MiTM'ing Secured Communications in Automotive Apps

Month 10, 2017 |

Subtitle

Ben Gardiner, Cloakware by Irdeto

Principal Security Engineer, GENIVI Alliance

About Me

- https://www.linkedin.com/in/ben0l0gardiner/

- 4 years experience in software security, 9+ years in embedded software design/development

GENIVI®

About Irdeto

irdeto
Building a Secure Future.™

- Irdeto is a pioneer in digital platform and application security.
- ~50 years of security expertise.
- It's software security technology and cyber services:
  - Protect more than 5B devices & applications.
  - Protect against cyberattacks for some of the world's best known brands.

cloakware™ by irdeto
for automotive

- Cloakware for Automotive by Irdeto helps automakers and tier-one suppliers protect their brand and save cost in the battle against cybercriminals by creating a secure, tamper-proof environment for vehicle software.

GENIVI®

Overview

- Man In The Middle
  - Background Information
    - SSL
    - Certificate Pinning
    - Integrity Verification
  - Protection Types
    - 1-6 levels
  - Attacks
    - Difficulties
      - In Field
      - At Home
- Tools
  - Proxy
    - Mitmproxy
    - Burp
    - OWASP ZAP
  - Hooking
    - Frida
    - JustTrustMe
- Examples
    - In Field Automotive Apps
      - Anonymous
- Appendix
  - Mitmproxy Setup

GENIVI®

What is MiTM?

- **Man in The Middle** attacks (MiTM / MITM)

- Intercepting and possibly modifying communications between two parties
  - without either party detecting the interception
  - assumed to mean bypassing encryption and authentication also -- when encrypted and/or authenticated communications are involved

- Where or not data is modified yields two *flavors* of the attack:
  - Siphon (no data modification)
  - Proxy

- Applicable to nearly all transport-layer and application-layer protocols in some form or another. In the following we will be discussing HTTP/HTTPS

Agenda

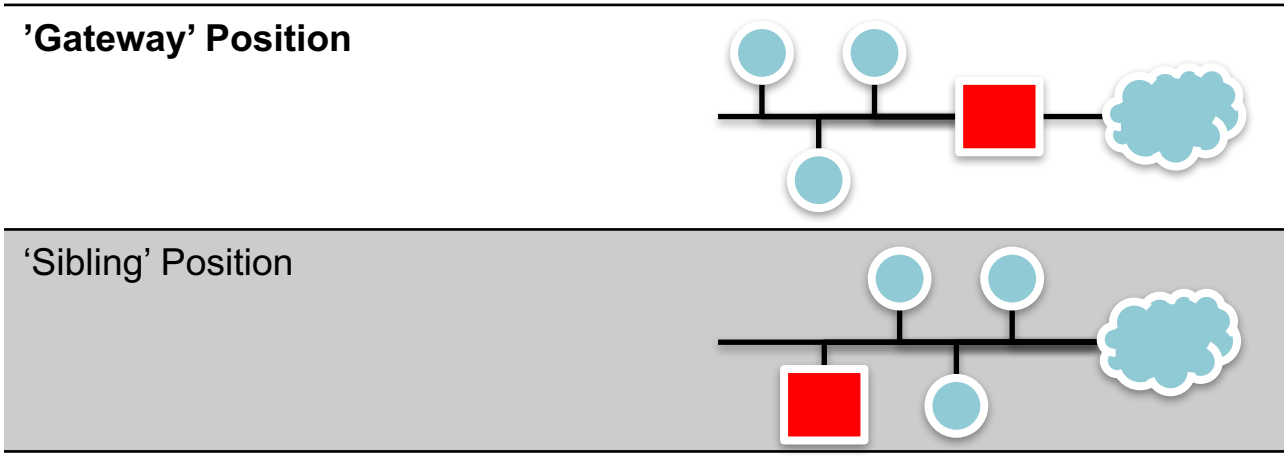
- Examine the increasing levels of protection in HTTP/HTTPS communications
  - In mobile apps (focus on Android),
  - They are borne from mitigations against attacks

"Type"	Trust
Type 1 	Trust anything (no SSL/TLS)
Type 2 	Trust any valid certificate
Type 3 	Trust any root-CA in OS "Trust Store"
Type 4 	Trust only (pin) the pub key of certificate
Type 5	Trust only (pin) the pub key of cert. signer
Type 6 	Pinning and Integrity Verification

GENIVI®

Type 1: Just HTTP (no S: SSL/TLS)

- No encryption, integrity or authentication in place
- A valid ‘position’ in the network is required
  - This depends on the network medium (see next slide)
- First; terminology for topologies



GENIVI®

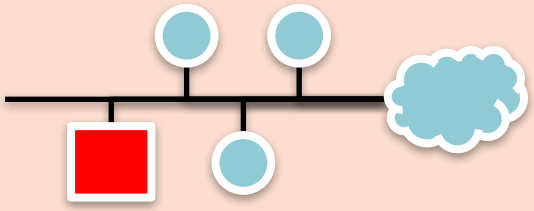
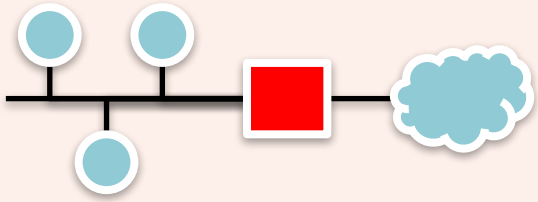
Type 1 Defeat: Traffic Capture

- To view the unencrypted traffic attackers need only to receive the packets

Medium	Visibility from Sibling Position	Visibility from Gateway Position
Ethernet, Hub	Yes	Yes
Ethernet, Switch	Yes, with ARP-poisoning attack	Yes
Wi-fi	Yes	Yes, with Wi-fi Pineapple
Wi-fi +WEP	Yes	Yes, with Pineapple
Wi-fi +WPA2 PSK	Yes, with PSK known and captured WPA2 negotiation (e.g. de-auth attacks)	Yes, with Pineapple and PSK known
Wi-fi +WPA2 PSK Enterprise	No	Yes, with Pineapple and PSK known

GENIVI®

Discussion: Difficulty of Achieving Positions

		Difficulty in-field	… at-home
‘Sibling’ Position		Trivial (Ethernet, WEP) to Moderate (WPA2 PSK)	Trivial
’Gateway’ Position		Easy (Wi-fi) to Unfeasible (WPA2 PSK Enterprise)	Trivial

GENIVI®

Type 1 Examples in Automotive Apps

- For all the applications surveyed, none. Which is good
- Minor exceptions: included some libraries pulled-in transitively by automotive apps
  - Remember to audit and/or test the behavior of your 3rd party components

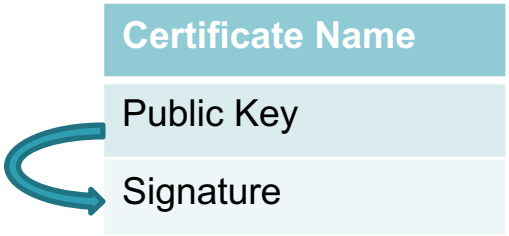
GENIVI®

Discussion: Encryption without Authentication?

