UNIVERSITY OF TRENTO

Network Security Lab Activity: Man in the Middle (MitM) attacks

Gabriele Gemmi
Lorenzo Brugnera

University of Trento

Wednesday, 18 April 2018

UNIVERSITY OF TRENTO

Outline

- How to mount a MitM attack
  - ARP Spoofing
  - DHCP (DHCPv6) poisoning
  - Evil Twin
- Attacks that can be mounted after the MitM
  - HTTP Interception
  - SSL Stripping
  - HSTS Bypass
  - DNS Spoofing

UNIVERSITY OF TRENTO

What is a MitM attack?

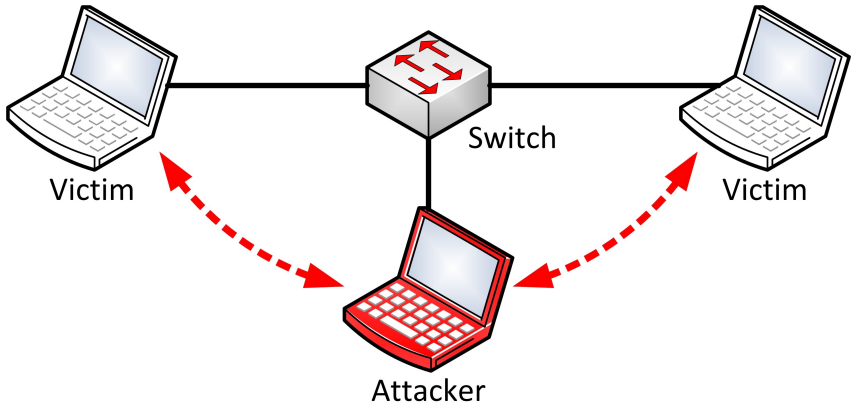


Diagram of a MitM attack

Requisites

- The attacker must be near the victim (in the same local network)

UNIVERSITY OF TRENTO

What is a MitM attack?

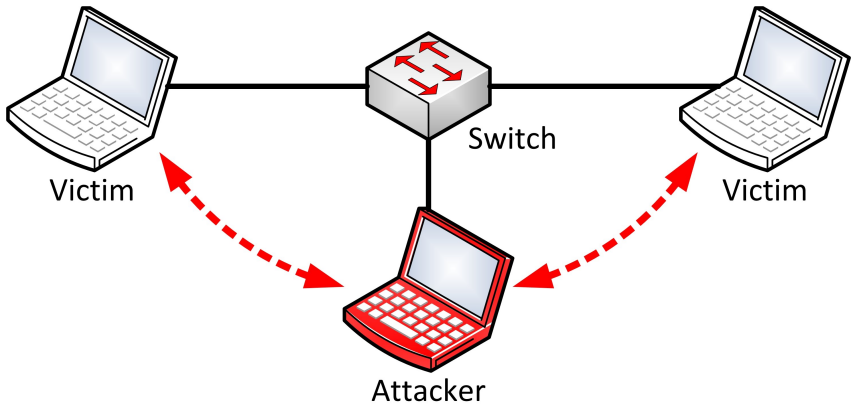


Diagram of a MitM attack

How to mount this attack

- The attacker must be physically connected between the victim and the rest of the network

or

- The attacker must hijack the traffic from the victim to himself

UNIVERSITY OF TRENTO

Network layer attacks

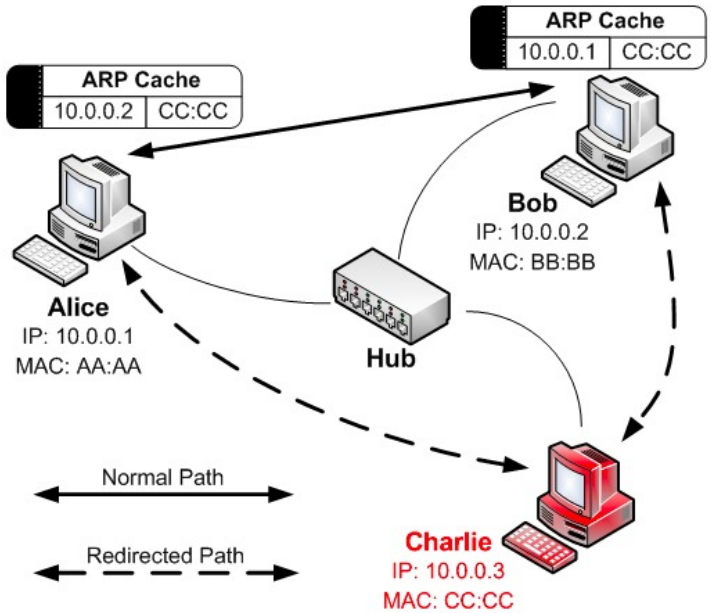
- ARP poisoning
- DHCP (DHCPv6) poisoning
- Evil Twin

UNIVERSITY OF TRENTO

ARP Poisoning

How it works

- The attacker floods the network with poisoned ARP messages
- The mapping between IPs and MACs is altered in order to hijack the communication through the attacker



ARP Spoofing attack diagram

UNIVERSITY OF TRENTO

ARP Poisoning

How to prevent it?

ARP Poisoning

How to prevent it?

