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

<table>
<thead>
<tr>
<th>Differences</th>
<th>Time-based 2FA</th>
<th>Key-based 2FA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time factor</td>
<td>Compulsory</td>
<td>Optional</td>
</tr>
<tr>
<td>Impact</td>
<td>Affect both sides</td>
<td>Affect individual</td>
</tr>
<tr>
<td>Secret sharing</td>
<td>Sharing same secret (secret seed)</td>
<td>Never share secret (private key)</td>
</tr>
<tr>
<td>DB security</td>
<td>Required</td>
<td>Not required</td>
</tr>
</tbody>
</table>

- **Secret sharing**: For time-based 2FA, secrets are shared (secret seed), whereas for key-based 2FA, secrets are never shared (private key).
OTP

- SMS, Voice
- Software token
  - e.g., Google Authenticator, Authy, Duo Mobile, etc
- Hardware token
  - e.g., Yubikey, etc
### OTP: Time vs HMAC

<table>
<thead>
<tr>
<th>Differences</th>
<th>TOTP</th>
<th>HOTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validity</td>
<td>Short-lived</td>
<td>Unknown</td>
</tr>
<tr>
<td>Moving factor</td>
<td>Time-based</td>
<td>Counter-based</td>
</tr>
<tr>
<td>Time Synchronization</td>
<td>Required</td>
<td>Not required</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Light</td>
<td>Heavy</td>
</tr>
</tbody>
</table>
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 ??

\[
\text{Probability of Winning with } x \text{ matching nos.} = \frac{C(k, x) \times C(n - k, k - x)}{C(n, k)} = \frac{k! (n-k)!}{(k-x)! x!} \frac{n!}{k!}
\]
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

User

Time
Secret

Cryptographic Hash

Database

Secret

Time

Cryptographic Hash

Match?

Login

login request

secret shared at registration

unique code

unique code
Engineering the Arbitrary TOTP

Secret seed + 🕒 Time = ☊ 102113 Token

- Regardless: label, issuer, any other info
Stealing Secret Key

- Code leak
- Vulnerabilities
- Social engineering
- Malware infection
Codes Leak

- 3rd 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

![Image showing secret keys stored in localStorage](image_url)
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

\[
\text{<secret seed>} + \text{<fixed timestamp>} = \text{<000000 – 999999>}
\]

<table>
<thead>
<tr>
<th>Secret Key</th>
<th>Local Unix Timestamp</th>
<th>TOTP</th>
<th>Methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td>nps3eos2yhmtd5fl2qb7oaosmivg68r4</td>
<td>1504981218</td>
<td>394244</td>
<td></td>
</tr>
<tr>
<td>nps3eos2yhmtd5fl2qb7oaosmivg68r4</td>
<td>1504981321</td>
<td>246524</td>
<td></td>
</tr>
<tr>
<td>nps3eos2yhmtd5fl2qb7oaosmivg68r4</td>
<td>1504981350</td>
<td>937375</td>
<td>Attacker exploits the ntpd daemon to forcefully set the internal clock to 1504981350, in order to get TOTP (937375)</td>
</tr>
<tr>
<td>nps3eos2yhmtd5fl2qb7oaosmivg68r4</td>
<td>1504981421</td>
<td>383285</td>
<td></td>
</tr>
</tbody>
</table>
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)
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" *`
Capturing OTP

