CSE508 Network Security

Stony Brook University

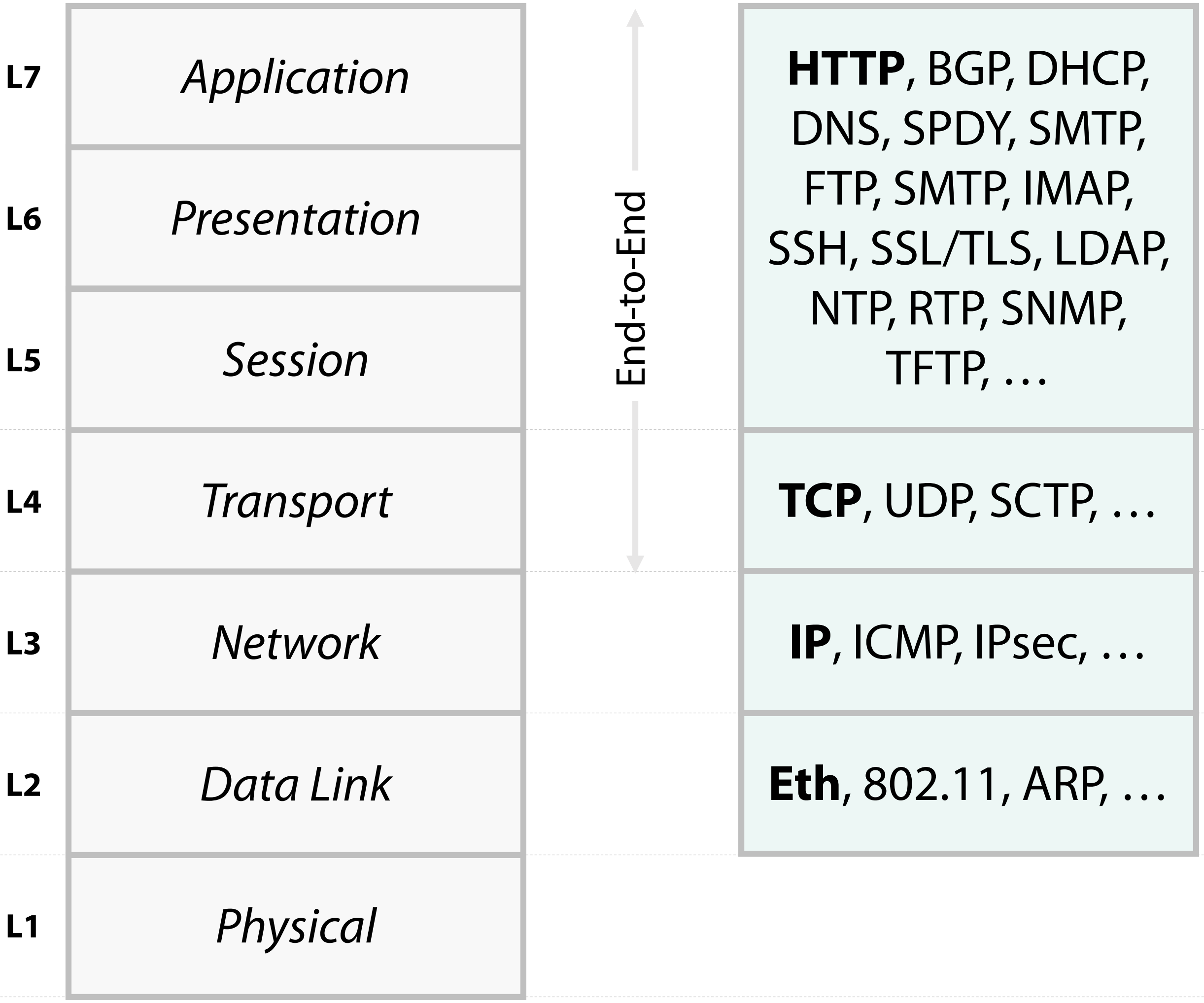
2021-02-16

Lower Layers (Part 2)

Michalis Polychronakis

Stony Brook University

Basic Internet Protocols (OSI Model vs. Reality)



				Deliver to:	**Based on:**
L7	*Application*	End-to-End	**HTTP**, BGP, DHCP, DNS, SPDY, SMTP, FTP, SMTP, IMAP, SSH, SSL/TLS, LDAP, NTP, RTP, SNMP, TFTP, …		
L6	*Presentation*				
L5	*Session*				
L4	*Transport*		**TCP**, UDP, SCTP, …	Dst. application	Port number
L3	*Network*		**IP**, ICMP, IPsec, …	Dst. host	IP address
L2	*Data Link*		**Eth**, 802.11, ARP, …	Next hop	MAC address
L1	*Physical*			Wire/air/pigeon	Network interface (NIC)

Internet Protocol (IP)

Routing: deliver packets from a source to a destination based on the destination IP address

- Through several hops (routers) – see `traceroute`, `tracepath`
- Connectionless, best effort: no ordering or delivery guarantees
- Source IP address is not authenticated ➔ can be easily spoofed!