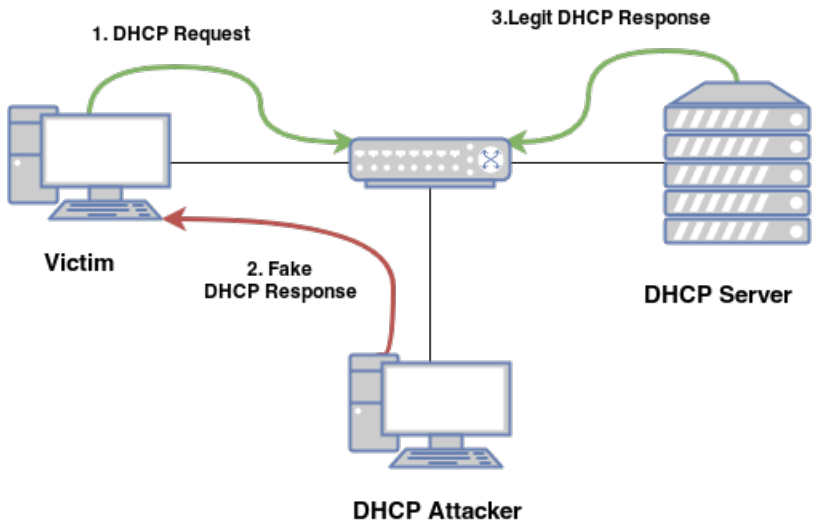
- ARP poisoning proof switches
- VPN

UNIVERSITY OF TRENTO

DHCP (DHCPv6) poisoning

How it works

- The attacker sets-up a rogue DHCP server
- Each time a victim sends a DHCP request the rogue server answers with a forged response
- The response contains a malicious default gateway to perform the MitM attack



DHCP poisoning attack diagram

DHCP (DHCPv6) poisoning

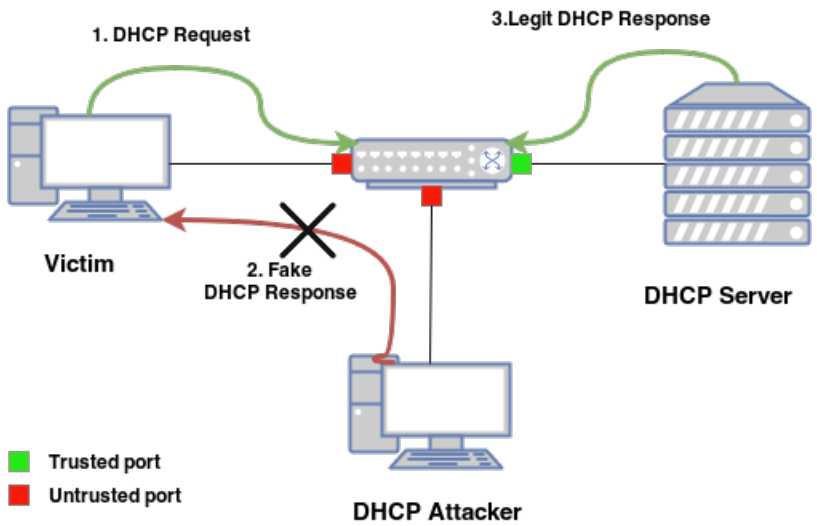
How to prevent it?

UNIVERSITY OF TRENTO

DHCP (DHCPv6) poisoning

How to prevent it?

- A smart switch can be configured to allow DHCP response only on certain trusted ports



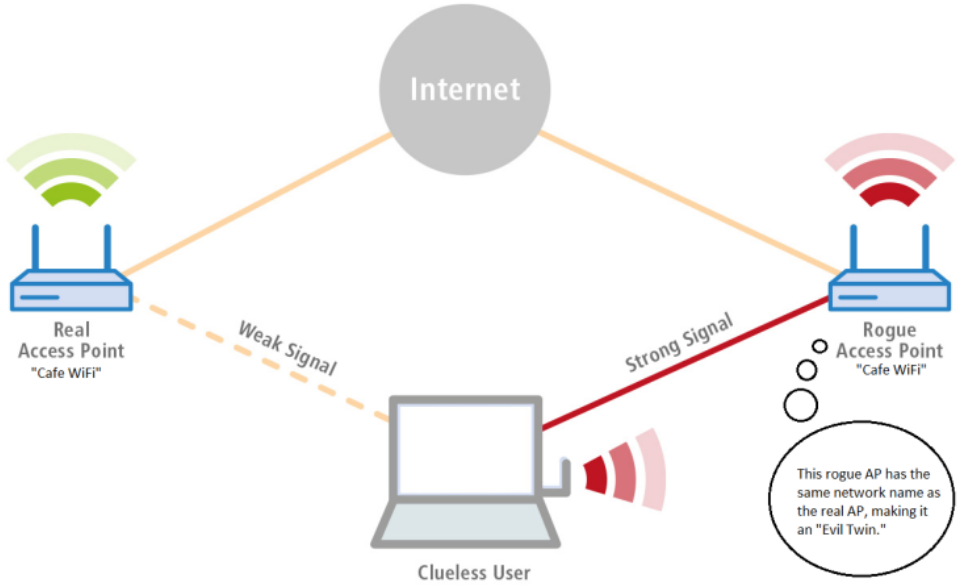
DHCP snooping diagram

UNIVERSITY OF TRENTO

Evil Twin

How it works

- The attacker sets-up a rogue Wi-Fi Access Point with the same ESSID as the target network
- The victim must receive the rogue AP with a stronger signal than the legit one



Evil Twin attack diagram

Evil Twin

How to prevent it?