- It is possible to setup encrypted connections with a very-large pre-shared secret.
  - Not really feasible to deploy these systems at scale
- It is technically possible to setup negotiated encrypted channels at scale without authentication. c.f. TCPInc
  - These systems must negotiate their encryption keys during setup of the channel
  - Anyone eavesdropping on the channel setup (see Type 1 defeats) can discover the encryption keys
- Enter public-key crypto…
  - Dual-purpose the private keys used for identity also for setting up channel encryption immune to eavesdropping
- Reminder public key crypto can: enable a proof that a party has the private part of any public key

Type 2: Trust ‘any’ Endpoint With Certificate

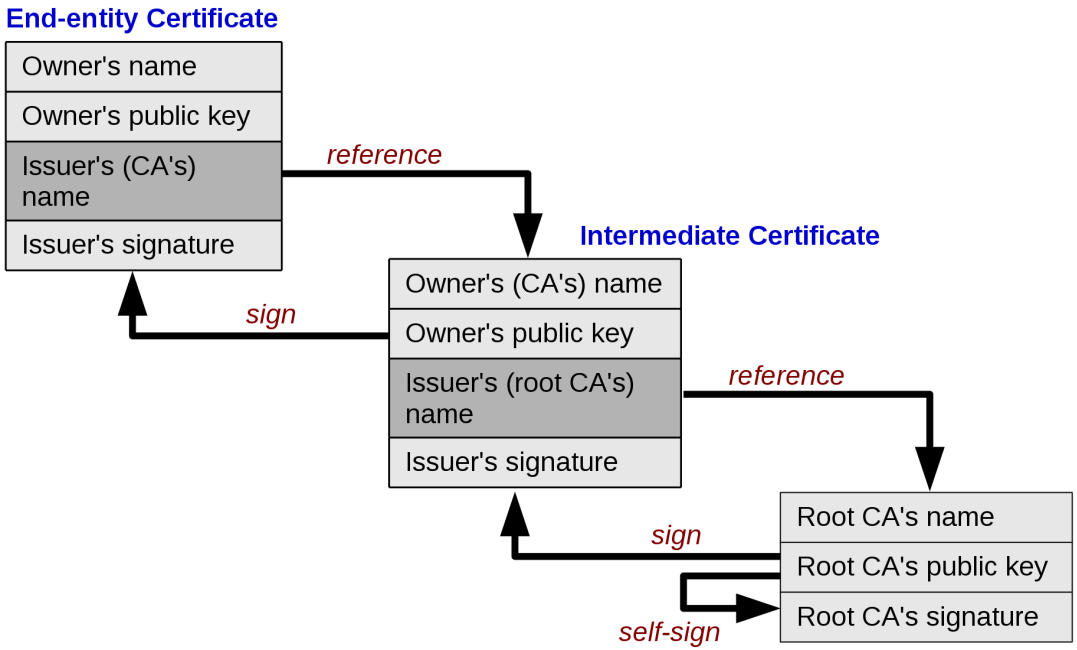
- Verify, in client HTTPS code, that the server has a valid certificate, any certificate
  - Validity here means signed by anything; including self-signed
  - Self-signed: signing the public key of the certificate with the private key of the same certificate



GENIVI®

Discussion: What are Certificates?

- Colloquially, for this presentation: the public keys and a signature of it against another certificate
  - In a chain



https://commons.wikimedia.org/wiki/File:Chain_of_trust.svg

GENIVI®

Type 2 Defeat: Burp et. al.

- Proxy the HTTPS traffic; supply your own certificate to the clients
- Requires a 'gateway' position on the network (see Type 1)
  - Achievable in many in-field situations and also in all at-home situations
- Many tool options: Burp, OWASP ZAP, mitmproxy

Attack	Difficulty in-field	… at-home
Proxy Https Traffic	Moderate	Trivial

GENIVI®

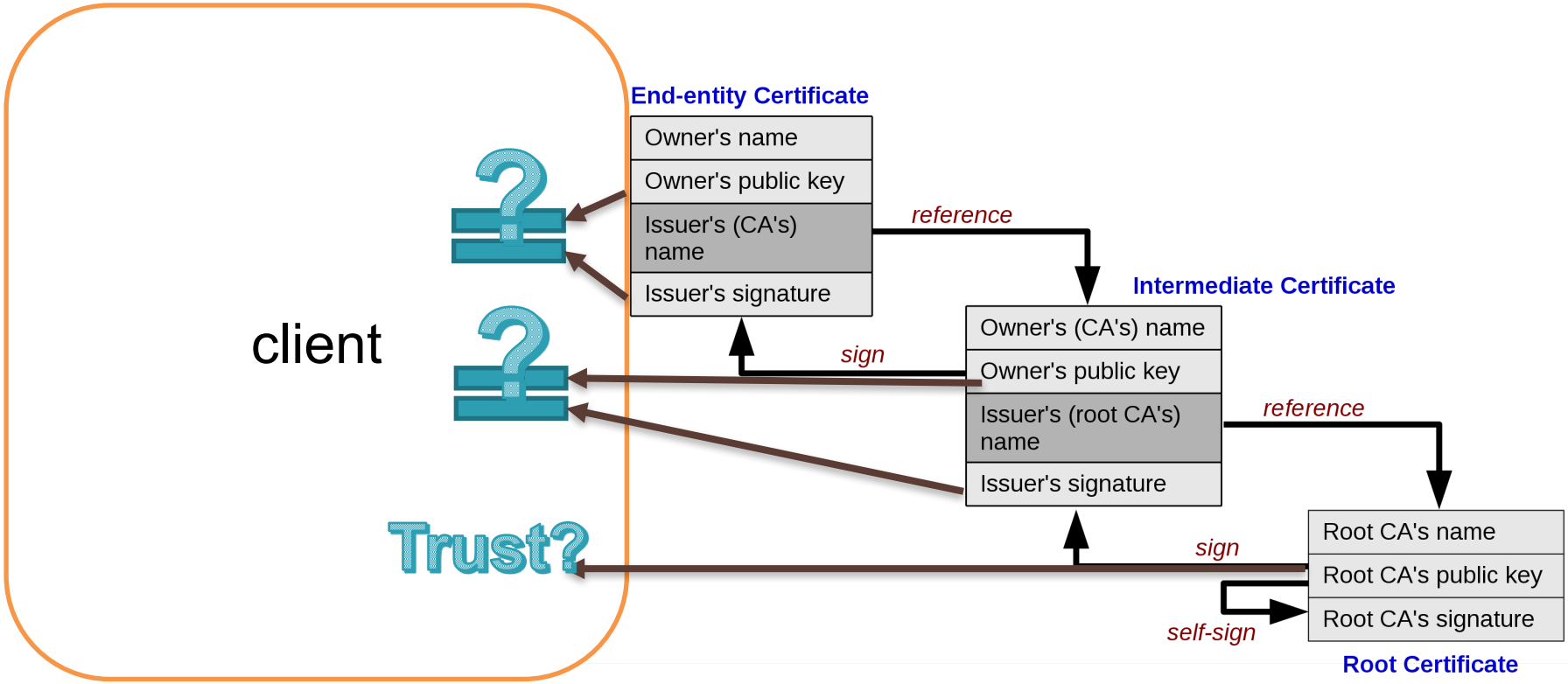
Type 2 Example in Automotive Apps

- One example, non-critical use

- We also found many libraries with this pattern – appeared to be unused at runtime
  - Reminder: check what your third-party libraries are doing at runtime

GENIVI®

Discussion: Authentication via Chains of Trust



https://commons.wikimedia.org/wiki/File:Chain_of_trust.svg

GENIVI®

Aside: HSTS

- Protects a website against these sorts of attacks, informing the user that this website is only to be accessed in HTTPS protocol

- Although certain sites are preloaded, it is possible to mitm the first connection to the site, and prevent the browser from receiving the HSTS flag

- This may also prevent cookie-downgrading attacks through a similar mechanism

Discussion: Authentication in Mobile OS and the Trust-Store.

