INSOMNIA

SECURITY SPECIALISTS :: REST SECURED

# Modern Corporate Wifi Rustling

# Who Am I

Chris Smith (@chrismsnz)

Previously:

- Network admin
- Polyglot Developer - Python, PHP, Go + more
- Linux Sysadmin

Currently:

- Pentester, Snr Consultant at Insomnia Security
- Little bit of research

# What is Corporate Wireless

- Corporate == large organisations

- Usually multiple networks, with differing levels of security and sensitivity

- Phones, laptops, tablets, hotdesks, BYOD, as well as better infrastructure

# Chris's Handwavey Guide to Corporate Wireless

# LAN/Corporate Wireless

## Network Access

- SENSITIVE
- Generally full access to internal corporate network
- AD, fileservers, business apps, other workstations

# LAN/Corporate Wireless

## Authentication Method and Credentials

- **SENSITIVE**
- WPA2 Enterprise
- Usually TLS certificates or User/Machine Domain credentials

# LAN/Corporate Wireless

## Common Issues

- "Single Factor" network authentication
- Poor client configuration
- Poor authentication lifecycle management

# Guest Wireless

## Network Access

- WHO CARES
- Internet Only

# Guest Wireless

## Authentication Method and Credentials

- **WHO CARES**
- Open Network, Captive Portal
- Time limited, on-demand, unique credentials

# Guest Wireless

## Common Issues

- Shared infrastructure/Bad segregation
- Application-level security
- Preauth Access (DNS, ICMP etc...)

# BYOD Wireless

## Network Access

- **WHO CARES**
- Internet Only
- Possibly some access to secured internal services (e.g. OWA, Citrix etc...)

# BYOD Wireless

## Authentication Method and Credentials

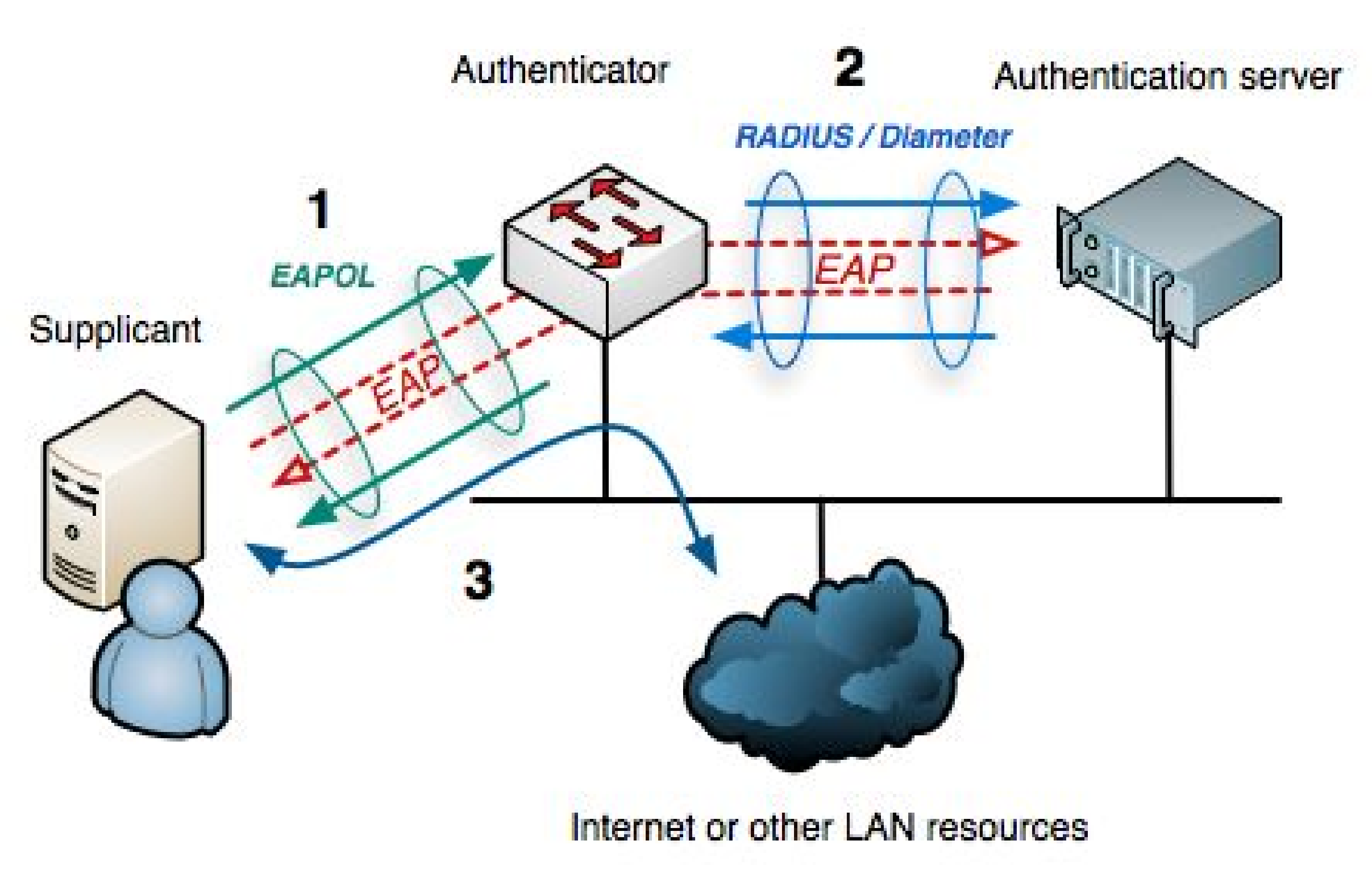
- SENSITIVE
- WPA2 Enterprise, EAP-PEAP/MSCHAPv2
- Corporate Domain User Credentials