UNIVERSITY OF TRENTO

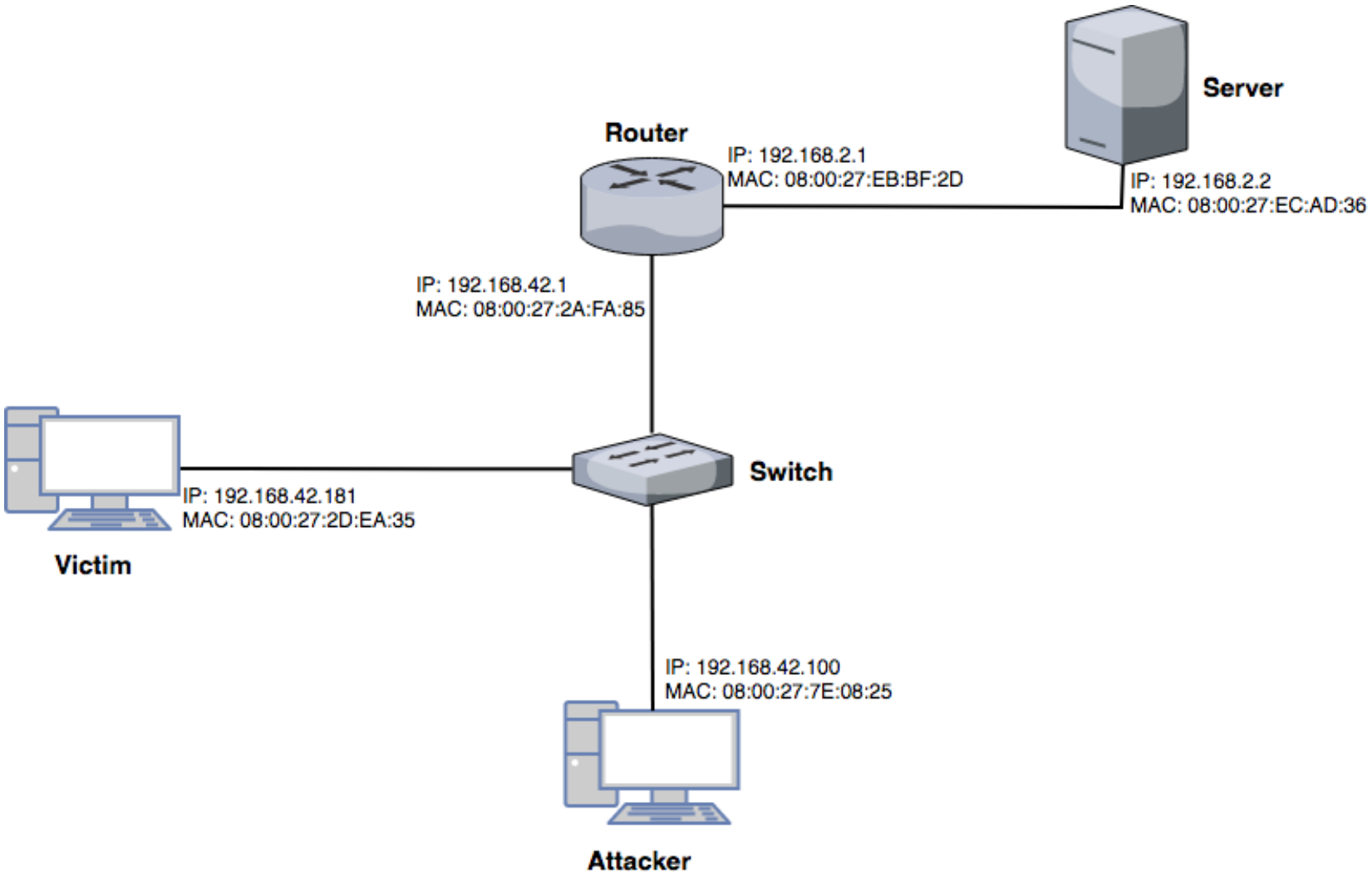
Evil Twin

How to prevent it?

- A simple authentication (WPA) doesn't ensure the client that the AP is legit (The attacker just need to discover the key)
- The client must authenticate the AP (802.1x) and verify its legitimacy

UNIVERSITY OF TRENTO

Network Topology



Topology of the VMs network

Tools

This is a list of tools we will be using in this lab, in the next slides the usage and the purpose will be explained

- arpspoof
- wireshark
- dnsspoof
- sslstrip
- sslstrip2
- dns2proxy

UNIVERSITY OF TRENTO

Tips and Tricks

Useful infos

- Type `sudo` before every command, the password is "netsec"

To do after every exercise

- Flush the DNS cache: `systemd-resolve --flush-cache`
- Clean the iptables chains `iptables -t <chain name> -F`

UNIVERSITY OF TRENTO

MitM Network attack

- To mount the following attacks you can use any of the attacks we illustrated you
- Since you already know how to mount it and due to its simplicity, we wll be using ARP spoofing

- You can use either ettercap or this simple command line tool `arpspoof -t <victim ip> -r <router ip>`

UNIVERSITY OF TRENTO

How to start the laboratory

- Open virtualbox
- Run "netsec_router" and "netsec_server" in headless mode
- Run "netsec_victim" and "netsec_malicious" normally

UNIVERSITY OF TRENTO

HTTP Interception

How it works

- Using wireshark it's possible to capture all the traffic that flows between the victim and the router
- Sensitive information can be sniffed by the attacker

UNIVERSITY OF TRENTO

HTTP Interception

Exercise 1

- Mount an MitM network attack
- Open a browser and navigate to "http://www.homepage.it"
- Sniff the HTTP traffic exchanged between the victim and the server

HTTP Interception

How to prevent?

UNIVERSITY OF TRENTO

HTTP Interception

How to prevent?

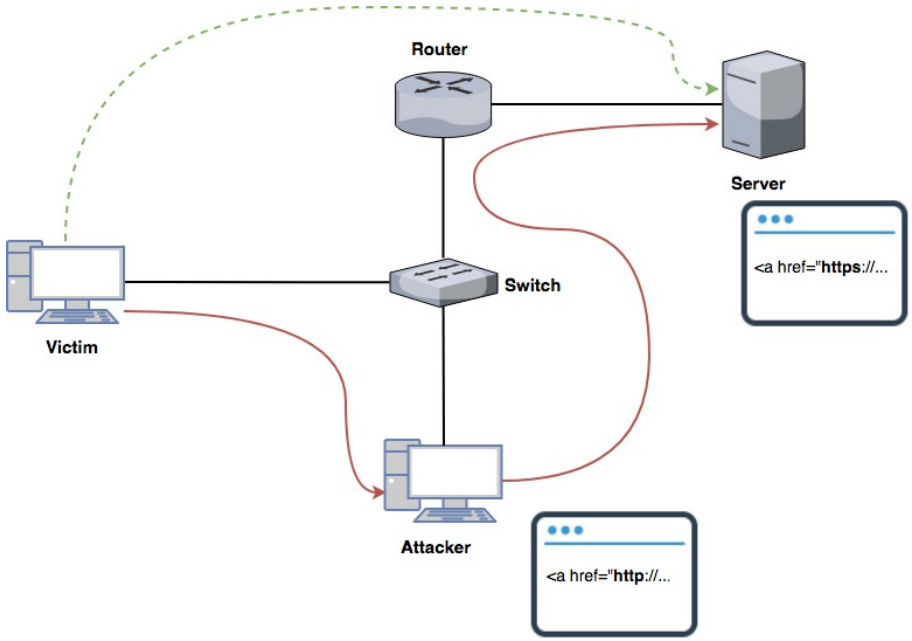
- An encrypted channel can preserve the confidentiality
  - SSL/TLS
  - VPN