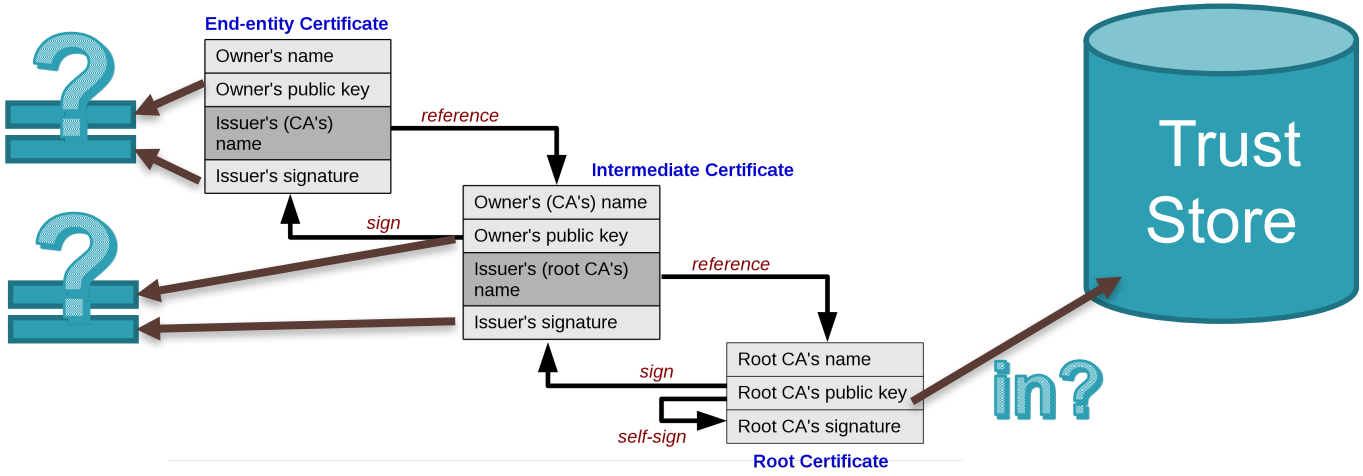
- Android devices come preloaded with a list of trusted root Certificate Authorities that are inherent as trusted anchors. Anything signed by them will not throw security exceptions when the server is accessed

- iOS similarly comes with a preloaded list of trusted root CAs

GENIVI®

Type 3: Trust the Trust-Store

- All chains must end with a valid root CA certificate found in the Trust Store



https://commons.wikimedia.org/wiki/File:Chain_of_trust.svg

GENIVI®

Type 3 Defeat: Add to the Trust-Store

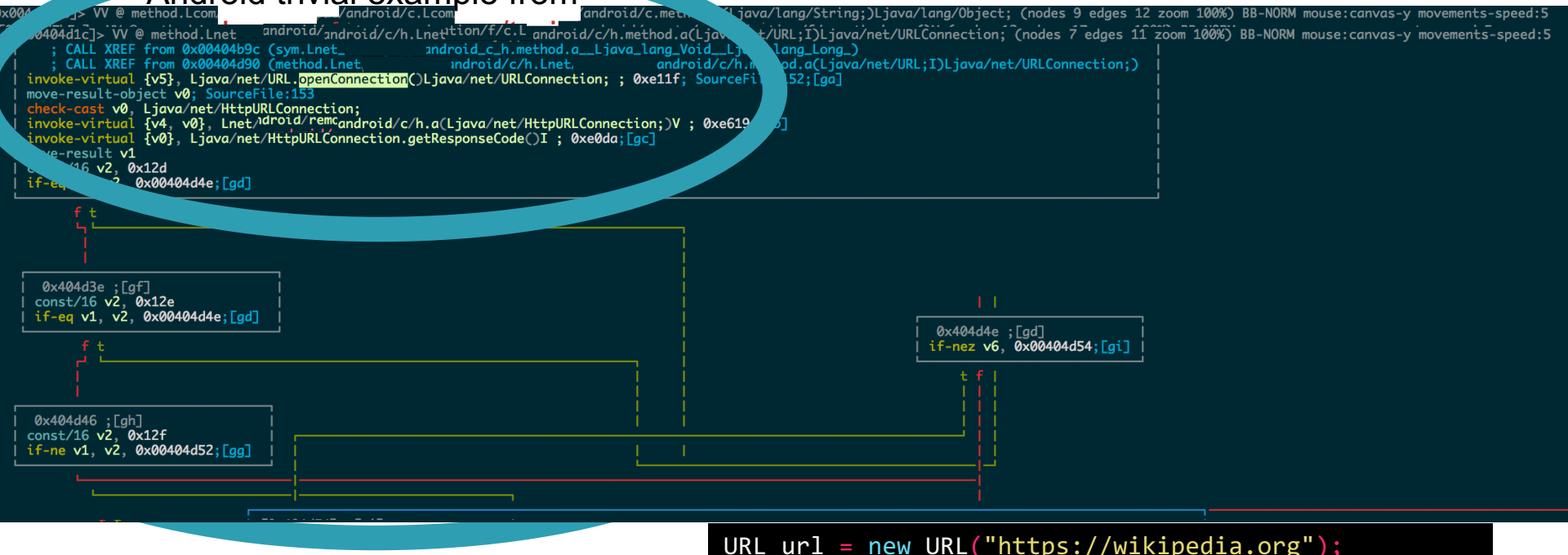
- Custom user certs can be added,
- If root access is granted, can insert certs without user knowledge
- Improperly signed Certificates
- Private key leaks

Attack	Difficulty in-field	… at-home
Improperly signed Certificates	Difficult	Difficult
Private key leaks	Difficult	Difficult
Custom certificate installation	Moderate	Easy

GENIVI®

Type 3 Examples in Automotive Apps

- The vast majority of apps surveyed relied on the system Trust Store
- Default in Android and iOS; therefore many ways this type gets implemented
  - Android trivial example from




```
URL url = new URL("https://wikipedia.org");
URLConnection urlConnection = url.openConnection();
InputStream in = urlConnection.getInputStream();
copyInputStreamToOutputStream(in, System.out);
```