# LIVE FIRE EXERCISE

# But... How?

- Corporate requires employees to authenticate to BYOD wireless network
- But does not/can not require that their device is configured securely
- Probably EAP-PEAP/EAP-MSCHAPv2 with domain credentials

# 802.1X & EAP

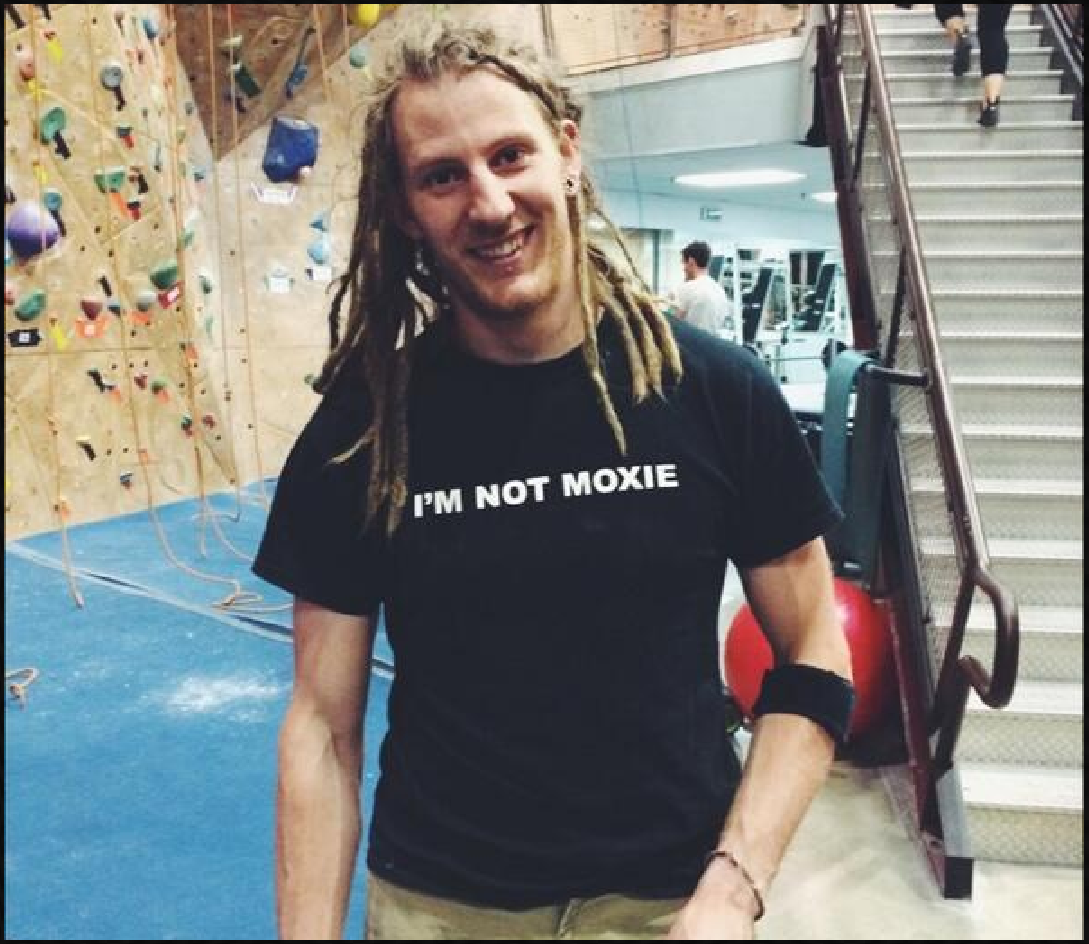


"802.1X wired protocols" by Arran Cudbard-Bell Arr2036 - Own work.