UNIVERSITY OF TRENTO

SSL Stripping

How it works

- An attacker *in the middle* manipulates the HTTP responses
- Every `https://` url in the response gets downgraded to `http://`



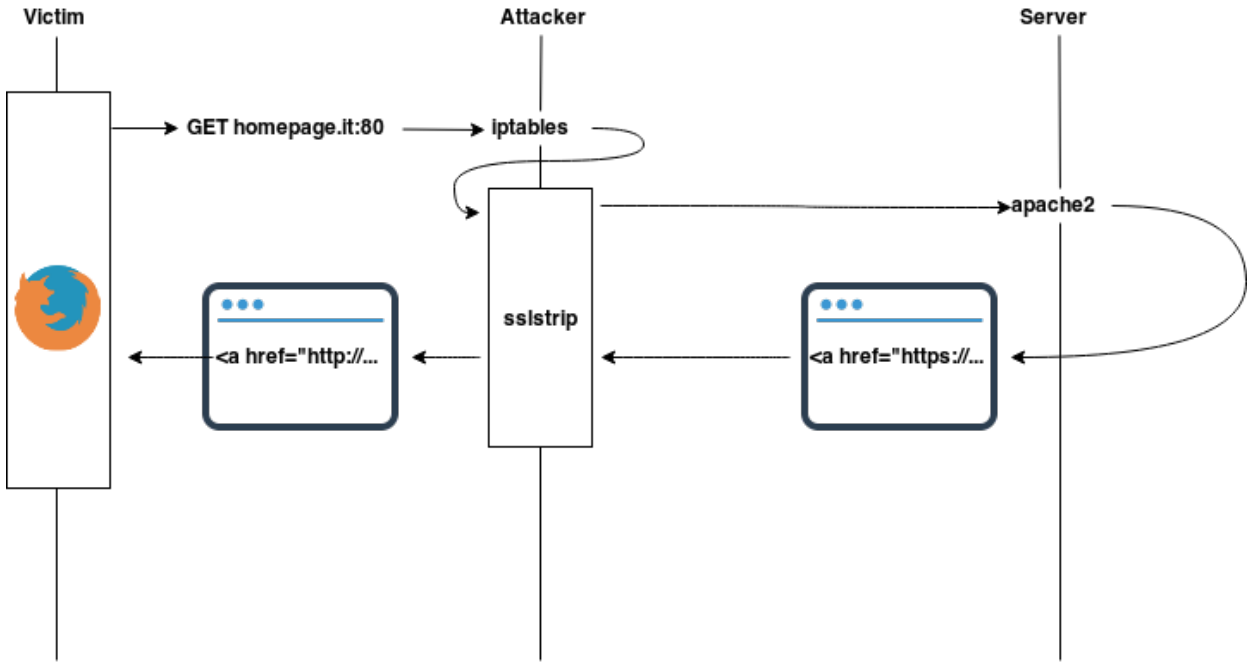
SSL Stripping attack diagram

UNIVERSITY OF TRENTO

SSL Stripping

How it works

- An attacker *in the middle* manipulates the HTTP responses
- Every `https://` url in the response gets downgraded to `http://`