IPv6: most recent version, uses 128-bit addresses

- IPv4 space was exhausted in 2011
- IPv6 deployment has been slow but is now ramping up

Packets too large for the next hop are *fragmented* into smaller ones

- Maximum transmission unit (MTU)

IPv4

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IPv6

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Traffic Class |           Flow Label                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |  Next Header  |   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                              Source                           |
|                             Address                           |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                           Destination                         |
|                             Address                           |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Network Layer Attacks

ICMP (Internet Control Message Protocol): Used to exchange error messages about IP datagram delivery

- Smurf Attack (DoS with spoofed broadcast Echo request) (future lecture)
- Reconnaissance (future lecture)
- Exfiltration using ICMP tunneling (future lecture)
- ICMP redirect MitM
- *Organizations typically block incoming/outgoing ICMP traffic due to all the above*

IP spoofing: conceal the real IP address of the sender

- Mostly used in DDoS attacks (future lecture)
- Ingress and egress filtering limit its applicability

IP fragmentation: confuse packet filters and intrusion detection systems

- Split important information across two or more packets (future lecture)

Transmission Control Protocol (TCP)

Provides *reliable* virtual circuits to user processes

- Connection-oriented, reliable transmission
- Packets are shuffled around, retransmitted, and reassembled to match the original data ***stream***

Sender: breaks data stream into packets

- Attaches a sequence number on each packet

Receiver: reassembles the original stream

- Acknowledges receipt of received packets
- Lost packets are sent again

TCP

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   .... data ....                                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

TCP 3-way Handshake

Sequence/acknowledgement numbers

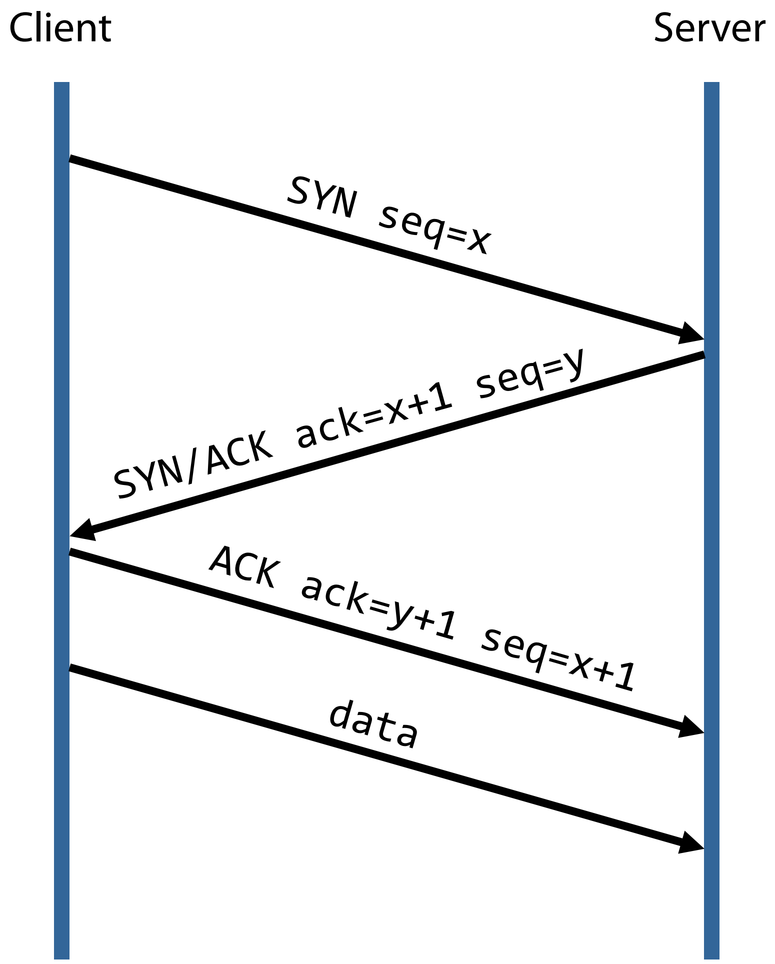
Retransmissions, duplicate filtering, flow control

Seq: the position of the segment's data in the stream

The payload of this segment contains data starting from X

Ack: the position of the next expected byte in the stream

All bytes up to X received correctly, next expected byte is X+1



Transport Layer Attacks

Sequence Number Attacks

- TCP connection hijacking/spoofing
- DoS (connection reset)

Port scanning (future lecture)

OS Fingerprinting (future lecture)

- Intricacies of TCP/IP stack implementations

Denial of Service (DoS) (future lecture)

- Resource exhaustion
- Blind RST injection

Content injection/manipulation (MotS, MitM)

TCP Sequence Number Prediction

Goal: spoof a trusted host (initially described by Robert Morris in 1985)

Construct a valid TCP packet sequence without ever receiving any responses from the server

Exploits *predictability* in initial sequence number (ISN) generation

TCP sessions are established with a three-way handshake:

Client → Server: SYN(ISN_C)
Server → Client: SYN($\mathbf{ISN_S}$), ACK(ISN_C)
Client → Server: ACK(ISN_S)

If the ISNs generated by a host are predictable, an attacker does not need to see the SYN response to successfully establish a TCP session

Impersonating a Trusted Host

Old TCP stacks would increment the sequence number once per second

- Highly predictable with a single observation at a known time [Bellovin ’89]

Host impersonation based on a previous **ISN** observation

Attacker	→	Server:	SYN(ISN_A), SRC IP = Attacker
Server	→	Attacker:	SYN(**ISN_S**), ACK(ISN_A)
Attacker	→	Server:	SYN(ISN_A), SRC IP = Trusted
Server	→	Trusted:	SYN(**ISN_S**), ACK(ISN_A)
Attacker	→	Server:	ACK(**ISN_S**), SRC IP = Trusted, **ATTACK DATA**

Execute commands based on lists of trusted hosts

- `rsh`, `rcp`, other “`r`” commands… (hopefully not used these days)

Solution: randomized ISN generation

Man-on-the-Side Attack

Main capabilities: packet capture + packet injection

- Sniff for requests and forge responses

Requires a privileged position between the victim and the destination

- Attackers **can** *observe* transmitted packets and *inject* new ones
- Attackers **cannot** *modify* or *drop* transmitted packets

But a *less privileged* position than what is required for a MitM attack (!)