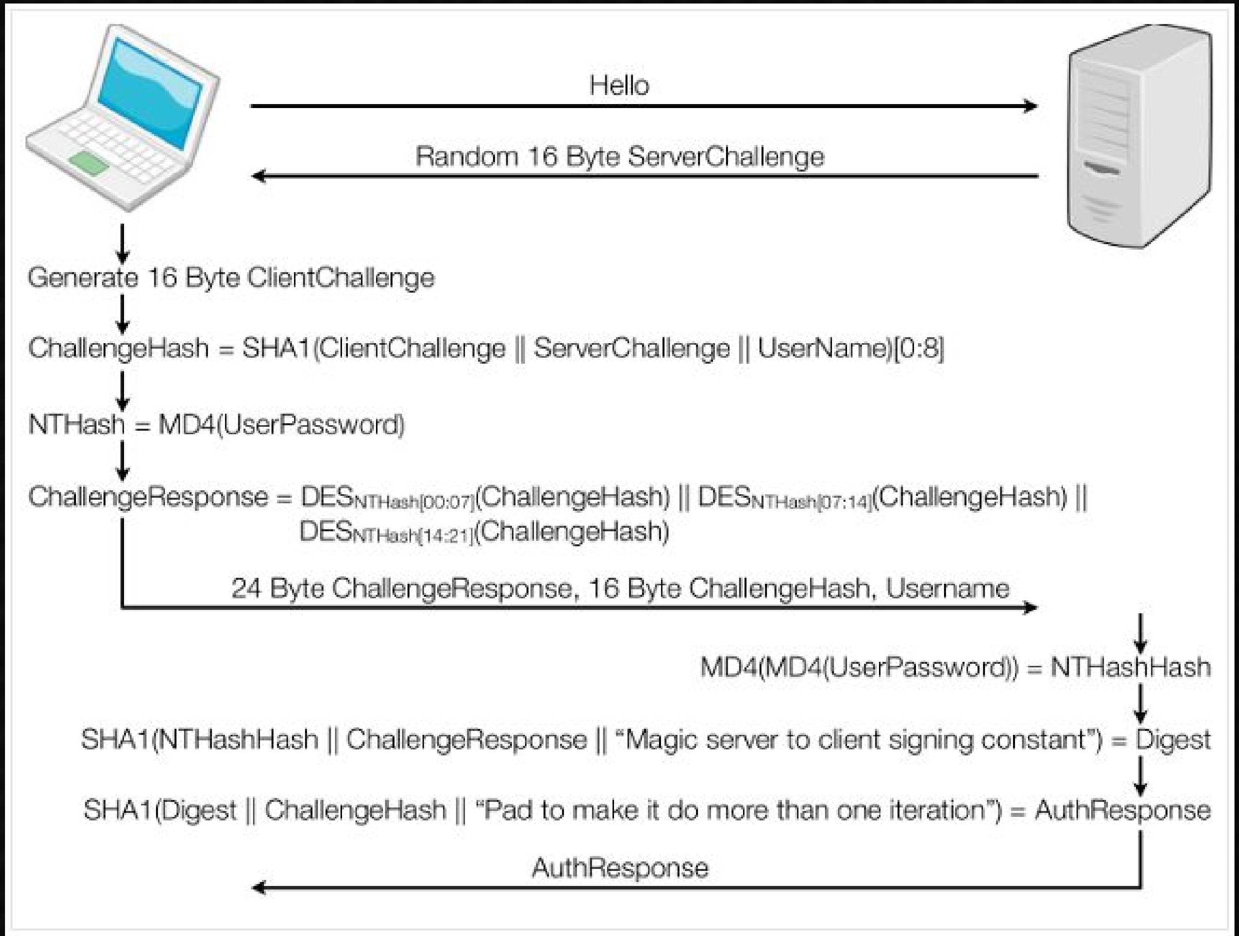
# EAP-MSCHAPv2

- Microsoft extended CHAP for various Windows integration reasons, e.g.
  - Supports domain-based password changes, expirations etc...
  - Use of MS primitives such as NTLMv2
- Both ends need knowledge of secret to properly authenticate

# Moxie Marlinspike



# MSCHAPv2



Hello

Random 16 Byte ServerChallenge

Generate 16 Byte ClientChallenge

ChallengeHash = SHA1(ClientChallenge || ServerChallenge || UserName)[0:8]

NTHash = MD4(UserPassword)

ChallengeResponse = $DES_{NTHash[00:07]}$(ChallengeHash) || $DES_{NTHash[07:14]}$(ChallengeHash) || $DES_{NTHash[14:21]}$(ChallengeHash)

