As we browse pages, images, videos, Xplico monitors our movements having the following results.

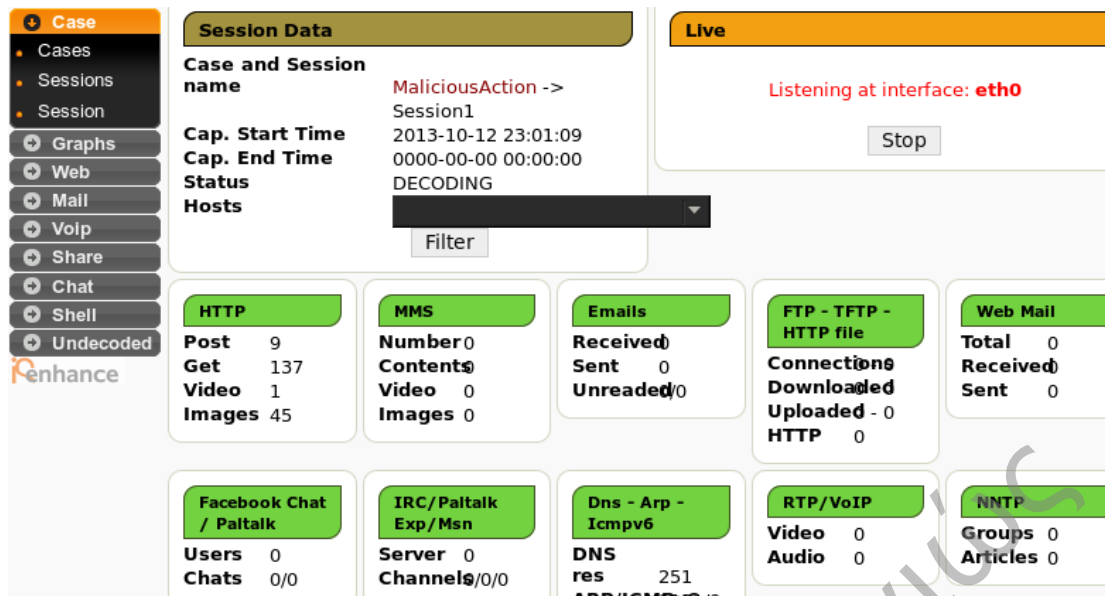


Figure 30 – General statistics on case “MaliciousAction - Session1”.