- Also much easier: no need to keep per-connection state and relay traffic

Example: unprotected (non-encrypted) WiFi network

- MotS: any client that joins the network can mount it *right away*
- MitM: need to compromise the access point or perform ARP poisoning

Man-on-the-Side Attack

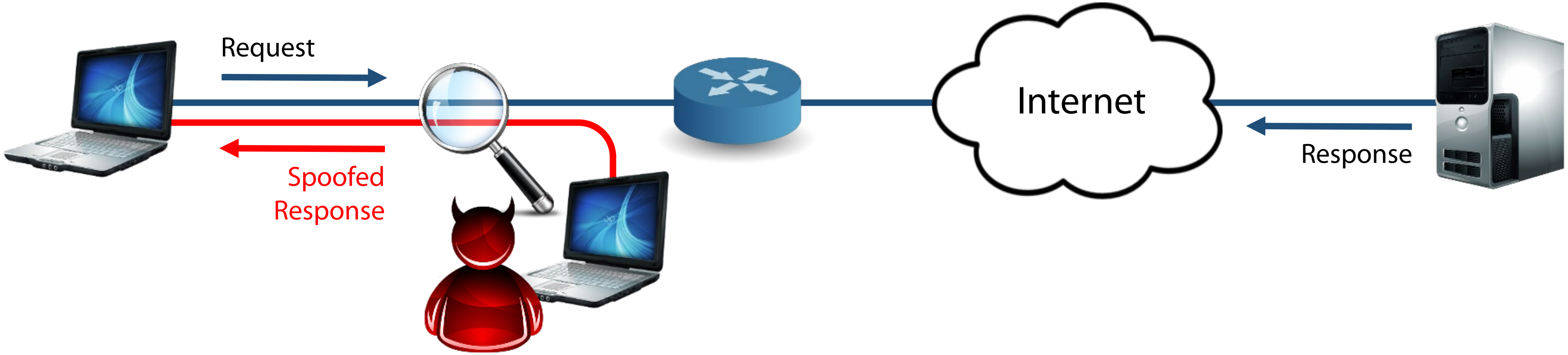
Race condition: the attacker's forged response should arrive to the victim before the actual response from the server

- Most operating systems will accept the first packet they see as valid

No need to guess seq/ack numbers—*just sniff them from the request (!)*

- The rest of the original stream can follow after the injected packet

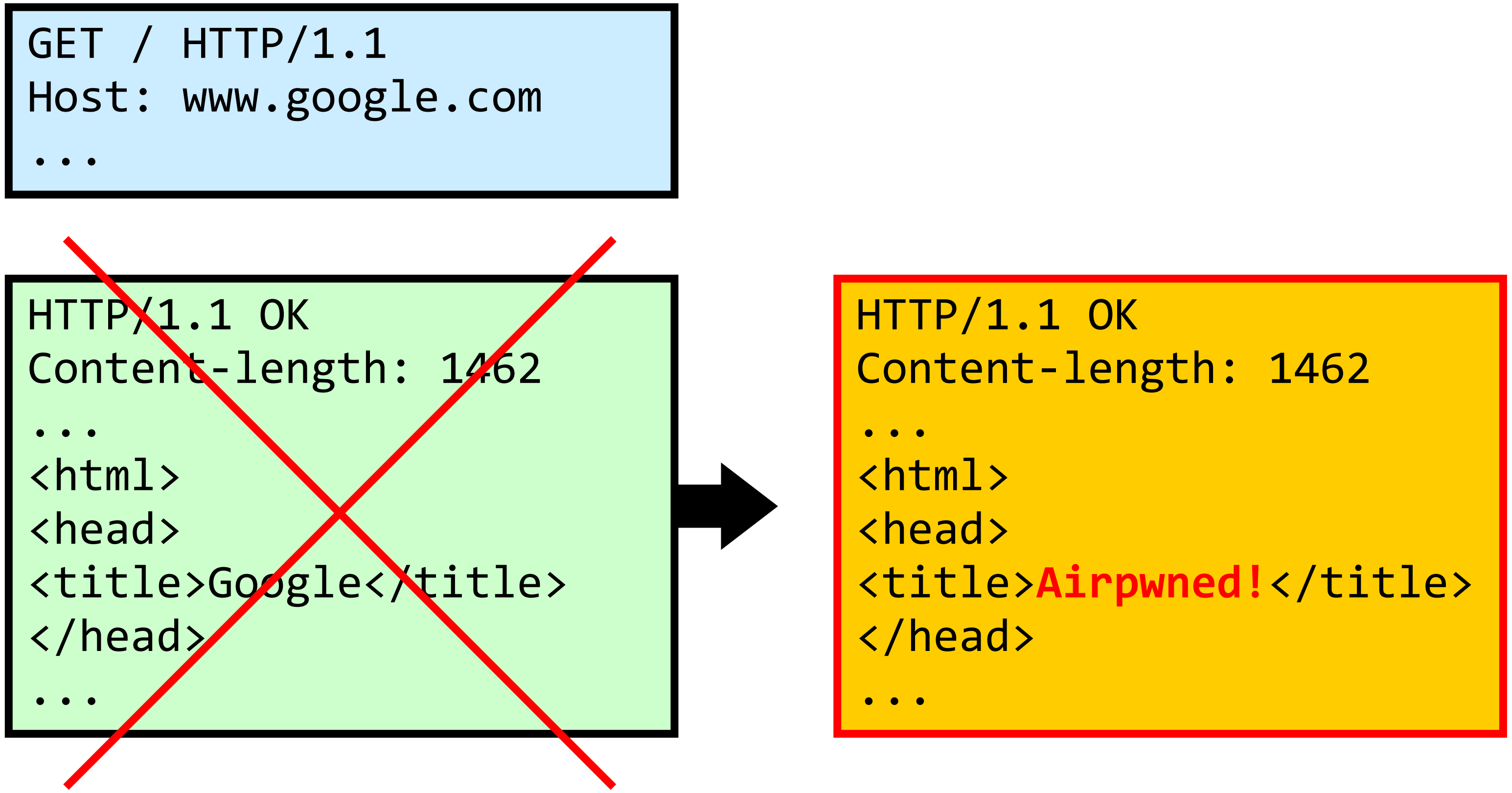
Powerful: redirect to rogue server, manipulate content, inject exploits



