]HackingTeam[

Project X Mass interception of encrypted connections

What?

SSL/TLS interception

TOR interception

...a thorny path

Public Key Pinning avoids rogue CA to sign certs

Google and Facebook actively search for rogue CA signed certs (no more governmental signing: France, India)

HSTS enforces https on a variety of hardcoded website (no more SSLstrip)



HTTPS Everywhere enforces https and could send rogue certificates to the EFF SSL Observatory

No solution for sniffing TOR available by now on the market

The Solution

Place an in-line Active Probe in the ISP's network



Exploit the target transparently by injecting a browser-based exploit while he's surfing the web (http)



Insert a trusted root CA certificate(s) for MITM

Redirect first TOR hop

Decrypt and Decode the traffic!

More in depth

Deployment phases

Identification Inoculation/Marking SSL MITM (only for SSL) Decoding Maintenance

Identification Phase

Uniquely identify a target on the internet (cookies, browser strings, etc.)

Create a profile for each target to know if it's exploitable

Avoid exploiting the same target "too much"

Avoid exploiting a target with "problematic"



Inoculation Phase

HTTP man-in-the-middle (transparent proxy)

Browser based exploits (all platforms)

Local to root exploits (sandbox escape)

Methods to insert root CA cert(s) into the keystore

Methods to divert TOR first HOP]HackingTeam[

Insert a "watermark" in target's environment to uniquely identify "inoculated" targets during SSL connections

Setting different TOR's SOCKS password

SSL MITM Phase

Transparent proxy performing SSL MITM only on "marked" targets

Serve the right certificate Avoid exposing fake certs Avoid checks to detect fake certs

Decoding Phase

A good partner with a consolidated decoding technology

Maintenance Phase

Automatic test to check if the certs are invalidated (Customer side)



Maintenance Phase

Automatic check for exploits effectiveness

Automatic check for inoculation phase (HT side)



Challenges

Identification Phase

Targets using multiple browsers Targets behind routers (NAT) Targets behind a TCP Proxy Targets changing IP address often



Inoculation Phase

Build or Buy exploits for several platforms

Write shellcodes to insert root CA certs

Write shellcodes to modify TOR environment

Marking the target cipher suites list Using client-side certificate (both good but fragile)



IP-to-Target Mapping Less reliable Same problems as Identification



Mixed approach is possible fake https image request host file modification

Marking must survive browser/os upgrades!

SSL MITM Phase

Find an appliance to handle the inline traffic

(no single point of failure)

SSL MITM Phase

Pay attention to Extended Validation certificate

Pay attention to EFF SSL Observatory

Pay attention to Trust Assertion for Cert Keys (TACK)

Decoding Phase

Where do we send sniffed traffic??

Feasibility Matrix

	Windows		OSX	Linux	iOS	Android
Exploit	IE	FF	Safari	Firefox	Safari	Browser
	Chrome		Chrome	Chrome	Chrome	Chrome
Root				Not needed (per user)		
Cert						
Finger (hello)	IE	FF	Safari	Firefox	Safari	Browser
	Chrome		Chrome	Chrome	Chrome*	Chrome
Finger (client cert)	IE	FF	Safari	Firefox	Safari	Browser
	Chrome		Chrome	Chrome	Chrome**	Chrome

]HackingTeam[

* Does not trust local CA certs ** Does not support client certs

Weak Points

Heavily relies on browser remote-toroot exploit availability

TOR manipulation is possible only through clear-text traffic

Browser/OS vendors may change parameters we use for identification

Weak Points

Certificate revocation leads to target loss

(impact reduced by using several certificates)

AV may detect our shellcodes (...btw no target loss)

Mass deployment increases the risk of leaking [HackingTeam]

And finally...

Strengths

Our solution bypasses certificate pinning since it uses a custom CA "manually" installed!!!

Our solution bypasses HSTS

Strengths

Our solution bypasses active MITM detection (France should have known it)

Our solution is the only way to intercept TOR traffic at the moment

Decisions

Hardware for the probes

iBypass TAP General purpose server

Modifying an existing SSL appliance

Decoding the traffic

Once decrypted the traffic must be decoded:

Forwarding to an existing monitoring center using standard protocols

Create a turn-key solution with a

"passive" partner]HackingTeam[

Resources

Time

First Minimal Demo: 2 months

First POC: 9 months

First Deployment: 15 months



Human

2 Exploit/Shellcode Developers

1 Network/Probe Developer

1 ISP SysAdmin (consultancy)

2 Backend/Logic Developers

1 Tester

In house but allocated

Future development

Other over-SSL protocols

Support for imaps, pops, etc.



RCS integration

Keep a DB of exploitable targets

Exploit them again to install RCS

Integration through the RCS Console (...or HT Monitoring Center)

Layer 3 MITM

Just mangle the handshake and forward the rest of the connection to improve performances

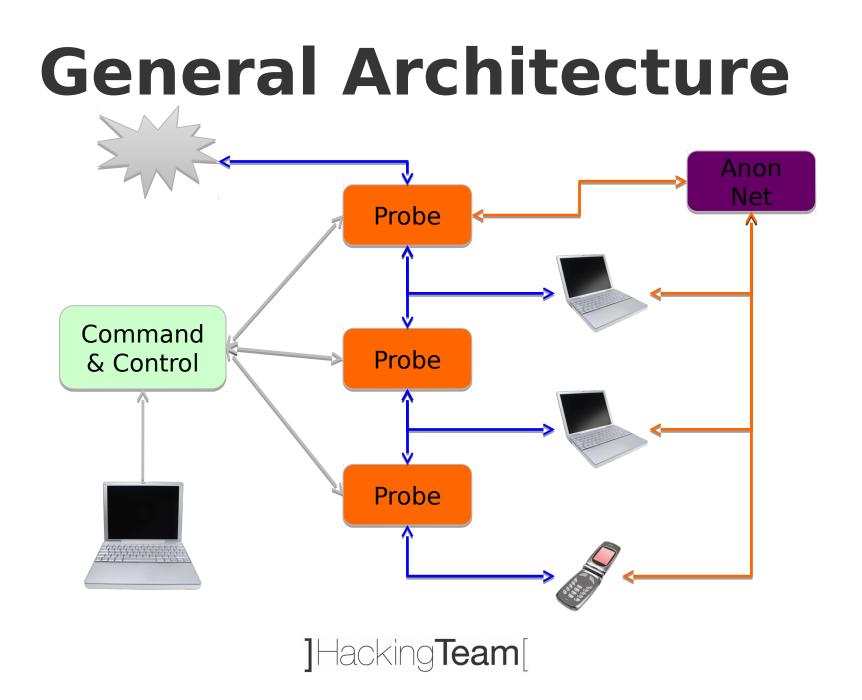


SSL TLS key dump

Just save the SSL keys and pass it to an SSL offload decrypter for maximum performances



Technical details



Command & Control

Ruby on Rails HTML5 interface Fault tolerant & scalable



Command & Control

Exploit repository (auto update from HT) Attack rules (global or per probe) Active/identified targets in realtime Probes configuration / update Anon network configuration Global system monitoring

Anon Network

Used to forward connections to public addresses to the probes

Useful if we set a socks/proxy in the target and the target is nomadic

Probe Architecture