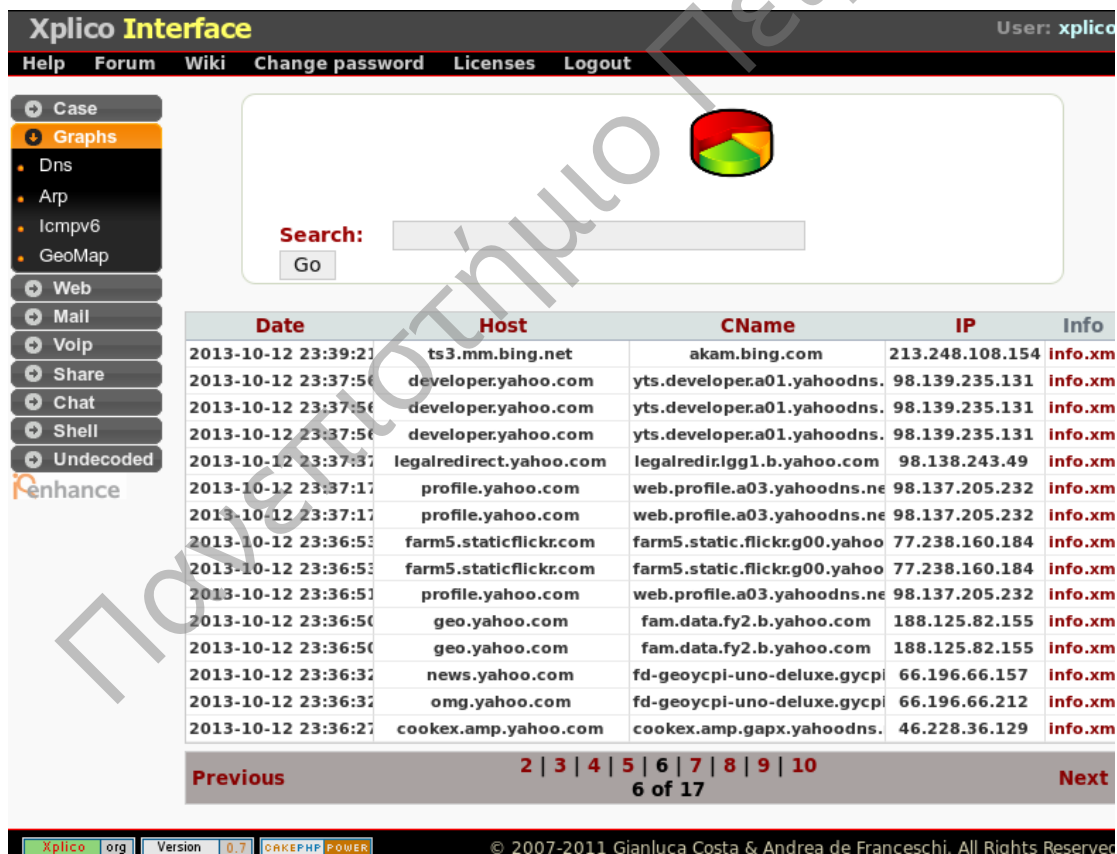


Figure 31 – DNS statistics from our browsing.

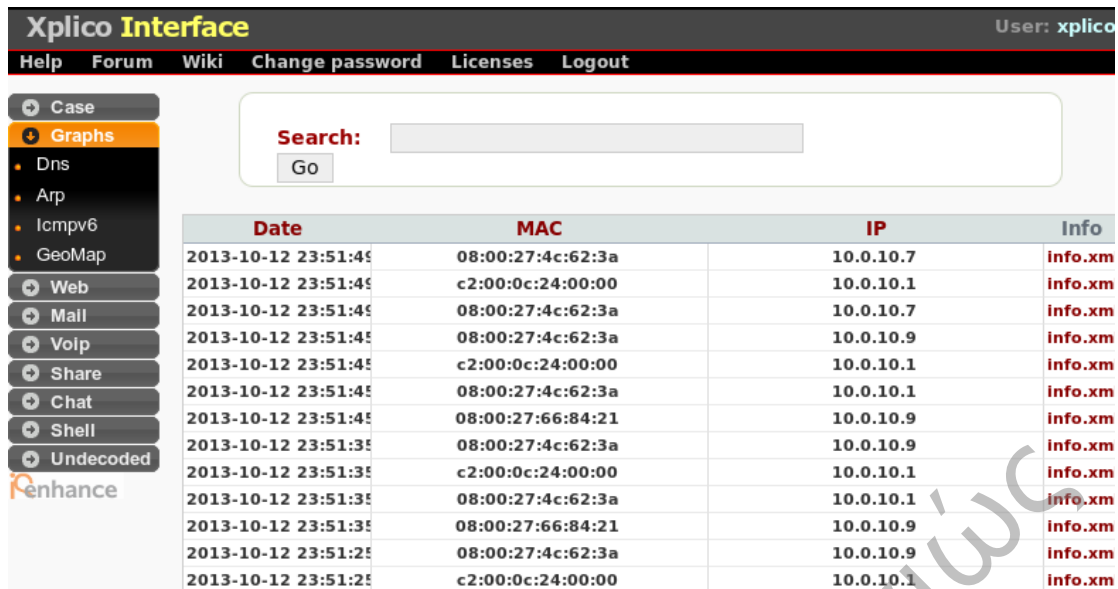


Figure 31 – ARP statistics from our browsing.



Figure 32 – Visited sites during browsing.



Figure 33 – Captured pictures during browsing.

Countermeasures

Dynamic ARP Inspection (DAI) is the most efficient security feature to confront this type of attack, because it intercepts and verifies IP-to-MAC address bindings and discards invalid ARP packets. DAI uses the DHCP snooping database to validate bindings. It associates a trust state with each interface on the switch or the router. Packets arriving on trusted interfaces bypass all DAI validation checks, and those arriving on untrusted interfaces undergo the DAI validation process. In a typical network, all ports on the switch connected to host are configured as untrusted, and switch ports are considered trusted. When DAI is configured, the device rate-limits incoming ARP packets to prevent DoS attacks. The default rate for an untrusted interface is 14 packets per second. Trusted interfaces are not rate-limited. DAI also uses the ARP access control lists (ACLs) and DHCP snooping database for the list of valid IP-to-MAC address bindings. ARP ACLs take precedence over entries in the DHCP snooping bindings database but have to be manually configured. The switch will drop a packet that is denied in the ARP ACLs even if the DHCP snooping database has a valid binding for it. When a switch or a router drops a packet, it is logged in the buffer and generates a system message.

6.5 ARP Spoofing Attack using Nmap, Arpspoof, Driftnet, Urlsnarf and Wireshark Tools

ARP spoofing is a technique whereby an attacker sends fake ("spoofed") Address Resolution Protocol (ARP) messages onto a Local Area Network. Generally, the aim is to associate the attacker's MAC address with the IP address of another host (such as the default gateway), causing any traffic meant for that IP address to be sent to the attacker instead.