Airpwn

Sniffs packets and acts on interesting HTTP requests based on rules

- Beating the server’s response is easy: the server is several hops away (10s-100s ms) while the attacker is in the local WiFi network

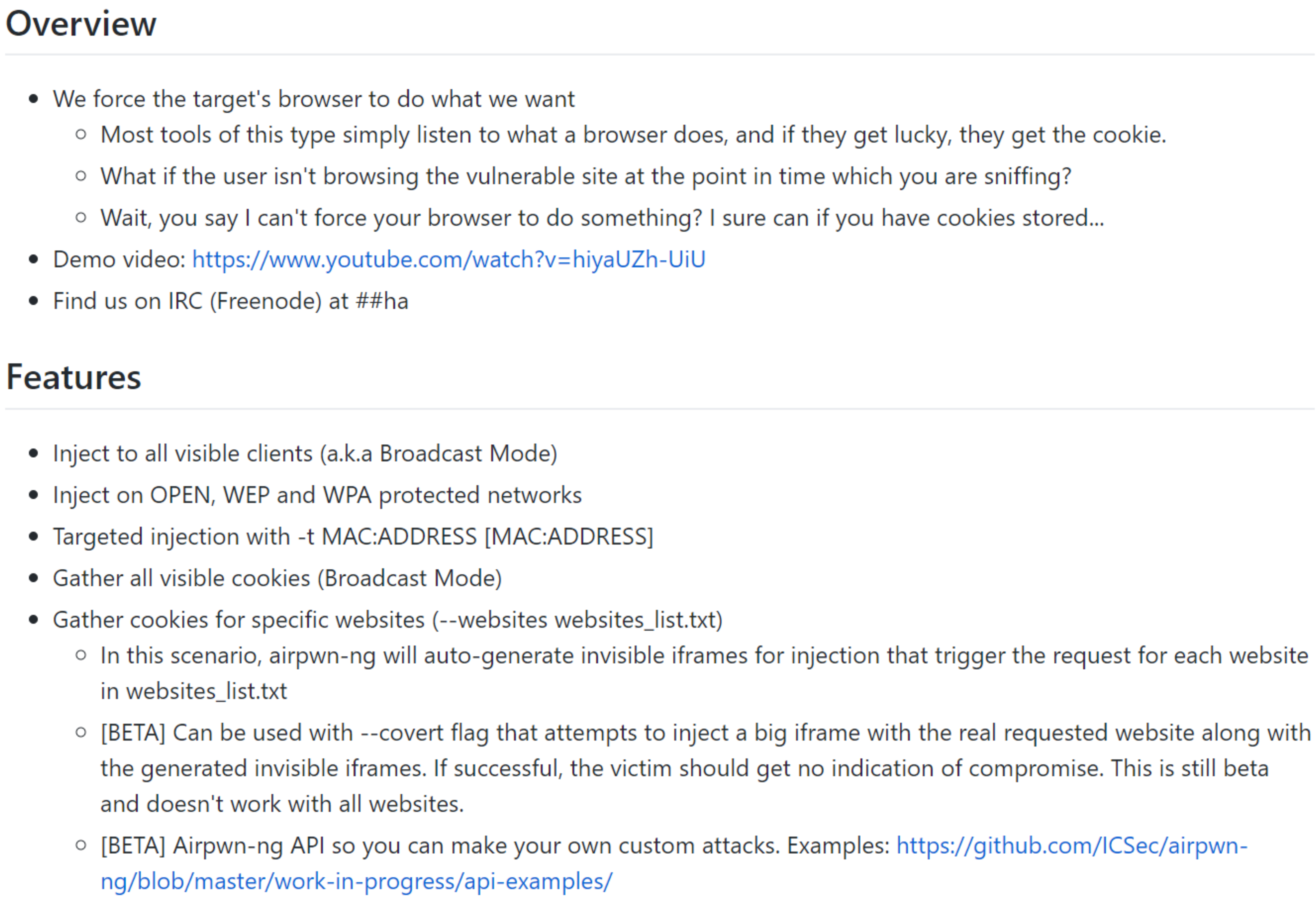


```
GET / HTTP/1.1
Host: www.google.com
...
```

```
HTTP/1.1 OK
Content-length: 1462
...
<html>
<head>
<title>Google</title>
</head>
...
```

```
HTTP/1.1 OK
Content-length: 1462
...
<html>
<head>
<title>Airpwned!</title>
</head>
...
```