24 Byte ChallengeResponse, 16 Byte ChallengeHash, Username

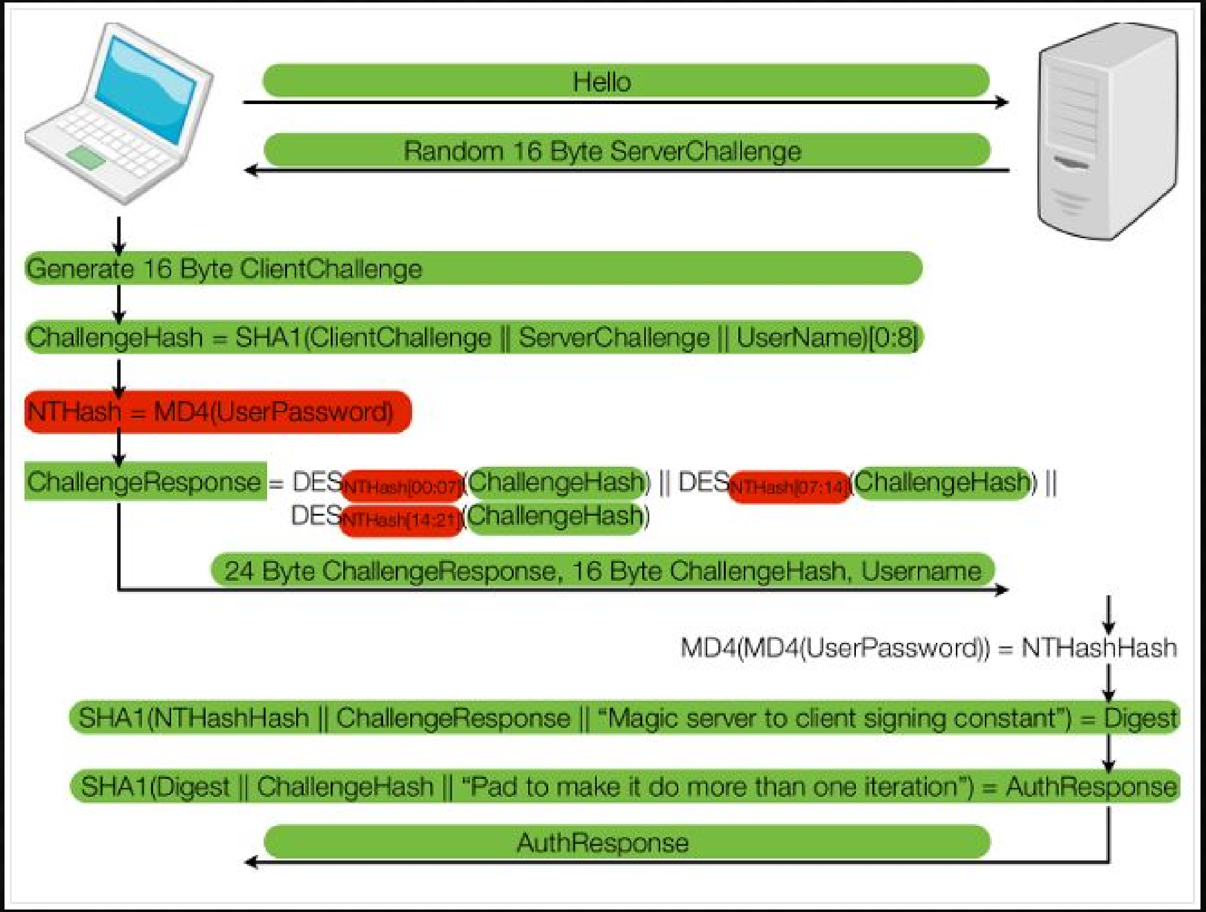
MD4(MD4(UserPassword)) = NTHashHash

SHA1(NTHashHash || ChallengeResponse || “Magic server to client signing constant”) = Digest

SHA1(Digest || ChallengeHash || “Pad to make it do more than one iteration”) = AuthResponse

AuthResponse

# MSCHAPv2 post Moxie



Hello

Random 16 Byte ServerChallenge

Generate 16 Byte ClientChallenge

ChallengeHash = SHA1(ClientChallenge || ServerChallenge || UserName)[0:8]

NTHash = MD4(UserPassword)

ChallengeResponse = $\text{DES}_{\text{NTHash[00:07]}}$(ChallengeHash) || $\text{DES}_{\text{NTHash[07:14]}}$(ChallengeHash) || $\text{DES}_{\text{NTHash[14:21]}}$(ChallengeHash)

24 Byte ChallengeResponse, 16 Byte ChallengeHash, Username

MD4(MD4(UserPassword)) = NTHashHash

SHA1(NTHashHash || ChallengeResponse || "Magic server to client signing constant") = Digest

SHA1(Digest || ChallengeHash || "Pad to make it do more than one iteration") = AuthResponse

AuthResponse

# EAP-MSCHAPv2 - Attacking

- Capture the NetNTLM encrypted challenge, feed to this:

# EAP-MSCHAPv2 - Attacking

- PCAP entire MSCHAPv2 handshake
- Give to Cloudcracker + USD + 24 hrs
- 100% recovery of NTHash (used as DES key)
- Can then crack it, or pass the hash on to the authenticated network, or other domain authenticated services

# EAP-MSCHAPv2 - Recap

- Pretty broken protocol
- Requires both ends to know password to complete handshake
- If attacker can observe the handshake, password or raw hash can be recovered
- Be sure to ask Moxie Mallardspike about Cloudquacker if you see him round at the con

# EAP-PEAP

- You got your TLS in my layer 2!
- Pretty much exactly the same thing as e.g. HTTPS connections, except has another EAP transaction inside the tunnel
- Successfully prevents eavesdropping of MSCHAPv2 handshakes as they float through the air
- But, confidentiality requires more than just encryption

# EAP-PEAP

- Full TLS negotation including Certificate
- Encryption methods are negotiated
- Tunnel is between supplicant (client) and Authentication Server - not authenticator!
- How well does this apply to information available during layer 2 authentication?