ARP spoofing may allow an attacker to intercept data frames on a LAN, modify the traffic, or stop the traffic altogether. Often the attack is used as an opening for other attacks, such as denial of service, man in the middle, or session hijacking attacks.

Vulnerabilities of the Address Resolution Protocol

The Address Resolution Protocol (ARP) is a widely used protocol for resolving network layer addresses into link layer addresses. When an Internet Protocol (IP) datagram is sent from one host to another on a local area network, the destination IP address must be converted into a MAC address for transmission via the data link layer. When another host's IP address is known, and its MAC address is needed, a broadcast packet is sent out on the local network. This packet is known as an ARP request. The destination machine with the IP in the ARP request then responds with an ARP reply, which contains the MAC address for that IP.

ARP is a stateless protocol. Network hosts will automatically cache any ARP replies they receive, regardless of whether or not they requested them. Even ARP entries which have not yet expired will be overwritten when a new ARP reply packet is received. There is no method in the ARP protocol by which a host can authenticate the peer from which the packet originated. This behavior is the vulnerability which allows ARP spoofing to occur.

Anatomy of an ARP spoofing attack

The basic principle behind ARP spoofing is to exploit the above mentioned vulnerabilities in the ARP protocol by sending spoofed ARP messages onto the LAN. ARP spoofing attacks can be run from a compromised host on the LAN or from an attacker's machine that is connected directly to the target LAN.

Generally, the goal of this attack is to associate the attacker's MAC address with the IP address of a target host, so that any traffic meant for the target host will be sent to the attacker's MAC instead. The attacker could then choose to: a. Inspect the packets, and forward the traffic to the actual default gateway (interception) affecting confidentiality, b. Modify the data before forwarding it (man-in-the-middle attack) violating their integrity, and c. Launch a denial-of-service attack by causing some or all of the packets on the network to be dropped abolishing availability.

Defenses

Static ARP entries

IP-to-MAC mappings in the local ARP cache can be statically defined, and then hosts can be directed to ignore all ARP reply packets. While static entries provide perfect security against spoofing if the operating systems handle them correctly, they result in quadratic maintenance efforts as IP-MAC mappings of all machines in the network have to be distributed to all other machines.

ARP spoofing detection software

Software that detects ARP spoofing generally relies on some form of certification or cross-checking of ARP responses. Uncertified ARP responses are then blocked. These techniques may be integrated with the DHCP server so that both dynamic and static IP addresses are certified. This capability may be implemented in individual hosts or may be integrated into Ethernet switches or other network equipment. The existence of multiple IP addresses associated with a single MAC address may indicate an ARP spoof attack, although there are legitimate uses of such a configuration. In a more passive approach a device listens for ARP replies on a network, and sends a notification via email when an ARP entry changes.

OS security

Operating systems react differently, e.g. Linux ignores unsolicited replies, but on the other hand uses seen requests from other machines to update its cache. Solaris accepts updates on entries only after a timeout. In Microsoft Windows, the behavior of the ARP cache can be configured through several registry entries under `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters,` `ArpCacheLife,` `ArpCacheMinReferenceLife,` `ArpUseEtherSNAP,` `ArpTRSingleRoute,` `ArpAlwaysSourceRoute,` `ArpRetryCount.`

AntiARP also provides Windows-based spoofing prevention at the kernel level. ArpStar is a Linux module for kernel 2.6 and Linksys routers that drops invalid packets that violate mapping, and contains an option to re-poison/heal.

The simplest form of certification is the use of static, read-only entries for critical services in the ARP cache of a host. This prevents only simple attacks and does not scale on a large network, since the mapping has to be set for each pair of machines resulting in n^2 ARP caches that have to be configured.

Topology

At the following topology, we examine the results of an ARP spoofing attack.