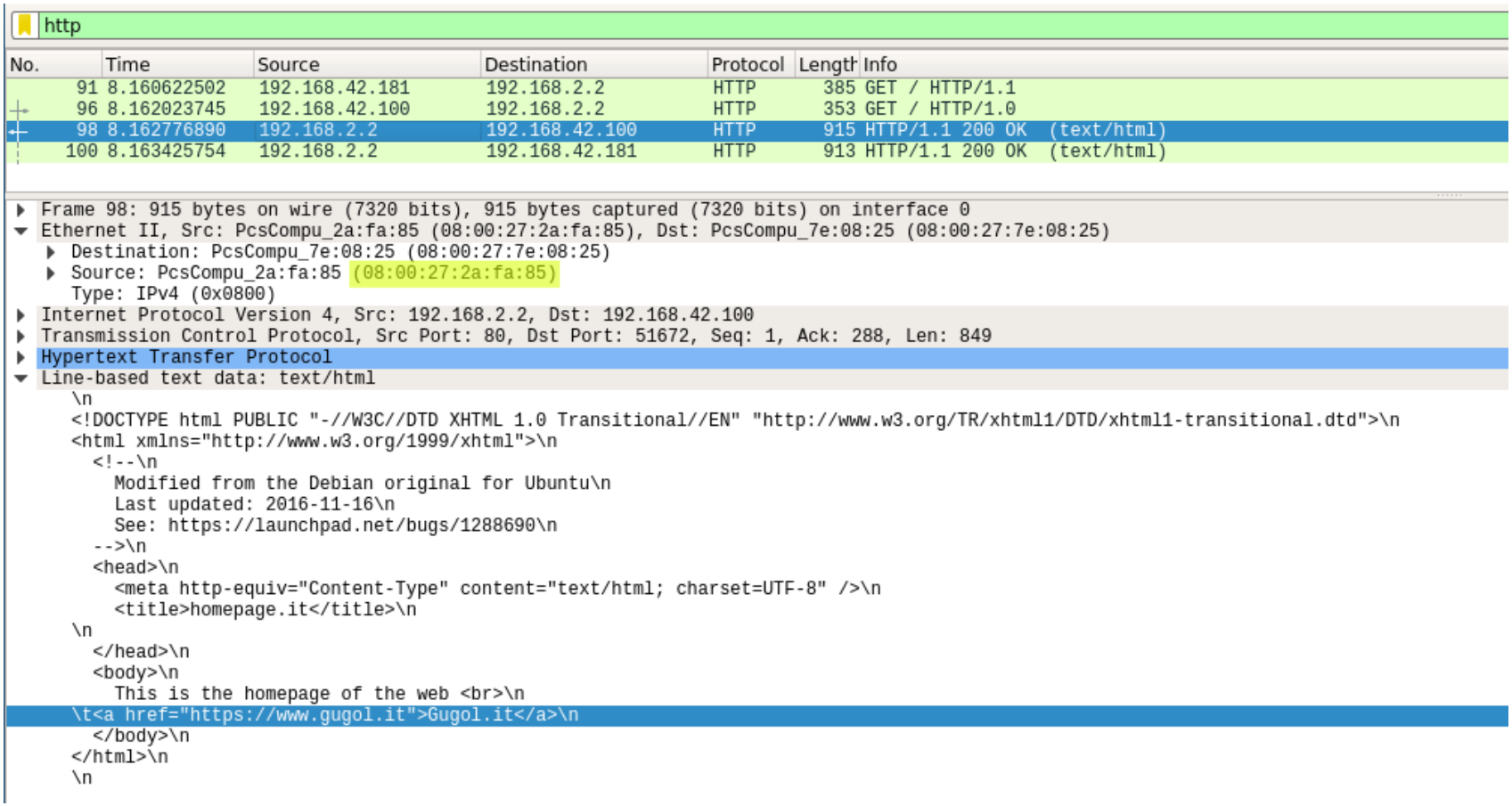
SSL Stripping attack diagram

UNIVERSITY OF TRENTO

SSL Stripping

How it works

- The URL in the page is sent from the server to SSLstrip

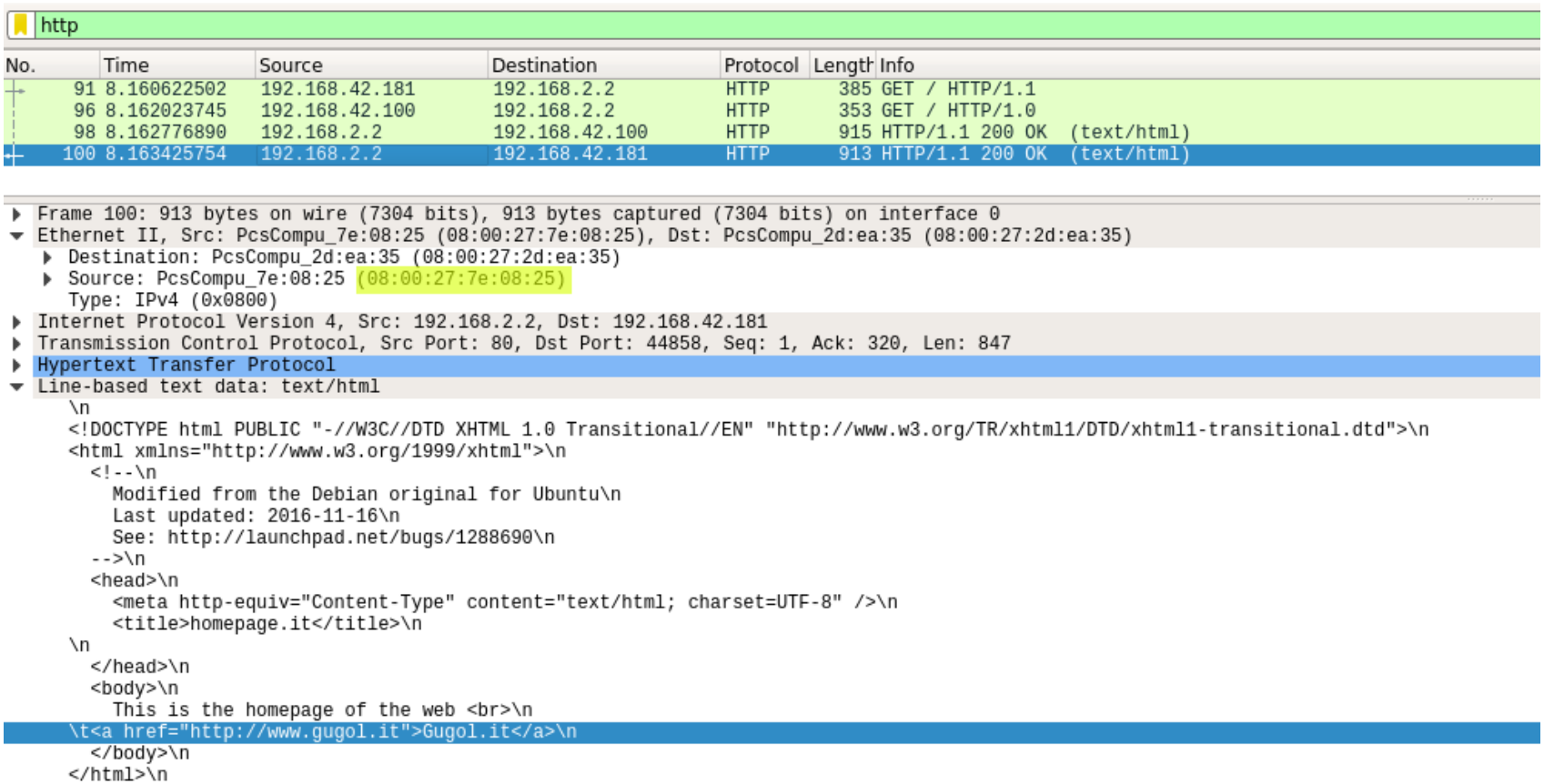


UNIVERSITY OF TRENTO

SSL Stripping

How it works

- SSLstrip replaces every `https://` link with the respective `http:// one`



UNIVERSITY OF TRENTO

SSL Stripping

In practice

- `sslstrip` is an HTTP proxy that manipulates the messages to perform the attack
- The HTTP traffic flowing through the attacker must be redirected to `sslstrip`