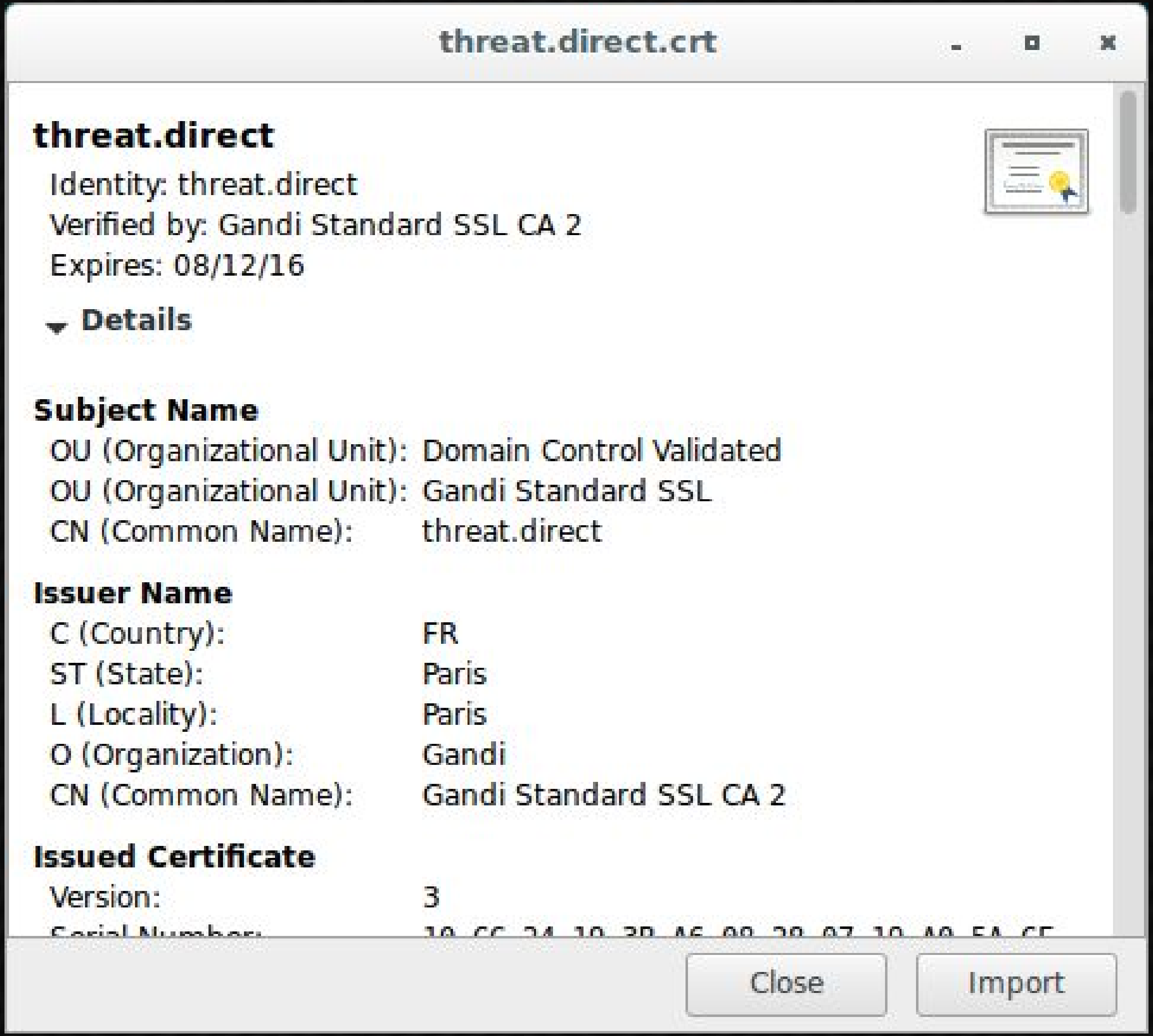
# EAP-PEAP - Trust

- HTTPS has the 3rd party CA system to bootstrap trust
- Browser can verify trust in a certificate by using its CA trust root
- EAP-PEAP can verify trust by ???
- Does the 3rd party CA system make sense in this context?

# EAP-PEAP - Identity

- Certificates certify a subject, AKA Common Name (CN)
- In HTTPS, this is generally the domain part of your URL
- With EAP-PEAP, it's ???
- What information does a client/supplicant have that can identify this authentication server?