- Social engineering attacks
- Eavesdrop on air, LAN, WAN
(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
<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Message</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>18646713318</td>
<td>12067925640</td>
<td>Данный номер уже используется</td>
<td>3 minutes ago</td>
</tr>
<tr>
<td>14088319233</td>
<td>12067925640</td>
<td>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&amp;C apply. <a href="http://leo.tips/unsubscribe">http://leo.tips/unsubscribe</a></td>
<td>13 minutes ago</td>
</tr>
<tr>
<td>16146954730</td>
<td>12067925640</td>
<td>474317 is your Google verification code.</td>
<td>26 minutes ago</td>
</tr>
<tr>
<td>19106844086</td>
<td>12067925640</td>
<td>Doh, you already have a profile attached to this number and we currently only allow one profile per mobile number.</td>
<td>49 minutes ago</td>
</tr>
<tr>
<td>1646439673</td>
<td>12067925640</td>
<td>Hello</td>
<td>1 hour ago</td>
</tr>
<tr>
<td>13233205506</td>
<td>12067925640</td>
<td>【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。</td>
<td>2 hours ago</td>
</tr>
<tr>
<td>12092664883</td>
<td>12067925640</td>
<td>【知乎】创建帐号的验证码是 683862，10 分钟内有效。</td>
<td>2 hours ago</td>
</tr>
<tr>
<td>818085172736</td>
<td>12067925640</td>
<td>你好</td>
<td>2 hours ago</td>
</tr>
<tr>
<td>14159428815</td>
<td>12067925640</td>
<td>Thumbtack reminder: Ralph K. still wants quotes thmtk.com/kilHTSQY. 2 quotes have been submitted.</td>
<td>3 hours ago</td>
</tr>
<tr>
<td>13236382150</td>
<td>12067925640</td>
<td>Your Shaadi.com Phone Verification PIN is 5621</td>
<td>3 hours ago</td>
</tr>
<tr>
<td>13025178583</td>
<td>12067925640</td>
<td>【腾讯云】 134358 (Tencent Cloud Verification Code)</td>
<td>4 hours ago</td>
</tr>
<tr>
<td>13013583168</td>
<td>12067925640</td>
<td>Doh, you already have a profile attached to this number and we currently only allow one profile per mobile number.</td>
<td>4 hours ago</td>
</tr>
<tr>
<td>12037385811</td>
<td>12067925640</td>
<td>709879 is your BIGO LIVE verification code.</td>
<td>5 hours ago</td>
</tr>
<tr>
<td>12064561499</td>
<td>12067925640</td>
<td>[Alibaba Group] Your verification code is 644971. Please do not disclose your code to anyone, including Alibaba staff!</td>
<td>5 hours ago</td>
</tr>
<tr>
<td>12823331149</td>
<td>12067925640</td>
<td>【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。</td>
<td>5 hours ago</td>
</tr>
</tbody>
</table>
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

PublishAuthentication yes

AuthorizedKeyFile .ssh/authorized_keys

PasswordAuthentication no

ChallengeResponseAuthentication no

```
root@metasploitable:/home/msfadmin/.ssh# cat authorized_keys
ssh-dss AAAAB3NzaC1kc3MAAACABWgbcHyvxvF2YRX0g7iyozaZazzHuiU5+63hKFOhntep6i2zAQ5oYc1kCyj8s8WHhcPB85uRPiXYL91rAAAAPQDL+pkRl6yv99HyXwyZWMxSAGyTvTkue4nvGCFxnDr58xalpZcSO65R5j5CSARMHU6WBIWd3Myz3JNqTn4uRaLrZ5/Y4pCra01bxtRSJah0RjK5xwAUPZ282N0fzJcyVlBojMyP1bAp1pS1eCcU/LG

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDQDQbAABAwg91K003r3v7eY3M65qq5aQzD9dSV9euffWmZm

pub@root@metasploitable:/home/msfadmin/.ssh# ls
authorized_keys config
r

root@metasploitable:/home/msfadmin/.ssh# cat 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
+ 
  |
  v

Building Target Database

+---------------------------------
| 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 Private Key Exposure
+---------------------------------

| POST Modules
v

Keys Extraction
+ 
  |
  v

Looted Keys

| v

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

Certificate Details

Certificate Information

Windows does not have enough information to verify this certificate.

Issued to: 2FA Assassin

Issued by: 2FA Assassin CARoot

Valid from 1/1/2014 to 1/1/2040

You have a private key that corresponds to this certificate.

Install Certificate... Issuer Statement

Learn more about certificates

OK
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
Finding SSL Server

1. Search for the actual URL

2. Authentication (with looted client certificate)
Setting up client system

```
./assassin.py --cert windows --user <username> --secret <password> --host <client_machine_ip>
```
Arbitrary SSH Key

```
sshkey
 +---------------------->
 |                       |
 | #1                    |
 | +---------------------->
 | | '2fassassin'         |
 | | +-------------------+ |
 | |                   | #1 {add keys}          |
 | |                   | +-------------------+ |
 | | create user        | +-------------------+ |
 | | +-------------------+ |
 | |                   | +-------------------+ |
 | | #2                | account_1            |
 | | +-------------------+ |
 | | generate RSA keypair| account_2           |
 | | +-------------------+ |
 | |                   | 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
Impersonating SSL Website

- Using looted server certificate
- Must get server key
- Client side attacks
Server certificate was stolen by attacker

Attacker (2FAssassin)

-\^-

reverse shell connects back to attacker

Windows Client (172.16.173.180)\n
\n
(normal) client auth

SSL Website (172.16.173.182)

-\^-

SSL website is now at 172.16.173.194

DNS Spoofing

DNS Server (172.16.173.191)

client download malware from the phishing website

Phishing Website (172.16.173.194)

Attacker cracked the server certificate, then use it 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

Example

`__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
kali

user1:password
(user1_private_key)

msfuser1

authorized_keys: user1_public_key

./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