GENIVI®

Deep-dive: HOWTO Setup mitmproxy for Android Testing

- Setting up environment
  - Longsword.sh Setup Scripts
  - Iptables modification
- Starting Proxy
  - Installing Certificate
  - Connection to Proxy
- Monitoring output

GENIVI®

Setting up Environment: Longsword init

```
deps () {
  printf "\nDownload the openvpn files to /etc/openvpn and add your secrets\s"
  set -x
  apt-get install hostapd isc-dhcp-server haveged
  systemctl enable hostapd
  systemctl enable haveged
  systemctl start haveged
  mkdir -p /etc/hostapd
  rm -f /etc/hostapd/hostapd.conf /etc/dhcp/dhcpd.conf /etc/default/isc-default-server
  ln -s "$PWD/hostapd.conf" /etc/hostapd/hostapd.conf
  ln -s "$PWD/dhcpd.conf" /etc/dhcp/dhcpd.conf
  ln -s "$PWD/isc-dhcp-server" /etc/default/isc-default-server
}


init () {
  WLAN="wlan0"
  [ -n "$1" ] && WLAN="$1"

  iptables-restore iptables.save
  sysctl -w net.ipv4.ip_forward=1 2>&1 >/dev/null

  #transparent proxying on $WLAN assuming the clients have custom gateway set, disable ICMP redirects
  echo 0 > /proc/sys/net/ipv4/conf/$WLAN/send_redirects

  systemctl restart hostapd

  ip link set "$WLAN" type wlan
  ifconfig "$WLAN" 192.168.25.1 netmask 255.255.255.0
  ip link set "$WLAN" up

  systemctl restart isc-dhcp-server

}
```

GENIVI®

Setting up Environment: Iptables

```
# Generated by iptables-save v1.6.0 on Tue Jul 19 08:35:14 2016
*filter
:INPUT ACCEPT [89463:118052725]
:FORWARD ACCEPT [3547:1414213]
:OUTPUT ACCEPT [50141:6008962]
COMMIT
# Completed on Tue Jul 19 08:35:14 2016
# Generated by iptables-save v1.6.0 on Tue Jul 19 08:35:14 2016
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
#Victims
-A PREROUTING -i wlan0 -p tcp -m tcp -s 192.168.25.100 --dport 443 -j REDIRECT --to-ports 8080
-A PREROUTING -i wlan0 -p tcp -m tcp -s 192.168.25.100 --dport 80 -j REDIRECT --to-ports 8080
-A PREROUTING -i wlan0 -p tcp -m tcp -s 192.168.25.51 --dport 443 -j REDIRECT --to-ports 8080
-A PREROUTING -i wlan0 -p tcp -m tcp -s 192.168.25.51 --dport 80 -j REDIRECT --to-ports 8080
#.4
-A PREROUTING -i wlan0 -p tcp -m tcp -s 192.168.25.4 --dport 8080 -j REDIRECT --to-ports 8080
-A PREROUTING -i wlan0 -p tcp -m tcp -s 192.168.25.4 --dport 7758 -j REDIRECT --to-ports 8080
-A PREROUTING -i wlan0 -p tcp -m tcp -s 192.168.25.205 --dport 443 -j REDIRECT --to-ports 8080
-A PREROUTING -i wlan0 -p tcp -m tcp -s 192.168.25.205 --dport 80 -j REDIRECT --to-ports 8080
#The attacker
-A PREROUTING -i wlan0 -p tcp -m tcp -s 192.168.25.5 --dport 443 -j REDIRECT --to-ports 8081
-A PREROUTING -i wlan0 -p tcp -m tcp -s 192.168.25.5 --dport 80 -j REDIRECT --to-ports 8081
-A POSTROUTING ! -o lo -j MASQUERADE
COMMIT
# Completed on Tue Jul 19 08:35:14 2016
```

GENIVI®

Starting Mitmproxy

Most common starting arguments: `mitmproxy –T –host –anticache`

-T – transparent mode: with iptables config

-host : In the current setup of mitmproxy, it is in gateway form

-anticache: this allows a verbose look at server interactions

Other Helpful Commands:

-z : Convince servers to send uncompressed data

-replace PATTERN : Replacing server response that matches regex

GENIVI®

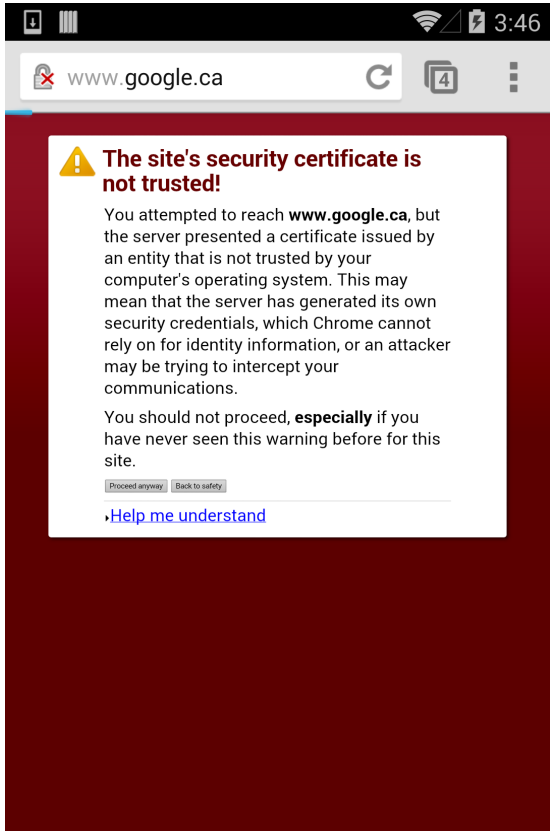
Details

```
2017-09-28 10:53:21 GET http://www.cnn.com/
                    ← 200 OK text/html 29k 460ms
                  Request                    Response                    Detail
Server Connection:
      Address           151.101.21.67:80
      Resolved Address  151.101.21.67:80
Client Connection:
      Address  192.168.25.205:38079
Timing:
      Client conn. established    2017-09-28 10:53:20.410
      First request byte          2017-09-28 10:53:21.426
      Request complete            2017-09-28 10:53:21.464
      Server conn. initiated      2017-09-28 10:53:21.837
      Server conn. TCP handshake  2017-09-28 10:53:21.849
      First response byte         2017-09-28 10:53:21.863
      Response complete           2017-09-28 10:53:21.886

[3/152]  [anticache:showhost]                                    ?:help q:back [*:8080]
Warn: 192.168.25.205:42834: Error in HTTP connection: TcpDisconnect('[Errno 32] Broken pipe',)
```

