PEAP: Pwned Extensible Authentication Protocol

ShmooCon 2008

Joshua Wright, jwright@willhackforsushi.com
Brad Antoniewicz, Brad.Antoniewicz@foundstone.com

Agenda

- Introductions
- Evolution of wireless
- IEEE 802.1X and EAP overview
- EAP attack surface
- Attacking EAP types
- Conclusion

Introductions

Works for Foundstone
Hacks stuff for a living
Can hold his liquor

Hacks for Sushi
Has mercury poisoning
Drunk on O'Douls

WLAN Security Evolution

- WEP has been dead since 2001
  - Thomas d'Otreppe et al at Aircrack-ng continue to do great work here
- LEAP deployments considerably fewer today than 2003
- WPA/WPA2 specify strong encryption, strong authentication mechanisms
- Commonly available EAP types provide reasonable security for most organizations

IEEE 802.1X in One Slide

- Network access authentication at layer 2
  - EAP provides authentication, WEP/TKIP/CCMP provides encryption support
- Supplicant, PAE (Authenticator), Authentication Server
- Supplicant and authentication server use an EAP type to authenticate, negotiate keys
  - PAE is agnostic to EAP type (except LEAP)
- Supplicant communicates via EAPOL, forwarded by PAE to auth. server in RADIUS TLV attribute

Not all EAP types are created equal

RFC4017 - EAP Requirements

- Specifies requirements for EAP methods
- All standard EAP methods must provide:
  - Mutual authentication
  - Resistance to dictionary attacks
  - Protection against MitM attacks
  - Protected ciphersuite negotiation
- EAP methods that fail these requirements
  - EAP-MD5, EAP-OTP, EAP-GTC, LEAP
- EAP methods that pass these requirements
  - PEAP, TTLS, EAP/TLS, EAP-FAST

EAP Attack Surface

How does EAP on wireless AP's expose your organization?

EAP Exposure

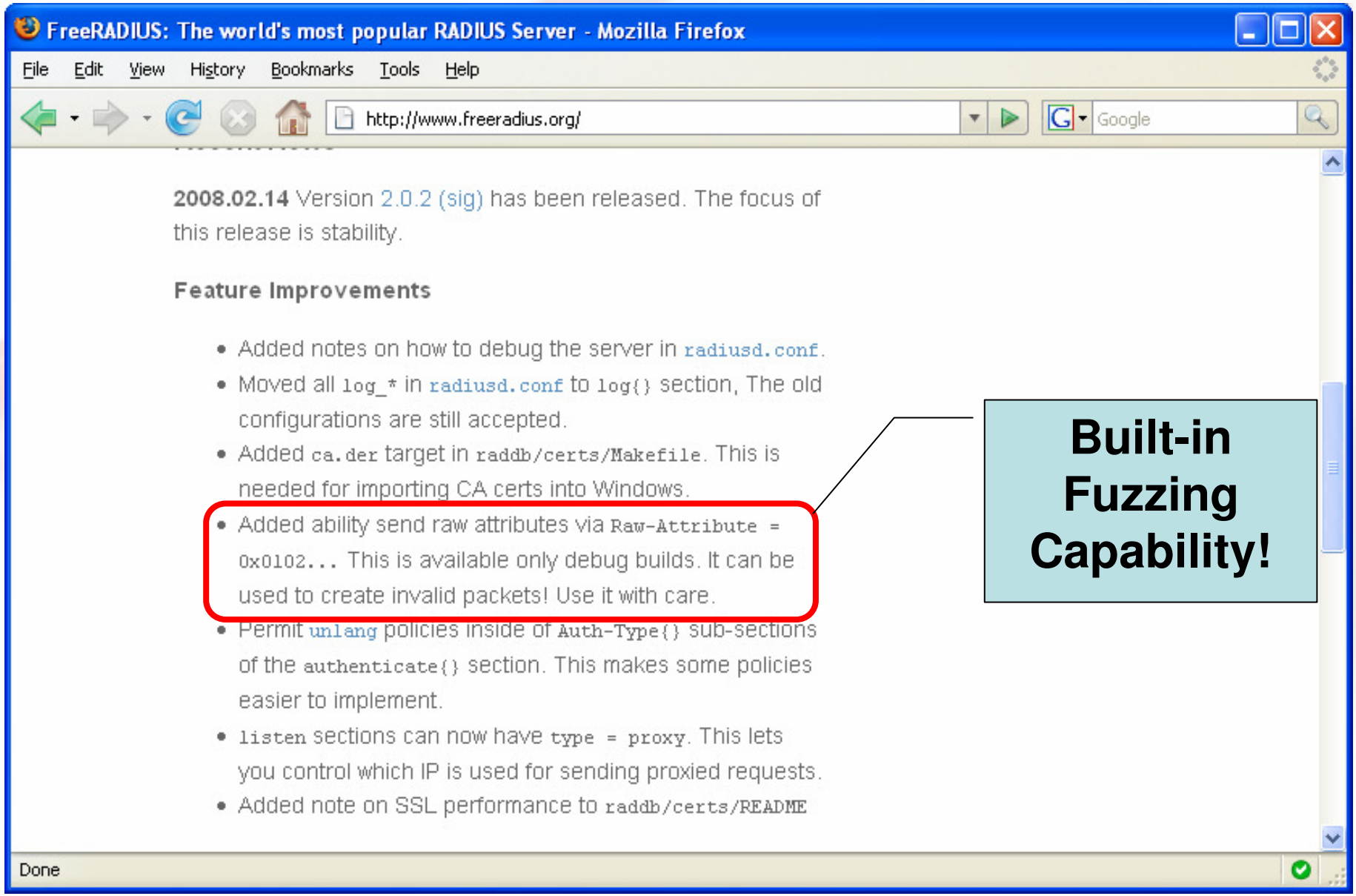
- Any unauthenticated user can initiate an EAP conversation
  - EAP can be complex to parse with support for fragmentation, retries, complex data structs
  - Cisco AP crash by Laurent Butti, Benoît Stopin, malformed EAP Identity Request
- EAP communicates with RADIUS server from any unauthenticated user
  - More complexity in EAP frame parsing
  - Pwn the RADIUS server, Pwn the World!