airpwn-ng https://github.com/ICSec/airpwn-ng

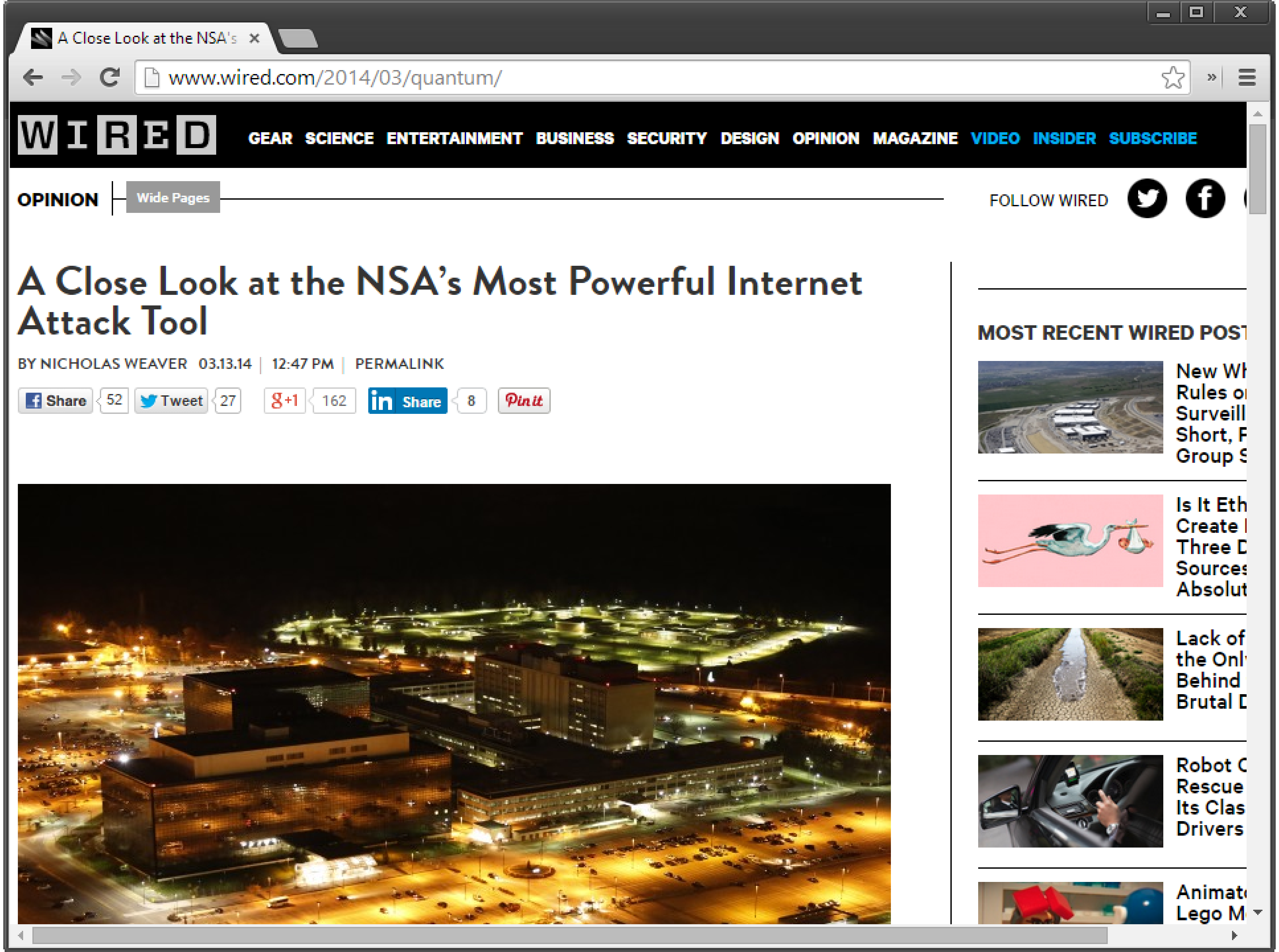


Overview

- We force the target's browser to do what we want
  - Most tools of this type simply listen to what a browser does, and if they get lucky, they get the cookie.
  - What if the user isn't browsing the vulnerable site at the point in time which you are sniffing?
  - Wait, you say I can't force your browser to do something? I sure can if you have cookies stored...
- Demo video: https://www.youtube.com/watch?v=hiyaUZh-UiU
- Find us on IRC (Freenode) at ##ha

Features

- Inject to all visible clients (a.k.a Broadcast Mode)
- Inject on OPEN, WEP and WPA protected networks
- Targeted injection with -t MAC:ADDRESS [MAC:ADDRESS]
- Gather all visible cookies (Broadcast Mode)
- Gather cookies for specific websites (--websites websites_list.txt)
  - In this scenario, airpwn-ng will auto-generate invisible iframes for injection that trigger the request for each website in websites_list.txt
  - [BETA] Can be used with --covert flag that attempts to inject a big iframe with the real requested website along with the generated invisible iframes. If successful, the victim should get no indication of compromise. This is still beta and doesn't work with all websites.
  - [BETA] Airpwn-ng API so you can make your own custom attacks. Examples: https://github.com/ICSec/airpwn-ng/blob/master/work-in-progress/api-examples/



A Close Look at the NSA's
www.wired.com/2014/03/quantum/
WIRED
GEAR SCIENCE ENTERTAINMENT BUSINESS SECURITY DESIGN OPINION MAGAZINE VIDEO INSIDER SUBSCRIBE
OPINION
Wide Pages
FOLLOW WIRED
Share 52
Tweet 27
+1 162
Share 8
Pin it