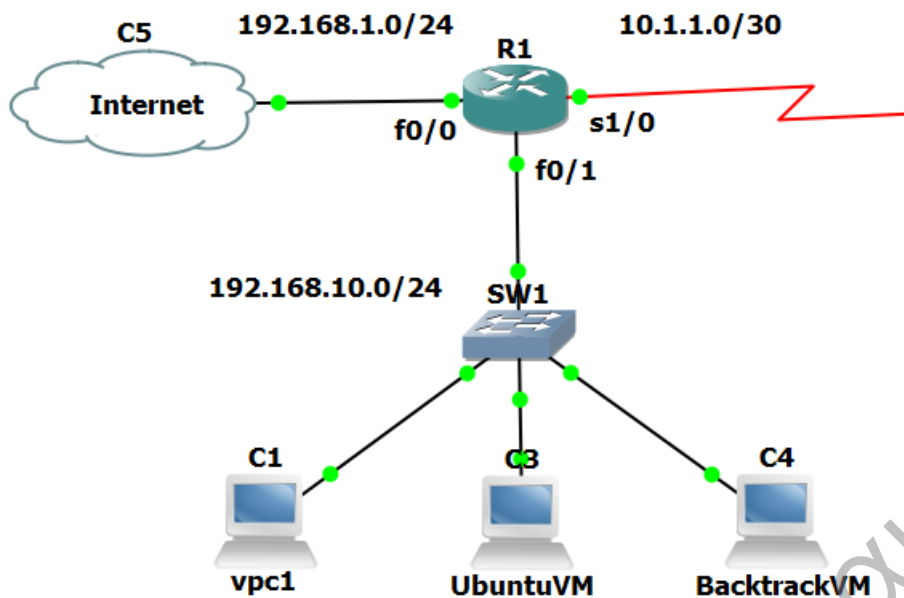


Figure 34 – The topology in which ARP spoofing attack took place.

At this part of the topology there are three users on the same subnet 192.168.10.0 /24. The third on the right side is a malicious user working on the Backtrack- PC and he will attack to the Ubuntu-PC user doing ARP spoofing. Achieving this, he will be the Man-in-the-Middle eavesdropping everything between the user and the router. There is also the case of interrupting the communication between the two devices causing Denial-of-Service, isolating the user, but in this section we won't occupy with this kind of attack.

In normal conditions, Ubuntu user (IP address:192.168.10.6) wants to forward traffic to its default gateway (R1 Router with IP address:192.168.10.1) he needs to know the Ethernet layer 2 address. In order to find out this, he uses the ARP protocol. So he sends an ARP request – broadcast message to everyone on the subnet requesting the MAC address of the device which has the 192.168.10.1 IP address. Router responds sending back his MAC address and the Ubuntu-PC stores this into his MAC-address table. The same function is being done with any other device in the subnet.

The real MAC addresses of the three devices are depicted below.

```
R1(config)#do show arp
Protocol  Address      Age (min)  Hardware Addr  Type   Interface
Internet  192.168.10.1  -          c200.0a74.0001  ARPA   FastEthernet0/1
Internet  192.168.10.2  0          0800.2766.6dc1  ARPA   FastEthernet0/1
Internet  192.168.10.6  15         0800.272a.0603  ARPA   FastEthernet0/1
```

Figure 35 – Device's MAC Addresses

The *show arp* command at R1's CLI shows the IP addresses and MAC addresses of the connected devices.

More specifically the real IP and MAC addresses of devices before the attack are depicted below.

NAME	IP/CIDR	GATEWAY	MAC	LPORT	RPORT
UPCS1	192.168.10.5/24	192.168.10.1	00:50:79:66:68:00	20000	30000
	fe80::250:79ff:fe66:6800/64				

Figure 36 - VPC1 has the 192.168.10.5 IP with 00:50:79:66:68:00 MAC.

```
ubuntu@ubuntu:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:2a:06:03
          inet addr:192.168.10.6  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe2a:603/64  Scope:Link
```

Figure 37 - Ubuntu-PC has the 192.168.10.6 IP with 08:00:27:2a:06:03 MAC.

```
root@bt:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:66:6d:c1
          inet addr:192.168.10.2  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe66:6dc1/64  Scope:Link
```

Figure 38 - Backtrack-PC has the 192.168.10.2 IP with 08:00:27:66:6d:c1 MAC.

When an ARP spoofing attack takes place, the malicious user intervenes replacing the real MAC addresses with fake. The malicious user sends constantly fake ARP messages to the router advertising his MAC address (08:00:27:66:6d:c1) as the MAC address of the Ubuntu-PC (08:00:27:2a:06:03), as it shows the figure 39 from Wireshark.

```
80 112.781000 CadmusCo_66:6d:c1 Broadcast ARP 42 who has 192.168.10.6? Tell 192.168.10.2
...
Frame 80: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)
Ethernet II, Src: CadmusCo_66:6d:c1 (08:00:27:66:6d:c1), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Destination: Broadcast (ff:ff:ff:ff:ff:ff)
Source: CadmusCo_66:6d:c1 (08:00:27:66:6d:c1)
Type: ARP (0x0806)
Address Resolution Protocol (request)
Hardware type: Ethernet (1)
Protocol type: IP (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: request (1)
[Is gratuitous: False]
Sender MAC address: CadmusCo_66:6d:c1 (08:00:27:66:6d:c1)
Sender IP address: 192.168.10.2 (192.168.10.2)
Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
Target IP address: 192.168.10.6 (192.168.10.6)
```