# So trustworthy



**threat.direct**

Identity: threat.direct
Verified by: Gandi Standard SSL CA 2
Expires: 08/12/16

**Subject Name**

| | |
|---|---|
| OU (Organizational Unit): | Domain Control Validated |
| OU (Organizational Unit): | Gandi Standard SSL |
| CN (Common Name): | threat.direct |

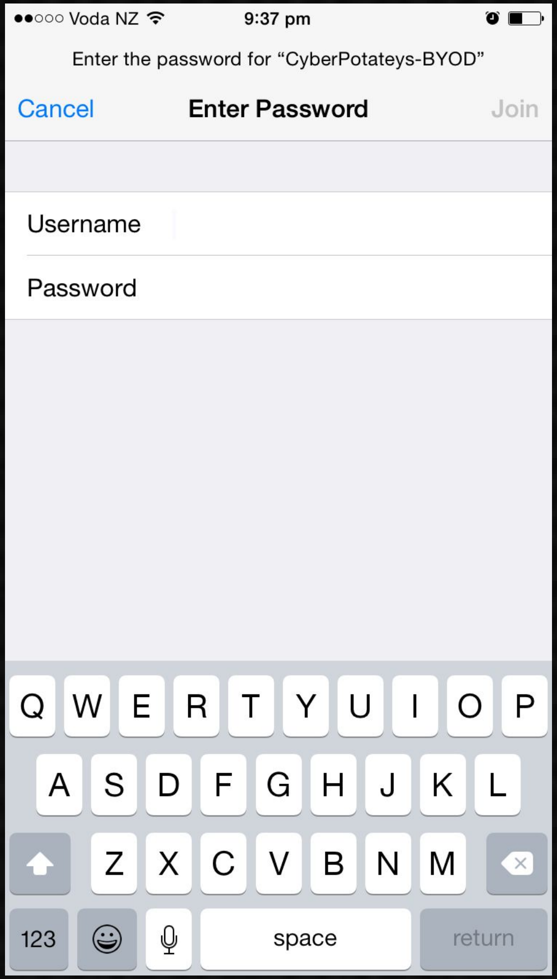
**Issuer Name**

| | |
|---|---|
| C (Country): | FR |
| ST (State): | Paris |
| L (Locality): | Paris |
| O (Organization): | Gandi |
| CN (Common Name): | Gandi Standard SSL CA 2 |

**Issued Certificate**

| | |
|---|---|
| Version: | 3 |
| Serial Number: | 10 CC 24 10 3B A6 00 28 07 10 A0 5A CF |

# EAP-PEAP - Configuration

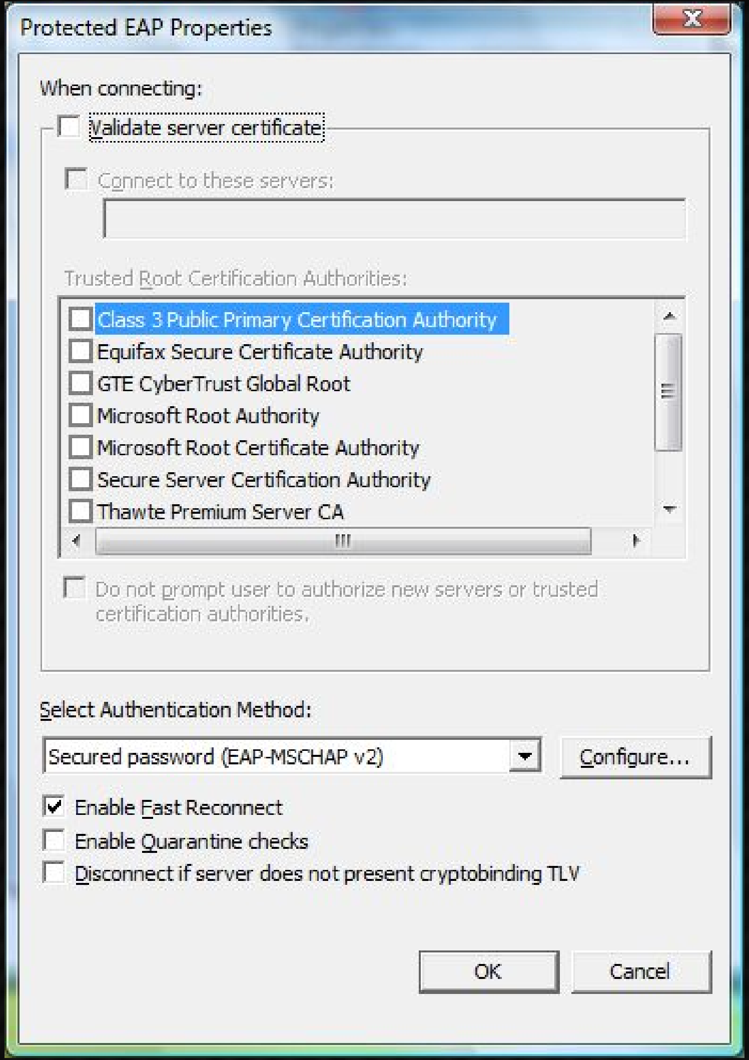


Enter the password for “CyberPotateys-BYOD”