Usage

```
sslstrip -l <port>
```

UNIVERSITY OF TRENTO

SSL Stripping

Exercise 2

- Mount a MitM attack
- Setup `sslstrip` to manipulate the HTTP traffic
- Using `iptables` redirect the traffic from the port 80 to the port that `sslstrip` is using
- Intercept the traffic using Wireshark
- Navigate to `www.homepage.it` and click to the URL
- Analyze the behaviour of `sslstrip`

UNIVERSITY OF TRENTO

SSL Stripping

Exercise 2

- Mount a MitM attack
- Setup `sslstrip` to manipulate the HTTP traffic
- Using `iptables` redirect the traffic from the port 80 to the port that `sslstrip` is using
- Intercept the traffic using Wireshark
- Navigate to `www.homepage.it` and click to the URL
- Analyze the behaviour of `sslstrip`

```
iptables -t nat -A PREROUTING -p tcp
--destination-port <web server port> -j REDIRECT
--to-port <sslstrip port>
```

SSL Stripping

How to prevent?

UNIVERSITY OF TRENTO

SSL Stripping

How to prevent?

- HTTP Strict Transport Security (HSTS) is a web security policy to protect against protocol downgraded attacks
- Declaring the HSTS policy, the web server forces a browser to use HTTPS
- The HSTS policy is communicated by the server via an HTTPS response header field named *Strict-Transport-Security*

UNIVERSITY OF TRENTO

SSL Stripping

How to prevent?

- HTTP Strict Transport Security (HSTS) is a web security policy to protect against protocol downgraded attacks
- Declaring the HSTS policy, the web server forces a browser to use HTTPS
- The HSTS policy is communicated by the server via an HTTPS response header field named *Strict-Transport-Security*

UNIVERSITY OF TRENTO

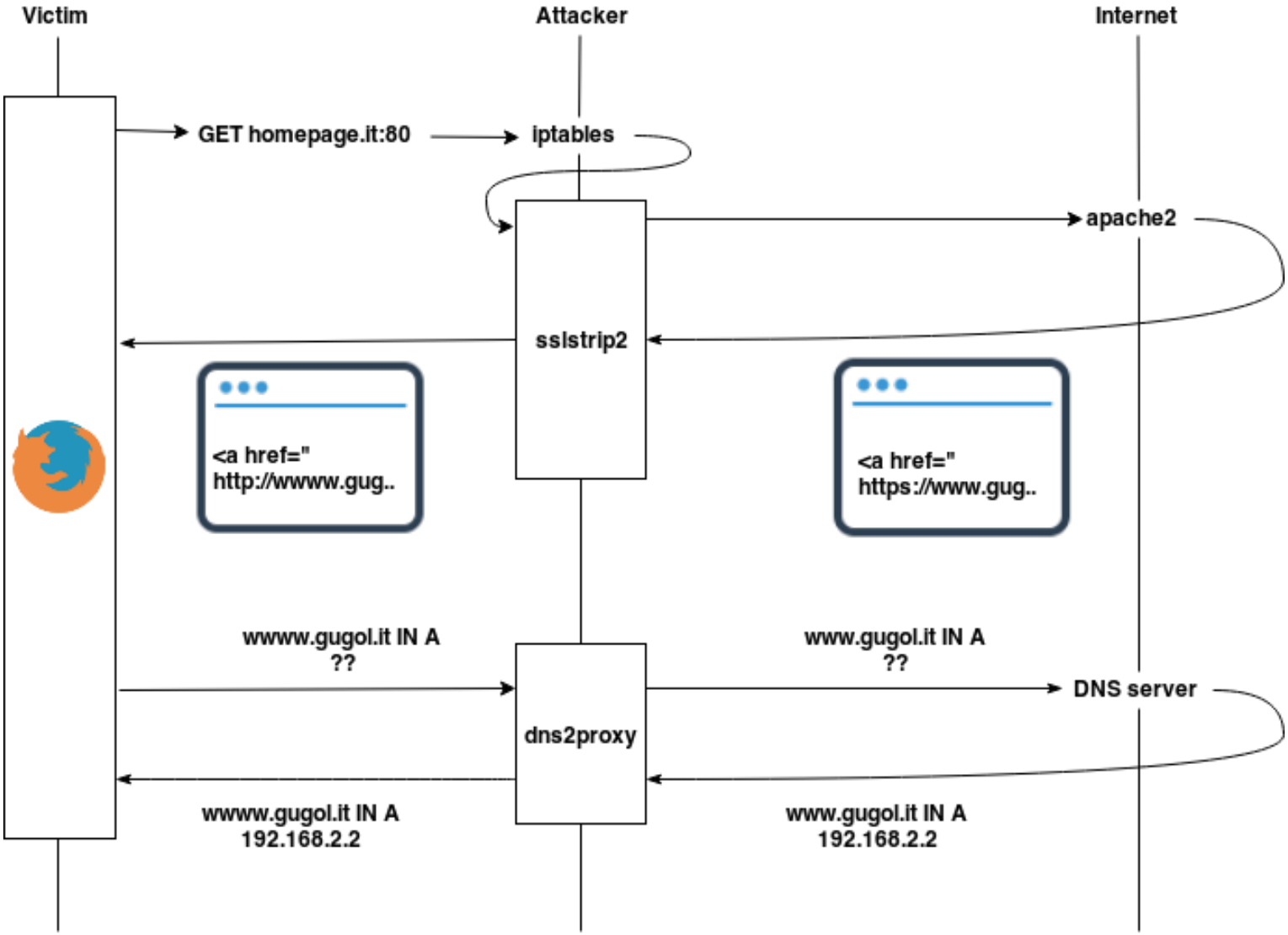
HSTS Bypass

How it works

- The HSTS policy is associated with a specific domain name
- Changing just by one letter the domain name, the browser will not apply the policy anymore
- For example an 'l' (uncapital L) could become an 'I' (capital i)