Figure 39 – Backtrack-PC (MAC: 08:00:27:66:6d:c1) asks for the MAC address of the device which has the 192.168.10.6 IP address (Ubuntu-PC).

```

103 133.789000 CadmusCo_66:6d:c1 00:00:00_00:00:00 ARP 42 192.168.10.1 is at 08:00:27:66:6d:c1
---
Frame 103: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)
Ethernet II, Src: CadmusCo_66:6d:c1 (08:00:27:66:6d:c1), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
Destination: 00:00:00_00:00:00 (00:00:00:00:00:00)
Source: CadmusCo_66:6d:c1 (08:00:27:66:6d:c1)
Type: ARP (0x0806)
Address Resolution Protocol (reply)
Hardware type: Ethernet (1)
Protocol type: IP (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: reply (2)
[Is gratuitous: False]
Sender MAC address: CadmusCo_66:6d:c1 (08:00:27:66:6d:c1)
Sender IP address: 192.168.10.1 (192.168.10.1)
Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
Target IP address: 192.168.10.6 (192.168.10.6)

```

Figure 40 – The fake message is advertising that the MAC address of the Ubuntu-PC is the 08:00:27:66:6d:c1 which in reality is the MAC address of Backtrack-PC.

Backtrack-PC also sends fake messages to the Ubuntu-PC with exact the same method, advertising his MAC address as the MAC address of the router, as it shows to the following pictures.

```

648 929.583000 CadmusCo_66:6d:c1 Broadcast ARP 42 Who has 192.168.10.1? Tell 192.168.10.2
---
Frame 648: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)
Ethernet II, Src: CadmusCo_66:6d:c1 (08:00:27:66:6d:c1), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Destination: Broadcast (ff:ff:ff:ff:ff:ff)
Source: CadmusCo_66:6d:c1 (08:00:27:66:6d:c1)
Type: ARP (0x0806)
Address Resolution Protocol (request)
Hardware type: Ethernet (1)
Protocol type: IP (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: request (1)
[Is gratuitous: False]
Sender MAC address: CadmusCo_66:6d:c1 (08:00:27:66:6d:c1)
Sender IP address: 192.168.10.2 (192.168.10.2)
Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
Target IP address: 192.168.10.1 (192.168.10.1)

```

Figure 41 – Backtrack-PC (MAC: 08:00:27:66:6d:c1) asks for the MAC address of the device which has the 192.168.10.1 IP address (router's default gateway).

```

681 974.586000 CadmusCo_66:6d:c1 c2:00:1a:74:00:01 ARP 42 192.168.10.6 is at 08:00:27:66:6d:c1
---
Frame 681: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)
Ethernet II, Src: CadmusCo_66:6d:c1 (08:00:27:66:6d:c1), Dst: c2:00:1a:74:00:01 (c2:00:1a:74:00:01)
Destination: c2:00:1a:74:00:01 (c2:00:1a:74:00:01)
Source: CadmusCo_66:6d:c1 (08:00:27:66:6d:c1)
Type: ARP (0x0806)
[Duplicate IP address detected for 192.168.10.6 (08:00:27:66:6d:c1) - also in use by 08:00:27:66:84:21 (frame 2)]
Address Resolution Protocol (reply)
Hardware type: Ethernet (1)
Protocol type: IP (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: reply (2)
[Is gratuitous: False]
Sender MAC address: CadmusCo_66:6d:c1 (08:00:27:66:6d:c1)
Sender IP address: 192.168.10.6 (192.168.10.6)
Target MAC address: c2:00:1a:74:00:01 (c2:00:1a:74:00:01)
Target IP address: 192.168.10.1 (192.168.10.1)

```

Figure 42 – The fake message is advertising that the MAC address of router's default gateway is the 08:00:27:66:6d:c1 MAC address of Backtrack-PC.

Thus, any traffic destined from router to Ubuntu-PC and the opposite, it will pass over the Backtrack-PC. Illustratively, in case of implementing a Denial-of-Service attack, Backtrack-PC will not forward anything to or from the Ubuntu-PC.

The tools that we will use to implement the ARP spoofing attack are the Nmap, Arpspoof, Driftnet, Urlsnarf and Wireshark from the Backtrack OS. Nmap tool will help in order to discover the other devices on the subnet. Arpspoof tool will use twice against two targets which are Ubuntu-PC and router's default-gateway. Driftnet will help in monitoring the graphics of websites, which are being visited by the Ubuntu-PC. Urlsnarf tool gives detailed information about the typed URLs, where they are going to, the version of the software that uses Ubuntu-PC and many more. Wireshark tool analyzes all the forwarded traffic between the devices, giving the ability to eavesdrop everything.