A Close Look at the NSA's Most Powerful Internet Attack Tool

BY NICHOLAS WEAVER 03.13.14 | 12:47 PM | PERMALINK

MOST RECENT WIRED POST

New Wh
Rules o
Surveill
Short, F
Group S

Is It Eth
Create
Three D
Sources
Absolut

Lack of
the Onl
Behind
Brutal D

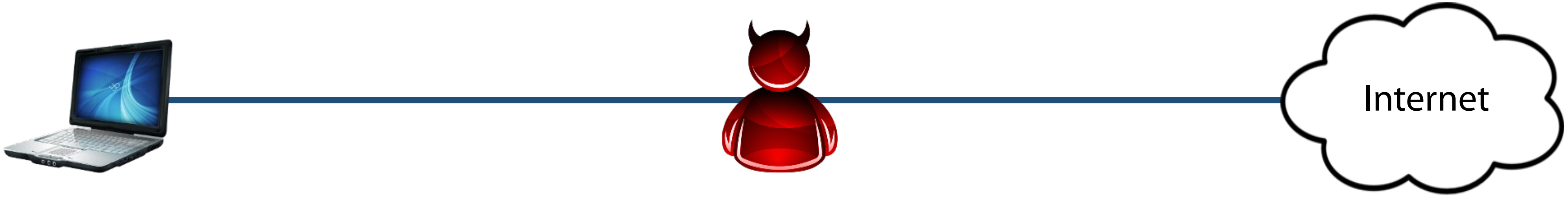
Robot C
Rescue
Its Clas
Drivers

Animat
Lego M

Man-in-the-Middle Attack

In-path attacker: can inject new and modify or drop existing packets

More powerful than an *on-path* adversary (Man-on-the-Side) who can inject new packets but cannot alter existing packets (just observe them)



Many ways to achieve an in-path position

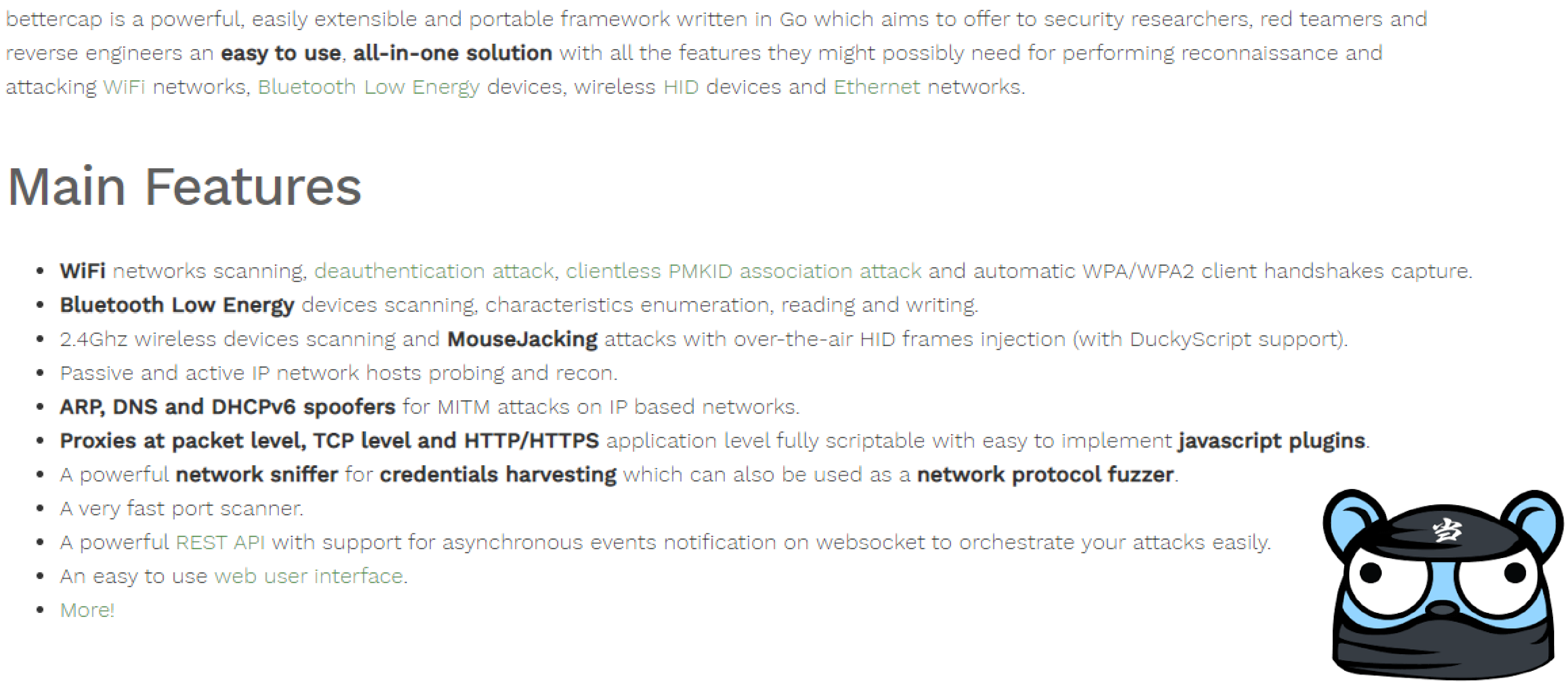
ARP poisoning

Rogue or compromised router, VPN server, firewall, gateway, …

Physically interjected network bridge or transparent/intercepting/inline proxy

Software-level interception (browser extension, parental control filter, anti-virus, …)