UNIVERSITY OF TRENTO

HSTS Bypass



UNIVERSITY OF TRENTO

HSTS Bypass

In practice

- `sslstrip` can be modified to change the domain name inside the url all the times it strips an HTTPS url
- The DNS queries must also be manipulated using `dns2proxy`

Usage

- sslstrip2 is in `Desktop/sslstrip2-exercise/`
- It can be executed running `./sslstrip2 -l <port>`
- dns2proxy is in `Desktop/dns2proxy-exercise/`
- It can be executed running `./dns2proxy`

UNIVERSITY OF TRENTO

HSTS Bypass

Exercise 3

- Mount a MitM attack
- Implement the missing code in `sslstrip/URLMonitor.py`
- Redirect all the traffic in the attacker VM from the port 80 to the port where sslstrip2 is running
- Implement the missing code in `dns2proxy.py`
- Verify that dns2proxy is working properly using `nslookup`
- Redirect all the traffic in the attacker VM changing the destination ip
- Analyze the behaviour using Wireshark

UNIVERSITY OF TRENTO

HSTS Bypass

Exercise 3

- Mount a MitM attack
- Implement the missing code in `sslstrip/URLMonitor.py`
- Redirect all the traffic in the attacker VM from the port 80 to the port where sslstrip2 is running
- Implement the missing code in `dns2proxy.py`
- Verify that dns2proxy is working properly using `nslookup`
- Redirect all the traffic in the attacker VM changing the destination ip
- Analyze the behaviour using Wireshark

```
iptables -t nat -A PREROUTING -p udp
--destination-port 53 -i enp0s8 -j DNAT --to
<attacker ip>
```

UNIVERSITY OF TRENTO

HSTS Bypass - sslstrip2 code

```
31      urlExpression       = re.compile(r"(https://[\w\d:#@%/;$()~_?\+-=\\\.&]*)", re.IGNORECASE)
```

```
138     def replaceSecureLinks(self, data):
139         substitution = {}
140         patchDict = self.urlMonitor.patchDict
141         if len(patchDict) > 0:
142             dregex = re.compile("(%s)" % "|".join(map(re.escape, patchDict.keys())))
143             data = dregex.sub(lambda x: str(patchDict[x.string[x.start():x.end()]]), data)
144             # apply the regex to find https URLs in the page
145             iterator = re.finditer(ServerConnection.urlExpression, data)
146         for match in iterator:
147             # for each match of the regexp it memorizes in substitution[]
148             # the original url associated with the spoofed one
149             url = match.group()
150             logging.debug("Found secure reference: " + url)
151             newurl = self.urlMonitor.addSecureLink(self.client.getClientIP(), url)
152             logging.debug("LEO replacing %s => %s" % (url, newurl))
153             substitution[url] = newurl
154         if len(substitution) > 0:
155             # apply a regexp to substitute all the occurencies
156             # of the original URLs with the spoofed ones
157             dregex = re.compile("(%s)" % "|".join(map(re.escape, substitution.keys())))
158             data = dregex.sub(lambda x: str(substitution[x.string[x.start():x.end()]]), data)
159         return data
```

UNIVERSITY OF TRENTO

HSTS Bypass - sslstrip2 code

```
40      def addSecureLink(self, client, url):
41          methodIndex = url.find("//") + 2
42          method      = url[0:methodIndex]
43          pathIndex   = url.find("/", methodIndex)
44          if pathIndex is -1:
45              pathIndex = len(url)
46              url += "/"
47          host        = url[methodIndex:pathIndex].lower()
48          path        = url[pathIndex:]
49          port        = 443
50          portIndex   = host.find(":")
51          if (portIndex != -1):
52              host = host[0:portIndex]
53              port = host[portIndex + 1:]
54              if len(port) == 0:
55                  port = 443
56          fake_domain = ''
57          # EDIT HERE:
58          # if host starts with "www." add a 4th w
59          # otherwise if there's no "www." at the beginning add something
60          # that the victim shouldn't notice (like "web")
61          #
62          #
63          # STOP EDIT HERE
64          logging.debug("LEO: ssl host      (%s) tokenized (%s)" % (host, fake_domain))
65          url = 'http://' + host + path
66          self.real[fake_domain] = host
67          self.strippedURLs.add((client, url))
68          self.strippedURLPorts[(client, url)] = int(port)
69          return 'http://' + fake_domain + path
```

UNIVERSITY OF TRENTO

HSTS Bypass - dns2proxy code

```
122 def requestHandler(address, message):
123     resp = None
124     qtime = time()
125     seconds_betwen_ids = 1
126     try:
127         message_id = ord(message[0]) * 256 + ord(message[1])
128         DEBUGLOG('msg id = ' + str(message_id))
129         if message_id in serving_ids:
130             if (qtime - serving_ids[message_id]) < seconds_betwen_ids:
131                 DEBUGLOG('I am already serving this request.')
132                 return
133         serving_ids[message_id] = qtime
134         DEBUGLOG('Client IP: ' + address[0])
135         src_ip = address[0]
136         try:
137             # parse the dns message
138             msg = dns.message.from_wire(message)
139             try:
140                 op = msg.opcode()
141                 if op == 0:
142                     # standard and inverse query
143                     qs = msg.question
144                     if len(qs) > 0:
145                         q = qs[0]
146                         DEBUGLOG('request is ' + str(q))
147                         save_req(LOGREQFILE, 'Client IP: ' + address[0] + '    request is    ' + str(q) + '\n')
148                         if q.rdtype == dns.rdatatype.A:
149                             DEBUGLOG('Doing the A query....')
150                             resp = std_A_qry(msg, src_ip)
151                         elif q.rdtype == dns.rdatatype.PTR:
152                             #DEBUGLOG('Doing the PTR query....')
153                             resp = std_PTR_qry(msg)
154                         elif q.rdtype == dns.rdatatype.MX:
```