From the given information of nmap tool, we can concentrate all the necessary details about addresses and ports of the devices in this subnet, as these are shown to the following snapshot.

```
root@bt:~# nmap -F 192.168.10.0/24

Starting Nmap 6.01 ( http://nmap.org ) at 2013-09-23 12:00 EDT
Nmap scan report for 192.168.10.1
Host is up (0.12s latency).
All 100 scanned ports on 192.168.10.1 are filtered
MAC Address: C2:00:0A:74:00:01 (Unknown)

Nmap scan report for 192.168.10.2
Host is up (0.000032s latency).
All 100 scanned ports on 192.168.10.2 are closed

Nmap scan report for 192.168.10.5
Host is up (0.083s latency).
All 100 scanned ports on 192.168.10.5 are filtered
MAC Address: 00:50:79:66:68:00 (Private)

Nmap scan report for 192.168.10.6
Host is up (1.1s latency).
All 100 scanned ports on 192.168.10.6 are closed
MAC Address: 08:00:27:2A:06:03 (Cadmus Computer Systems)

Nmap done: 256 IP addresses (4 hosts up) scanned in 230.70 seconds
root@bt:~#
```

Figure 43 – Results from nmap tool scanning the subnet.

In this case of ARP spoofing attack, ip-forwarding option must be enabled on Backtrack system in order to forward IP packets for other devices (Router and Ubuntu-PC.)

```
root@bt:~# echo 1 > /proc/sys/net/ipv4/ip_forward
root@bt:~# more /proc/sys/net/ipv4/ip_forward
1
```

Figure 44 – Enabling ip-forwarding and verifying that.

Using the arpspoof tool we set the 08:00:27:66:6d:c1 MAC address of Backtrack-PC as the MAC address of router's default gateway to anyone who searches for

The same results at the CLIs of two PCs.

```
root@bt:~# arp -a
? (192.168.10.6) at 08:00:27:66:84:21 [ether] on eth0
? (192.168.10.1) at c2:00:1a:74:00:01 [ether] on eth0
```

Figure 48 – The ARP results on Backtrack-PC after the attack.

```
jb@jb-VirtualBox:~$ arp -a
? (192.168.10.1) at c2:00:1a:74:00:01 [ether] on eth0
? (192.168.10.2) at 08:00:27:66:6d:c1 [ether] on eth0
```

Figure 49 – The ARP results on Ubuntu-PC after the attack.

In order to verify that the launched attack is working, we use the driftnet and urlsnarf tools.

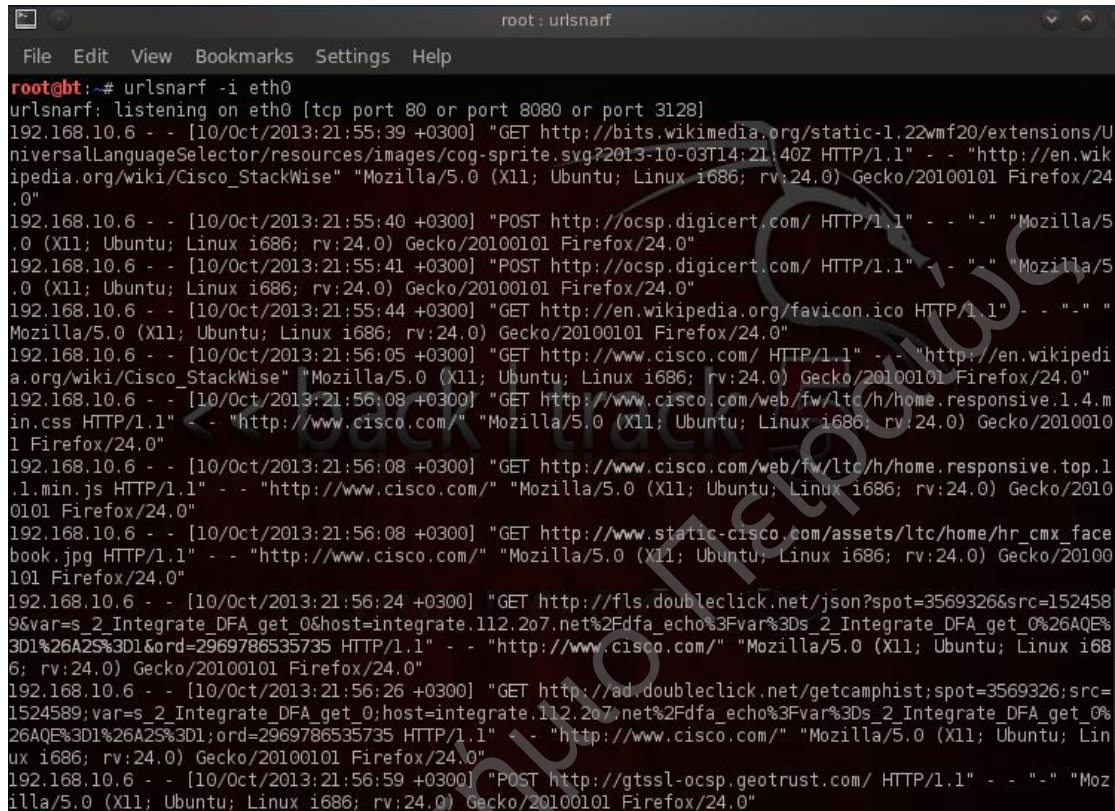
Driftnet starts with the `driftnet -i eth0` command and opens a new window in which appears the graphical content from the websites that visits the victim on Ubuntu-PC.