Bettercap https://www.bettercap.org/

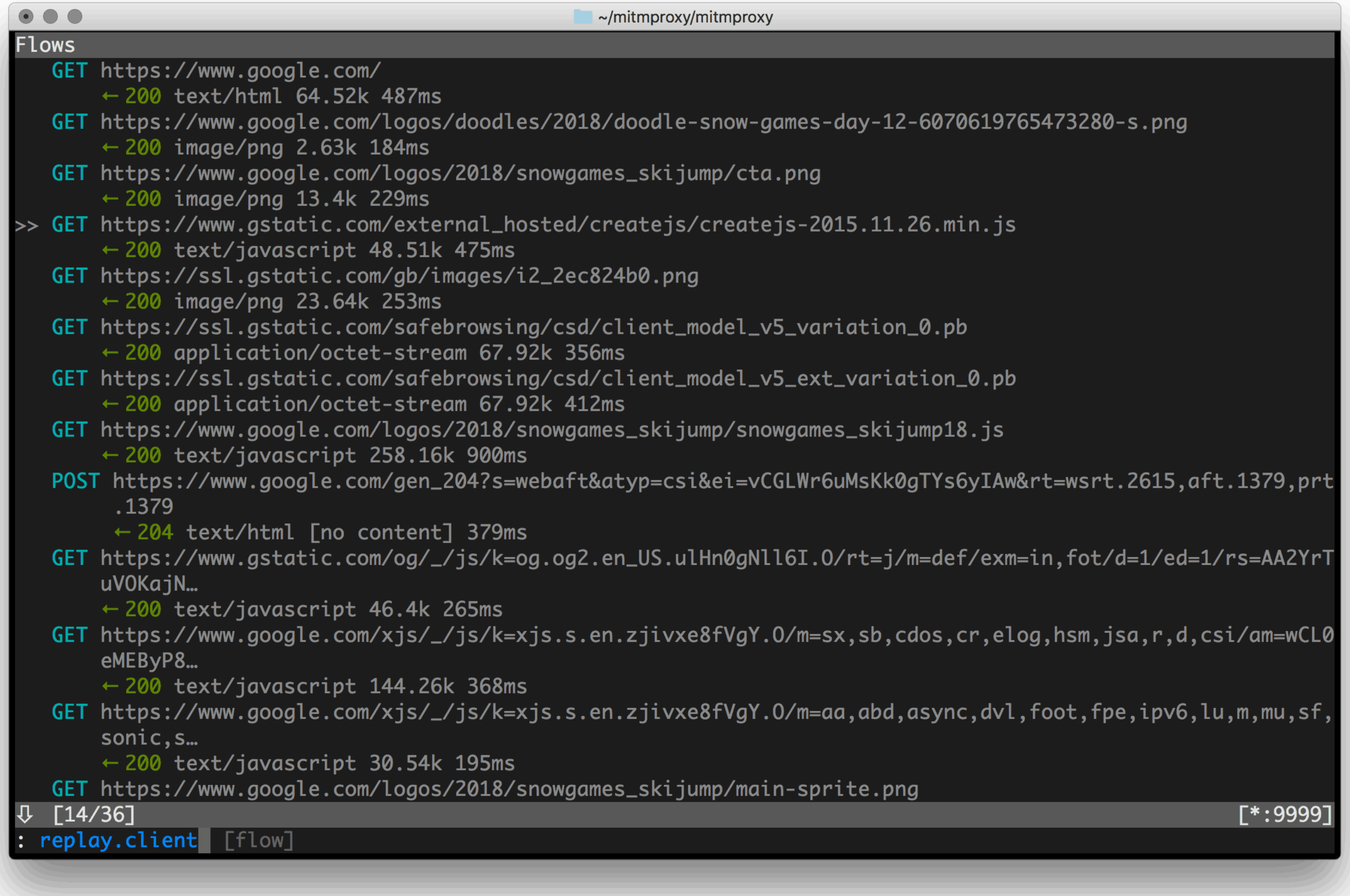


bettercap is a powerful, easily extensible and portable framework written in Go which aims to offer to security researchers, red teamers and reverse engineers an **easy to use**, **all-in-one solution** with all the features they might possibly need for performing reconnaissance and attacking WiFi networks, Bluetooth Low Energy devices, wireless HID devices and Ethernet networks.