UNIVERSITY OF TRENTO

HSTS Bypass - dns2proxy code

```
279 def std_A_qry(msg, src_ip):
280     global requests
281     global fake_ips
282 
283     qs = msg.question
284     DEBUGLOG(str(len(qs)) + ' questions.')
285     resp = make_response(qry=msg)
286     for q in qs:
287         qname = q.name.to_text()[:-1]
288         DEBUGLOG('q name = ' + qname)
289         host = qname.lower()
290         ips = DNSanswer(qname.lower(), 'A')
291         # If the domain requested doesn't exists, strips the domaina adding a 4th w
292         if isinstance(ips, numbers.Integral):
293             # SSLSTRIP2 transformation
294             real_domain = ''
295             # EDIT HERE:
296             # if the host starts with "wwww." remove one 'w'
297             # otherwise remove the string that you added ("web")
298             #
299             #
300             #
301             #
302             # STOP EDITING HERE:
303             # If the real domain exists return the answer to the client
304             if real_domain != '':
305                 DEBUGLOG('SSLStrip2 transforming host: %s => %s ...' % (host, real_domain))
306                 ips = DNSanswer(real_domain, 'A')
307         # If the real domain doesn't exist answer with NXDOMAIN
308         if isinstance(ips, numbers.Integral):
309             DEBUGLOG('No host....')
310             resp = make_response(qry=msg, RCODE=3)  # RCODE =  3        NXDOMAIN
311             return resp
```

HSTS Bypass

How to prevent?

UNIVERSITY OF TRENTO

HSTS Bypass

How to prevent?

- The user must always check the correctness of the URL in the address bar

UNIVERSITY OF TRENTO

DNS Spoofing

How it works

- DNS messages are exchanged in clear using the UDP protocol on port 53
- An attacker who is *in the middle* can manipulate the DNS responses

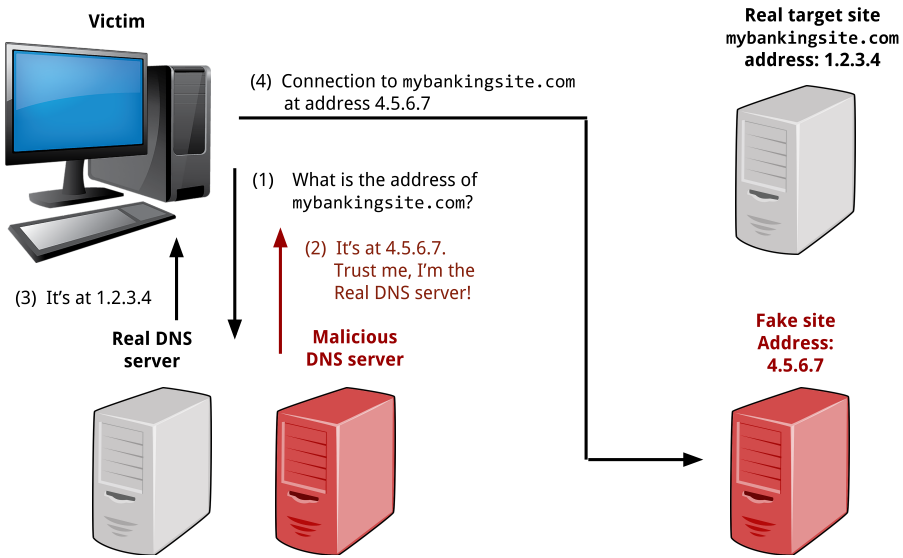


Diagram of the attack

UNIVERSITY OF TRENTO

DNS Spoofing

In practice

- `dnsspoof` forges replies to arbitrary DNS queries on the LAN

Usage

```
dnsspoof [-i interface] [-f hostsfile]
```

The hostfile contains the record associated with the A response for example:

```
www.google.it      192.168.1.1
www.facebook.com   192.168.1.1
```

UNIVERSITY OF TRENTO

DNS Spoofing

Exercise

- There's a malicious webserver running on the attcker VM
- Create a proper hostsfile to spoof requests for `www.gugol.it` pointing to the malicious webserver
- Mount a MitM attack
- Setup `dnsspoof` to answer to the DNS query of the victim
- Navigate to `www.gugol.it` to verify that the attacks has succeeded

UNIVERSITY OF TRENTO

DNS Spoofing

Exercise

- There's a malicious webserver running on the attcker VM
- Create a proper hostsfile to spoof requests for `www.gugol.it` pointing to the malicious webserver
- Mount a MitM attack
- Setup `dnsspoof` to answer to the DNS query of the victim
- Navigate to `www.gugol.it` to verify that the attacks has succeeded
- Block the DNS response from the legit server using `iptables`

UNIVERSITY OF TRENTO

DNS Spoofing

Exercise

- There's a malicious webserver running on the attcker VM
- Create a proper hostsfile to spoof requests for `www.gugol.it` pointing to the malicious webserver
- Mount a MitM attack
- Setup `dnsspoof` to answer to the DNS query of the victim
- Navigate to `www.gugol.it` to verify that the attacks has succeeded
- Block the DNS response from the legit server using `iptables`

```
iptables -A FORWARD -s <victim ip> -p udp --dport
<dns port> -j DROP
```

UNIVERSITY OF TRENTO

DNS Spoofing

How to prevent?

UNIVERSITY OF TRENTO

DNS Spoofing

How to prevent?

- Cached responses cannot be spoofed
- DNSSec guarantees integrity of the records by using digital signature

Thank you