As Ubuntu-PC visit websites and do search about cisco catalyst switches, driftnet monitors all the non-encrypted graphical material of visited websites.



Figure 50 – Search results on driftnet window.

The other tool `urlsnarf` starts with `urlsnarf -i eth0` command and monitors all the urls and their details from the websites that visits the victim on Ubuntu-PC, as we can distinguish to the following screenshot.



```
root@ubuntu:~# urlsnarf -i eth0
urlsnarf: listening on eth0 [tcp port 80 or port 8080 or port 3128]
192.168.10.6 - - [10/Oct/2013:21:55:39 +0300] "GET http://bits.wikimedia.org/static-1.22wmf20/extensions/UniversalLanguageSelector/resources/images/cog-sprite.svg?2013-10-03T14:21:40Z HTTP/1.1" - - "http://en.wikipedia.org/wiki/Cisco_StackWise" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:24.0) Gecko/20100101 Firefox/24.0"
192.168.10.6 - - [10/Oct/2013:21:55:40 +0300] "POST http://ocsp.digicert.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:24.0) Gecko/20100101 Firefox/24.0"
192.168.10.6 - - [10/Oct/2013:21:55:41 +0300] "POST http://ocsp.digicert.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:24.0) Gecko/20100101 Firefox/24.0"
192.168.10.6 - - [10/Oct/2013:21:55:44 +0300] "GET http://en.wikipedia.org/favicon.ico HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:24.0) Gecko/20100101 Firefox/24.0"
192.168.10.6 - - [10/Oct/2013:21:56:05 +0300] "GET http://www.cisco.com/ HTTP/1.1" - - "http://en.wikipedia.org/wiki/Cisco_StackWise" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:24.0) Gecko/20100101 Firefox/24.0"
192.168.10.6 - - [10/Oct/2013:21:56:08 +0300] "GET http://www.cisco.com/web/fw/ltc/h/home.responsive.1.4.min.css HTTP/1.1" - - "http://www.cisco.com/" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:24.0) Gecko/20100101 Firefox/24.0"
192.168.10.6 - - [10/Oct/2013:21:56:08 +0300] "GET http://www.cisco.com/web/fw/ltc/h/home.responsive.top.1.1.min.js HTTP/1.1" - - "http://www.cisco.com/" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:24.0) Gecko/20100101 Firefox/24.0"
192.168.10.6 - - [10/Oct/2013:21:56:08 +0300] "GET http://www.static-cisco.com/assets/ltc/home/hr_cm_xface_book.jpg HTTP/1.1" - - "http://www.cisco.com/" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:24.0) Gecko/20100101 Firefox/24.0"
192.168.10.6 - - [10/Oct/2013:21:56:24 +0300] "GET http://fls.doubleclick.net/json?spot=3569326&src=1524589&var=s_2_Integrate_DFA_get_0&host=integrate.112.207.net%2Fdfa_echo%3Fvar%3Ds_2_Integrate_DFA_get_0%26AQE%3D1%26A2S%3D1&ord=2969786535735 HTTP/1.1" - - "http://www.cisco.com/" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:24.0) Gecko/20100101 Firefox/24.0"
192.168.10.6 - - [10/Oct/2013:21:56:26 +0300] "GET http://ad.doubleclick.net/getcamhist;spot=3569326;src=1524589;var=s_2_Integrate_DFA_get_0;host=integrate.112.207.net%2Fdfa_echo%3Fvar%3Ds_2_Integrate_DFA_get_0%26AQE%3D1%26A2S%3D1;ord=2969786535735 HTTP/1.1" - - "http://www.cisco.com/" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:24.0) Gecko/20100101 Firefox/24.0"
192.168.10.6 - - [10/Oct/2013:21:56:59 +0300] "POST http://gtssl-ocsp.geotrust.com/ HTTP/1.1" - - "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:24.0) Gecko/20100101 Firefox/24.0"
```

Figure 51 – Urlsnarf tool monitoring all the urls and their details from the visited websites.