Cancel | **Enter Password** | Join

Username

Password



Protected EAP Properties

When connecting:

- □ Validate server certificate
- □ Connect to these servers:

Trusted Root Certification Authorities:

- □ Class 3 Public Primary Certification Authority
- □ Equifax Secure Certificate Authority
- □ GTE CyberTrust Global Root
- □ Microsoft Root Authority
- □ Microsoft Root Certificate Authority
- □ Secure Server Certification Authority
- □ Thawte Premium Server CA

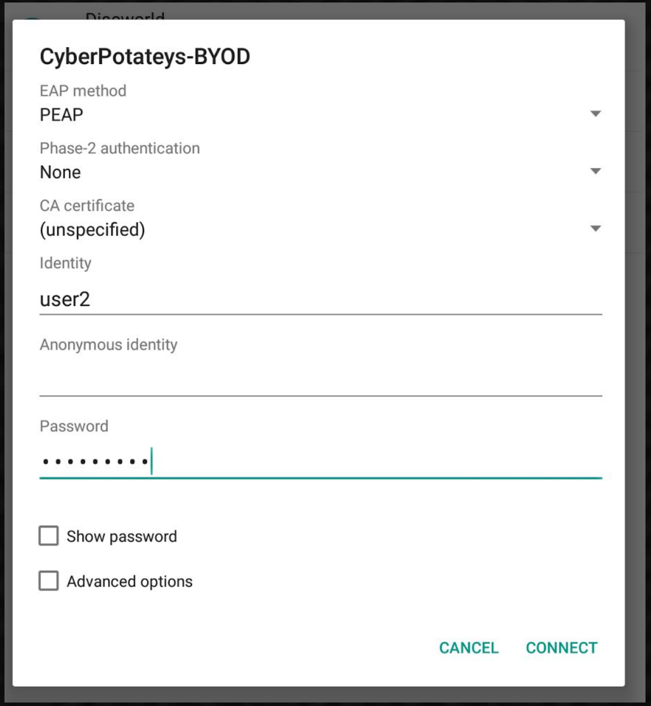
□ Do not prompt user to authorize new servers or trusted certification authorities.

Select Authentication Method:

Secured password (EAP-MSCHAP v2) | Configure...

- ☑ Enable Fast Reconnect
- □ Enable Quarantine checks
- □ Disconnect if server does not present cryptobinding TLV

OK | Cancel



**CyberPotateys-BYOD**

EAP method: PEAP

Phase-2 authentication: None

CA certificate: (unspecified)

Identity: user2

Anonymous identity:

Password: ••••••••••

- □ Show password
- □ Advanced options

CANCEL | CONNECT

# Officer Unfriendly & PERVERT COWBOY

# Post-attack - Corp

- Try using the captured credentials/hashes to authenticate directly to the network

- Look for any "Special" devices

- User auth may be allowed for uncommon/unmanaged devices, or for special users.

# Post-attack - BYOD

- Semi-trusted network
- Check Shared Infrastructure
- Check Network Segregation
- Other services available on BYOD network - OWA, Citrix, etc...

# Recap

- Cannot ensure safe EAP-PEAP/MSCHAPv2 client configuration without management
- Attackers want BYOD authentication info, generally not BYOD network access
- Wireless IDS/IPS won't save you, will find you at carpark, coffee shop, airport or Kiwicon

# Recap

- Manage your wireless clients
- Use the strongest authentication you can stomach
- Don't neglect physical and network-level security, protect your infrastructure

INSOMNIA

SECURITY SPECIALISTS :: REST SECURED

www.insomniasec.com

Chris Smith - @chrismsnz