Main Features

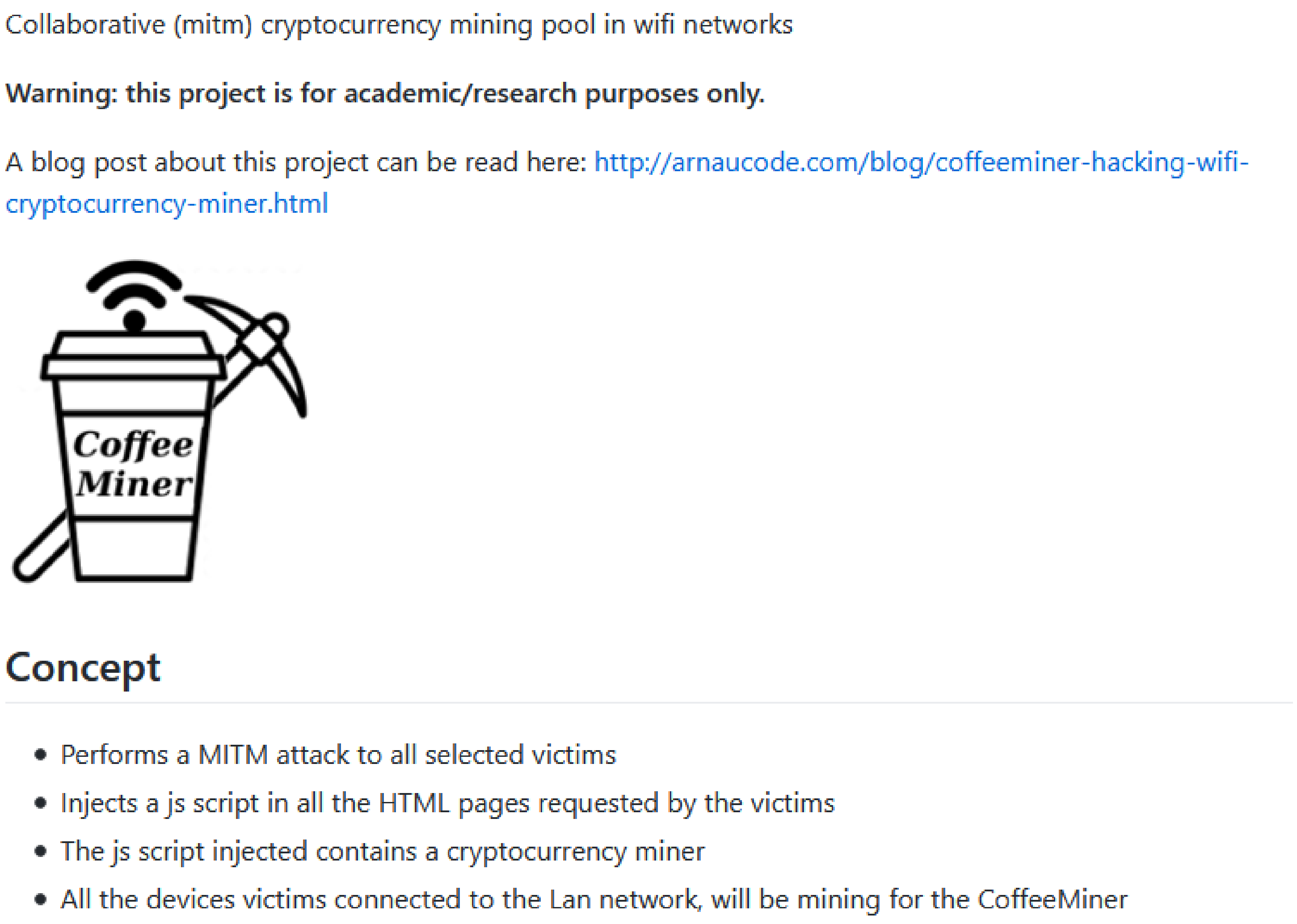
- **WiFi** networks scanning, deauthentication attack, clientless PMKID association attack and automatic WPA/WPA2 client handshakes capture.
- **Bluetooth Low Energy** devices scanning, characteristics enumeration, reading and writing.
- 2.4Ghz wireless devices scanning and **MouseJacking** attacks with over-the-air HID frames injection (with DuckyScript support).
- Passive and active IP network hosts probing and recon.
- **ARP, DNS and DHCPv6 spoofers** for MITM attacks on IP based networks.
- **Proxies at packet level, TCP level and HTTP/HTTPS** application level fully scriptable with easy to implement **javascript plugins**.
- A powerful **network sniffer** for **credentials harvesting** which can also be used as a **network protocol fuzzer**.
- A very fast port scanner.
- A powerful REST API with support for asynchronous events notification on websocket to orchestrate your attacks easily.
- An easy to use web user interface.
- More!

mitmproxy https://mitmproxy.org/



```
Flows
   GET https://www.google.com/
       ←200 text/html 64.52k 487ms
   GET https://www.google.com/logos/doodles/2018/doodle-snow-games-day-12-6070619765473280-s.png
       ←200 image/png 2.63k 184ms
   GET https://www.google.com/logos/2018/snowgames_skijump/cta.png
       ←200 image/png 13.4k 229ms
>> GET https://www.gstatic.com/external_hosted/createjs/createjs-2015.11.26.min.js
       ←200 text/javascript 48.51k 475ms
   GET https://ssl.gstatic.com/gb/images/i2_2ec824b0.png
       ←200 image/png 23.64k 253ms
   GET https://ssl.gstatic.com/safebrowsing/csd/client_model_v5_variation_0.pb
       ←200 application/octet-stream 67.92k 356ms
   GET https://ssl.gstatic.com/safebrowsing/csd/client_model_v5_ext_variation_0.pb
       ←200 application/octet-stream 67.92k 412ms
   GET https://www.google.com/logos/2018/snowgames_skijump/snowgames_skijump18.js
       ←200 text/javascript 258.16k 900ms
   POST https://www.google.com/gen_204?s=webaft&atyp=csi&ei=vCGLWr6uMsKk0gTYs6yIAw&rt=wsrt.2615,aft.1379,prt
        .1379
        ←204 text/html [no content] 379ms
   GET https://www.gstatic.com/og/_/js/k=og.og2.en_US.ulHn0gNll6I.O/rt=j/m=def/exm=in,fot/d=1/ed=1/rs=AA2YrT
       uVOKajN…
       ←200 text/javascript 46.4k 265ms
   GET https://www.google.com/xjs/_/js/k=xjs.s.en.zjivxe8fVgY.O/m=sx,sb,cdos,cr,elog,hsm,jsa,r,d,csi/am=wCL0
       eMEByP8…
       ←200 text/javascript 144.26k 368ms
   GET https://www.google.com/xjs/_/js/k=xjs.s.en.zjivxe8fVgY.O/m=aa,abd,async,dvl,foot,fpe,ipv6,lu,m,mu,sf,
       sonic,s…
       ←200 text/javascript 30.54k 195ms
   GET https://www.google.com/logos/2018/snowgames_skijump/main-sprite.png
⇩ [14/36]                                                                                          [*:9999]
: replay.client  [flow]
```

CoffeeMiner https://github.com/arnaucube/coffeeMiner



Collaborative (mitm) cryptocurrency mining pool in wifi networks

Warning: this project is for academic/research purposes only.

A blog post about this project can be read here: http://arnaucode.com/blog/coffeeminer-hacking-wifi-cryptocurrency-miner.html

Concept

- Performs a MITM attack to all selected victims
- Injects a js script in all the HTML pages requested by the victims
- The js script injected contains a cryptocurrency miner
- All the devices victims connected to the Lan network, will be mining for the CoffeeMiner

Hands-on Session