Another effective way to have a closer look at victim's steps on local network and Internet is Wireshark. As we see below there is a detailed analysis of victim's movements.

No.	Time	Source	Destination	Protocol	Length	Info
93	19.46816200	192.168.10.2	192.168.10.6	ICMP	82	Redirect
94	19.46819200	192.168.10.6	131.253.61.86	TCP	54	[TCP Dup ACK 92#1]
95	19.47261900	192.168.10.6	131.253.61.86	SSL	257	Client Hello
96	19.47287000	192.168.10.6	131.253.61.86	SSL	257	[TCP Retransmission]
97	19.49705900	131.253.61.86	192.168.10.6	TCP	60	https > 41420 [ACK]
98	19.49707000	131.253.61.86	192.168.10.6	TCP	54	[TCP Dup ACK 97#1]
99	19.88887600	192.168.10.6	23.67.128.170	TCP	74	39844 > http [SYN]
100	19.88901300	192.168.10.2	192.168.10.6	ICMP	102	Redirect
101	19.88905000	192.168.10.6	23.67.128.170	TCP	74	39844 > http [SYN]
102	19.94357000	192.168.10.6	212.205.126.10	HTTP	882	GET /fd/ls/GlinkPir
103	19.94365500	192.168.10.2	192.168.10.6	ICMP	590	Redirect
104	19.94369200	192.168.10.6	212.205.126.10	HTTP	882	[TCP Retransmission]
105	19.94683700	192.168.10.2	192.168.192.2	DNS	85	Standard query 0xfc
106	19.95237900	23.67.128.170	192.168.10.6	TCP	60	http > 39844 [SYN,
107	19.95252100	23.67.128.170	192.168.10.6	TCP	58	http > 39844 [SYN,
108	19.96294700	212.205.126.10	192.168.10.6	TCP	60	http > 40452 [ACK]
109	19.96300600	212.205.126.10	192.168.10.6	TCP	54	[TCP Dup ACK 108#1]
110	19.96821000	192.168.10.6	23.67.128.170	TCP	60	39844 > http [ACK]
111	19.96826900	192.168.10.2	192.168.10.6	ICMP	82	Redirect
112	19.96830000	192.168.10.6	23.67.128.170	TCP	54	[TCP Dup ACK 110#1]

Figure 52 – Wireshark capturing the traffic of the attack.

Countermeasures

As we mentioned at the previous scenario's countermeasures, Dynamic ARP Inspection (DAI) is the most efficient security feature to confront this type of attack, because it intercepts and verifies IP-to-MAC address bindings and discards invalid ARP packets. DAI uses the DHCP snooping database to validate bindings. It associates a trust state with each interface on the switch or the router. Packets arriving on trusted interfaces bypass all DAI validation checks, and those arriving on untrusted interfaces undergo the DAI validation process. In a typical network, all ports on the switch connected to host are configured as untrusted, and switch ports are considered trusted. When DAI is configured, the device rate-limits incoming ARP packets to prevent DoS attacks. The default rate for an untrusted interface is 14 packets per second. Trusted interfaces are not rate-limited. DAI also uses the ARP access control lists (ACLs) and DHCP snooping database for the list of valid IP-to-MAC address bindings. ARP ACLs take precedence over entries in the DHCP snooping bindings database but have to be manually configured. The switch will drop a packet that is denied in the ARP ACLs even if the DHCP snooping database has a valid binding for it. When a switch or a router drops a packet, it is logged in the buffer and generates a system message.

Resources

[1] CCNA Security 640-554 Official Cert Guide
Keith Barker, Scott Morris
Published by Cisco Press, July 2012

[2] CCSP SNRS Exam Self-Study: Mitigating Layer 2 Attacks
By Christian Degu, Greg Bastien, Sara Nasseh.
Published by Cisco Press, July 7, 2006.

[3] CCNP Security Secure 642-637 Official Cert Guide
By Sean Wilkins, Trey Smith
Published by Cisco Press, June 2011

[4] DHCP Consumption Attack and Mitigation Techniques
By Kevin Lauerma and Jeff King.
White paper from www.cisco.com

[5] Network Security Technologies and Solutions (CCIE Professional Development Series)
By Yusuf Bhaiji.
Published by Pearson Education, March 2008

[6] Cisco Router Firewall Security
By Richard Deal.
Published by Cisco Press, August 2004

[7] Computer Forensics
By Gabriele Faggioli, Andrea Ghirardini
Italy: Apogeo, 2009

[8] Malware Forensics Field Guide for Windows Systems: Digital Forensics Field Guides.
By Cameron H. Malin, Eoghan Casey BS MA, 2012.

[9] pcap2wav Xplico interface <http://www.xplico.org/archives/1287>