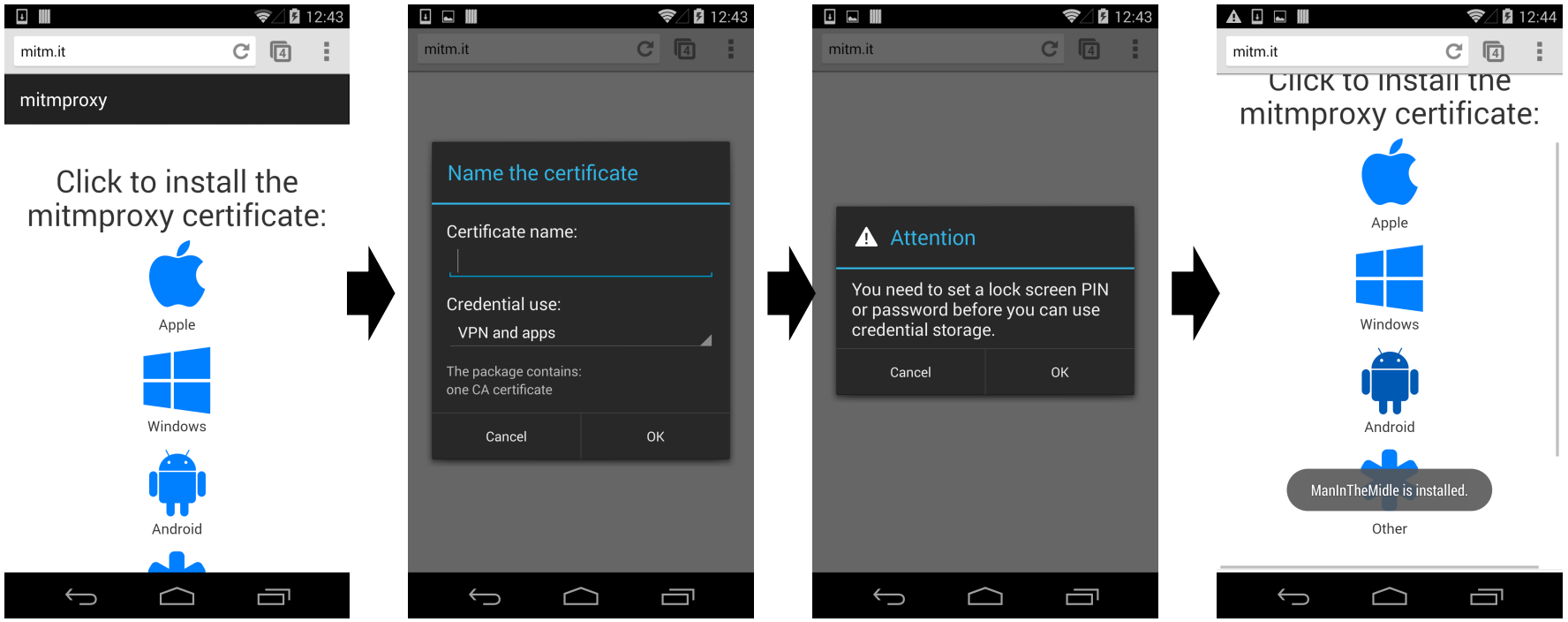
GENIVI®

javax.net.ssl.SSLHandshakeException: java.security.cert.CertPathValidatorException: Trust anchor for certification path not found.
… 18 more
Caused by: java.security.cert.CertificateException: java.security.cert.CertPathValidatorException: Trust anchor for certification path not found.
 ... 17 more
Caused by: java.security.cert.CertPathValidatorException: Trust anchor for certification path not found.
 ... 22 more



GENIVI®

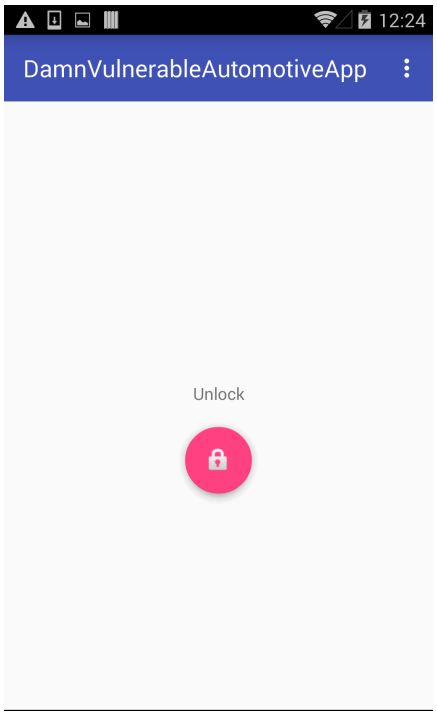
Installing the Android Certificate

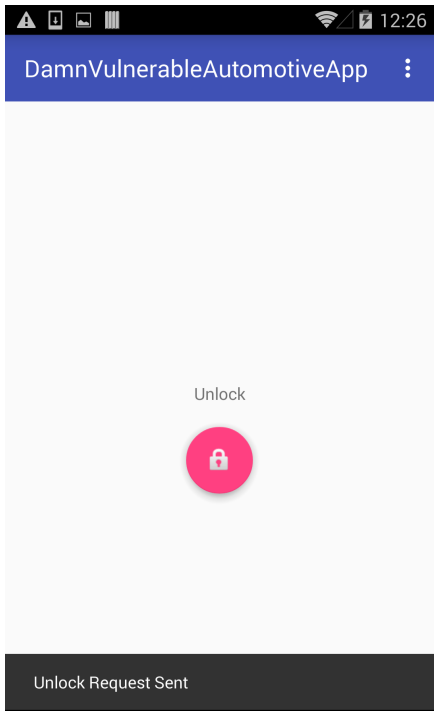


GENIVI®

Sending ssl data to arbitrary web page

```
fab.setOnClickListener((view) → {
        new RequestTask().execute("https://golang.org", "DVKey: GETPWNED");
        Snackbar.make(view, "Unlock Request Sent", Snackbar.LENGTH_LONG)
                .setAction("Action", null).show();
});
```

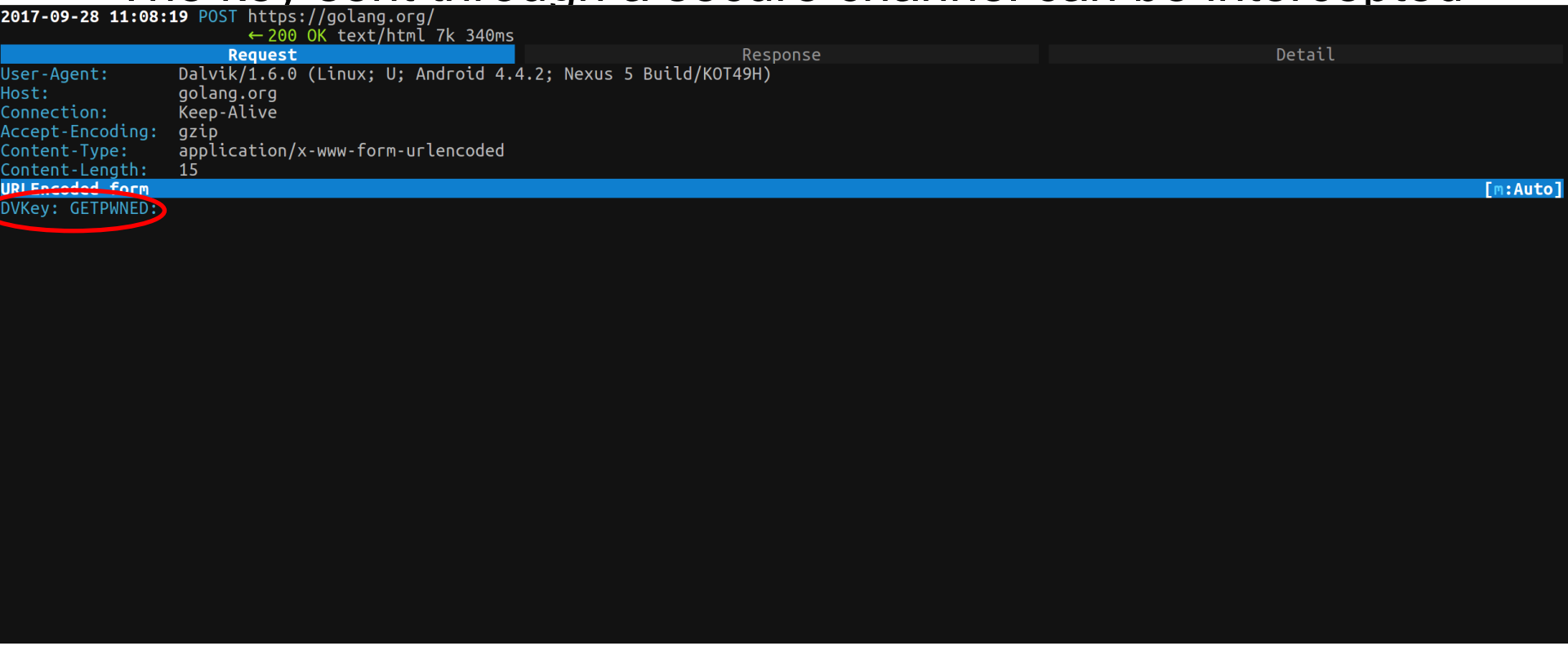




GENIVI®

Monitoring Output

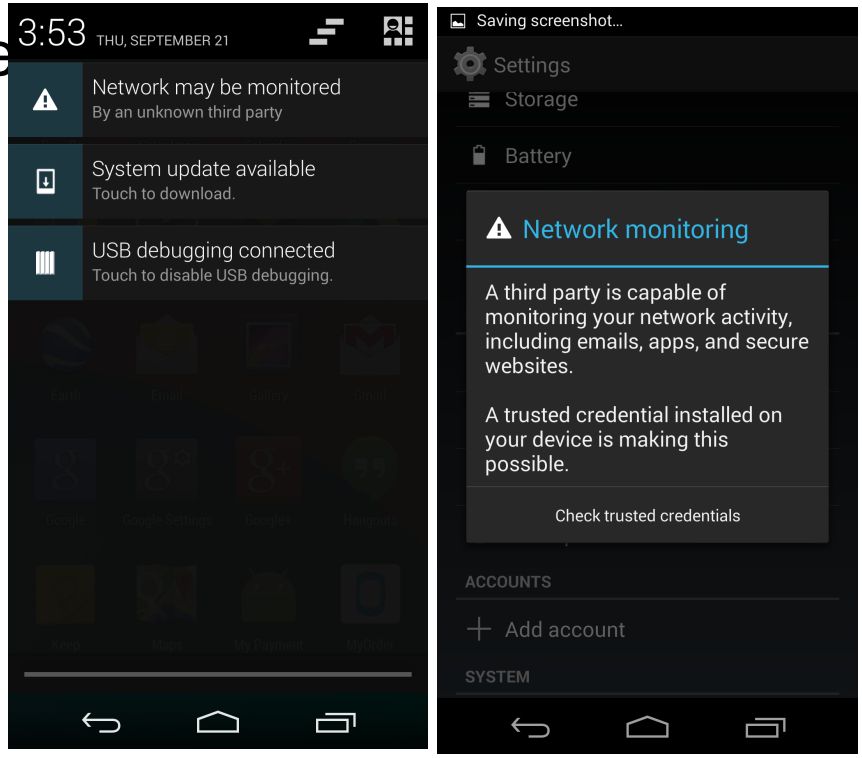
The key sent through a secure channel can be intercepted



GENIVI®

The default Mitigation: Notification

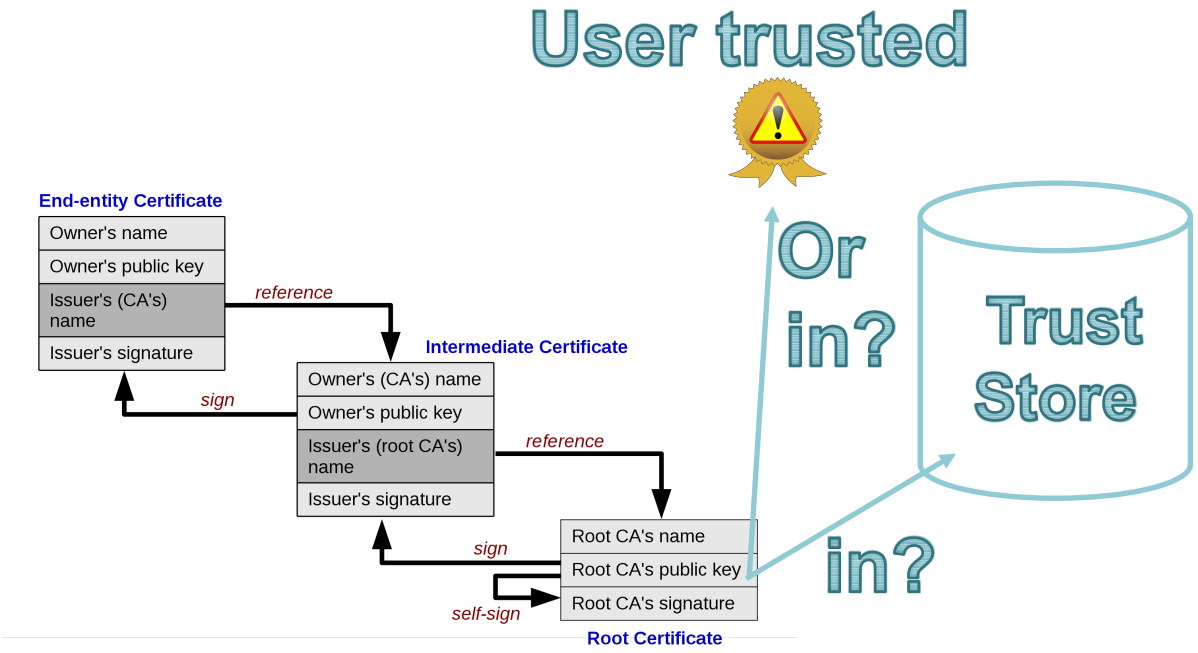
Android will alert the use
When a suspicious trust
has been added



GENIVI®

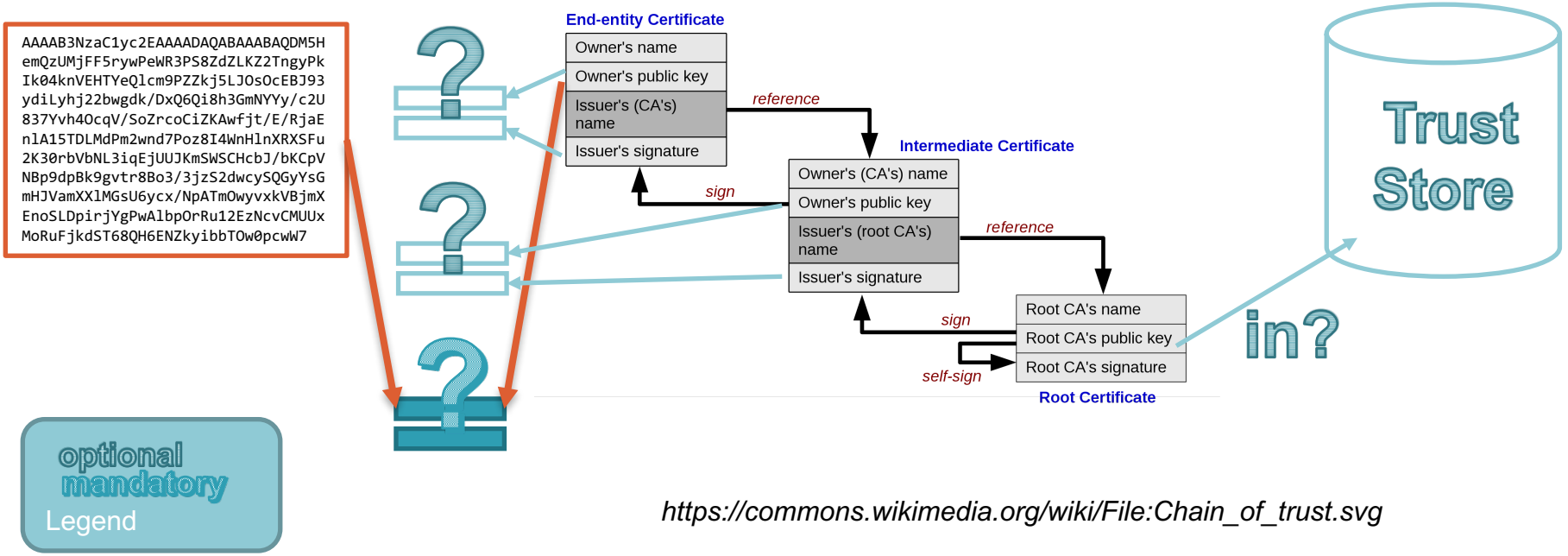
Recap: how the Trust-Store was ‘bypassed’

A User knowingly added the certificate to his trusted certs



Type 4: Trust the Public-Key of the endpoint.

- Verify server has an expected public key
  - variant: verify hash of public key



https://commons.wikimedia.org/wiki/File:Chain_of_trust.svg

GENIVI®

Type 4 Defeats

- Collision with pub key
- Patch Software
- Disable SSL entirely with JustTrustMe (github.com/Fuzion24/JustTrustMe)
- Patch certificate checks in Java using FRIDA (demo in Type 5)

Attack	Difficulty in-field	… at-home
Public key Collision	Infeasible	Infeasible
Patch Software	Infeasible	Moderate.
Disable SSL	Infeasible	Easy

GENIVI®

Type 4 Examples in Automotive Apps

- For all the applications surveyed, none.
  - This method is inflexible to changes in the server certificates.

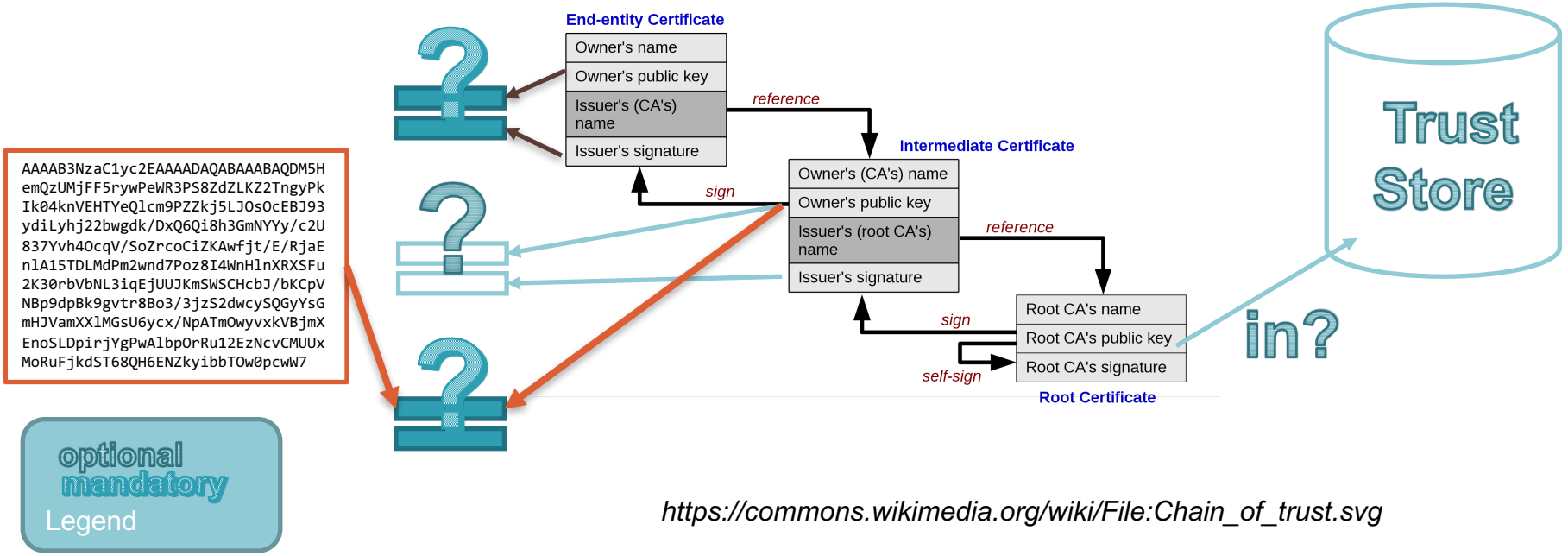
Discussion: More on Pinning

- Type 4 (and Type 5, coming up) are ”Certificate Pinning”
- This is the de-facto design to mitigate nearly all in-field attacks.
  - At-home attacks are still possible
- For many more details; consult owasp.org/index.php/Certificate_and_Public_Key_Pinning

GENIVI®

Type 5: Trust the Public-Key of a Signer of the Endpoint.

- Verify signer of the server's certificate has an expected public key
  - variant: verify hash of public key



https://commons.wikimedia.org/wiki/File:Chain_of_trust.svg

GENIVI®

Type 5 Defeats: Same as Type 4

- All the type 4 defeats apply here too.
- Demo:

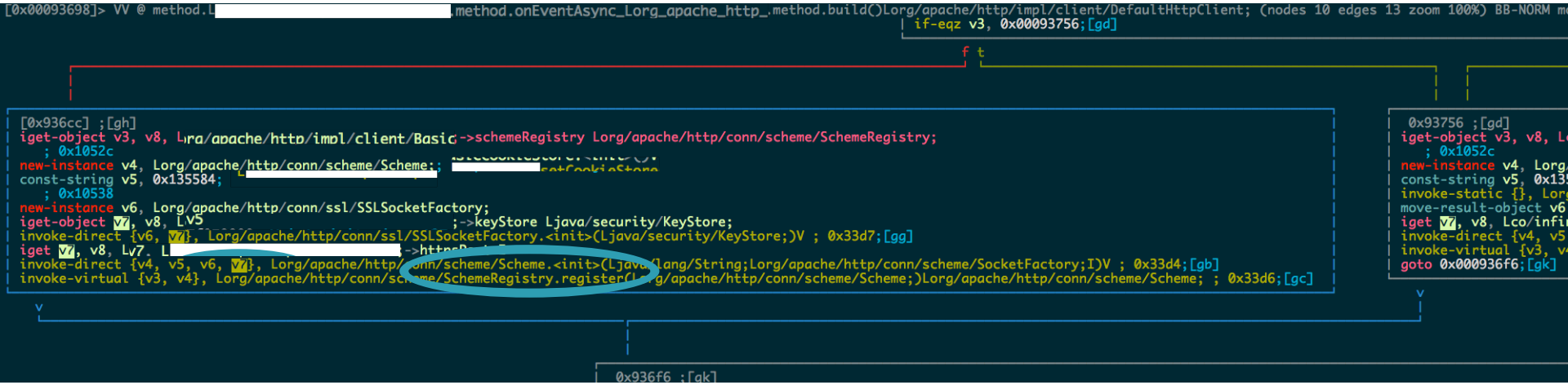
FRIDA

- Frida, a javascript hooking engine.
  - Installed through simple commands on rooted device
    - `adb push frida-server /data/local/tmp`
    - `adb shell “chmod 755 /data/local/tmp/frida-server”`
    - `Adb shell “su –c chown root:root /data/local/tmp/frida-server”`
    - `adb shell “/data/local/tmp/frida-server &”`
  - Connect from host
    - `frida –U -f <application> --codeshare pcipolloni/universal-android-ssl-pinning-bypass-with-Frida –no-pause`
    - Additional api available in documentation page: (www.frida.re/docs/home/)
- Capture data before encrypted

GENIVI®

Type 5 Examples in Automotive Apps

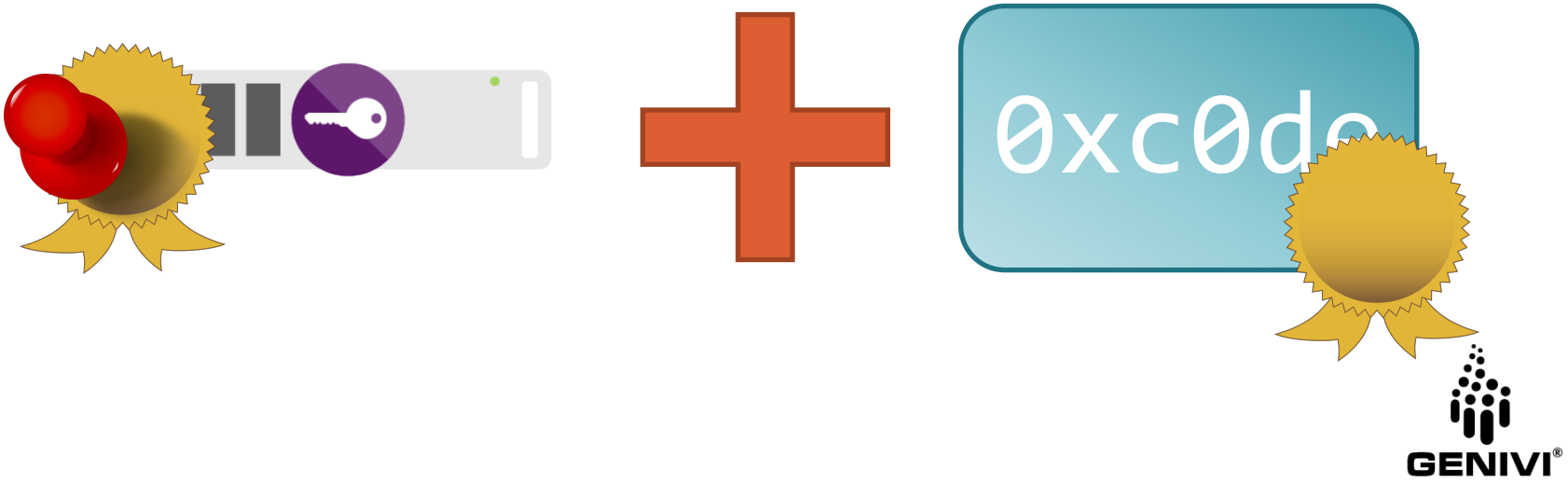
- For all the applications surveyed, one.
  - But we can't show code sample from that one (even anonymized ones)



GENIVI®

Type 6+: Integrity-Verification of Public Key Pinning Data

- Pin the certificate (or an intermediate certificate) AND verify that the application performing the check hasn't been tampered
  - Easy: verifying it hasn't been tampered on-disk (or NAND or whatever)
  - Harder: verifying it hasn't been tampered in-memory
  - Hardest: doing either of those in a way that an attacker at-home can't easily disable



GENIVI®

Type 6 Defeats

Bypass certificate pinning AND	Difficulty in-field	… at-home
Bypass simple on-disk IV	Infeasible	Easy (e.g. repackage APK) to Difficult (e.g. patch-out IV, use iOS jailbreak)
Bypass simple in-memory IV	"	Difficult (patch-out IV)
Bypass mutually-reinforcing protections around on-disk/in-memory IV	"	“Very” Difficult (reverse-engineer & patch-out all)
Bypass renewable mutually-…	"	Infeasible (attacker efforts restarted repeatedly)

GENIVI®

Type 6 Examples in Automotive Apps

- None (yet).
  - This is a common design for media players and mobile banking apps, but the automotive apps haven't reached this level of sophistication yet.

Review

- Type system progression
  - Protections
  - Defeats
  - Automotive Examples
- Tools
  - MitmProxy
  - FRIDA
- What Type level is correct?

Thank you!

Visit GENIVI at http://www.genivi.org or http://projects.genivi.org

Contact us: help@genivi.org

Appendix 1: Mitmproxy setup

- Detailed documents listed at http://docs.mitmproxy.org/en/latest/install.html
  - It might be worth using a specific device for the setup.
- Installation Options
  - Binary Download options: releases page
  - Package manager options:
    - Homebrew on OSX
      - `brew install mitmproxy`
    - Pacman on Arch Linux
      - `sudo pacman -S mitmproxy`
  - Source:
    - Python
      - `pip3 install mitmproxy`

GENIVI®

Appendix 1: Mitmproxy Setup Certs

- mitmproxy generates its own certificate into the ~/.mitmproxy/mitmproxy-ca.pem

- If a custom certificate is needed, this can be specified with the –client-certs option.
  - Requirements:
    - installation of new root certificates onto android device
    - Creation of self signed root CA through openssl

GENIVI®