Client and Server Choices

- Many supplicant choices available
  - Native supplicants in Windows/WZC and OSX
  - Commercial supplicants from Funk/Juniper and MeetingHouse/Cisco
  - Free supplicants including wpa_supplicant, SecureW2, Open1X
- Several RADIUS choices available
  - Windows IAS, Cisco ACS, Juniper SBR, FreeRADIUS

Represents lots of unexplored code paths

New FreeRADIUS Release!



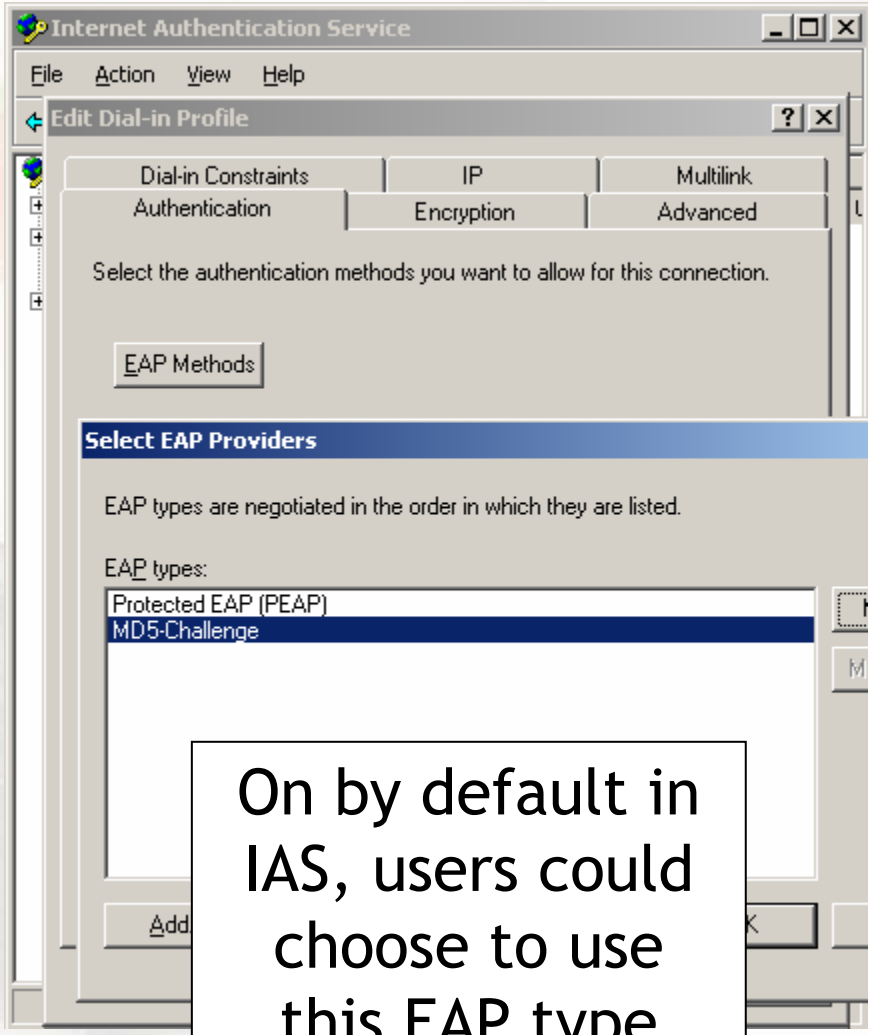
Built-in Fuzzing Capability!

Attacking EAP Types

A look at EAP-MD5, LEAP, EAP-FAST, PEAP and TTLS

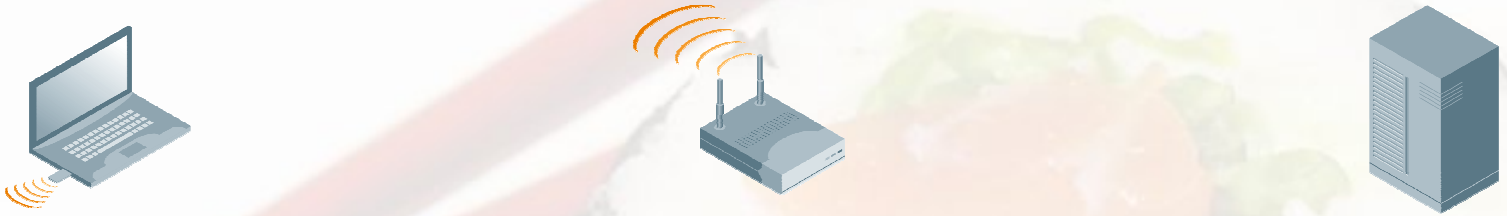
EAP-MD5

- Early, basic authentication mechanism
- Not RFC4017 compliant
- No support for encryption key delivery
- No native supplicant support in Windows
- Available native in OSX or Odyssey
- Server support in IAS, ACS, SBR, FreeRADIUS



On by default in IAS, users could choose to use this EAP type over PEAP

EAP-MD5 Exchange



802.11 authentication/association

EAP Identity Request

EAP Identity Response

RADIUS Identity

[RADIUS server generates a 16-byte MD5-Challenge, sends as EAP-Request]

EAP Request

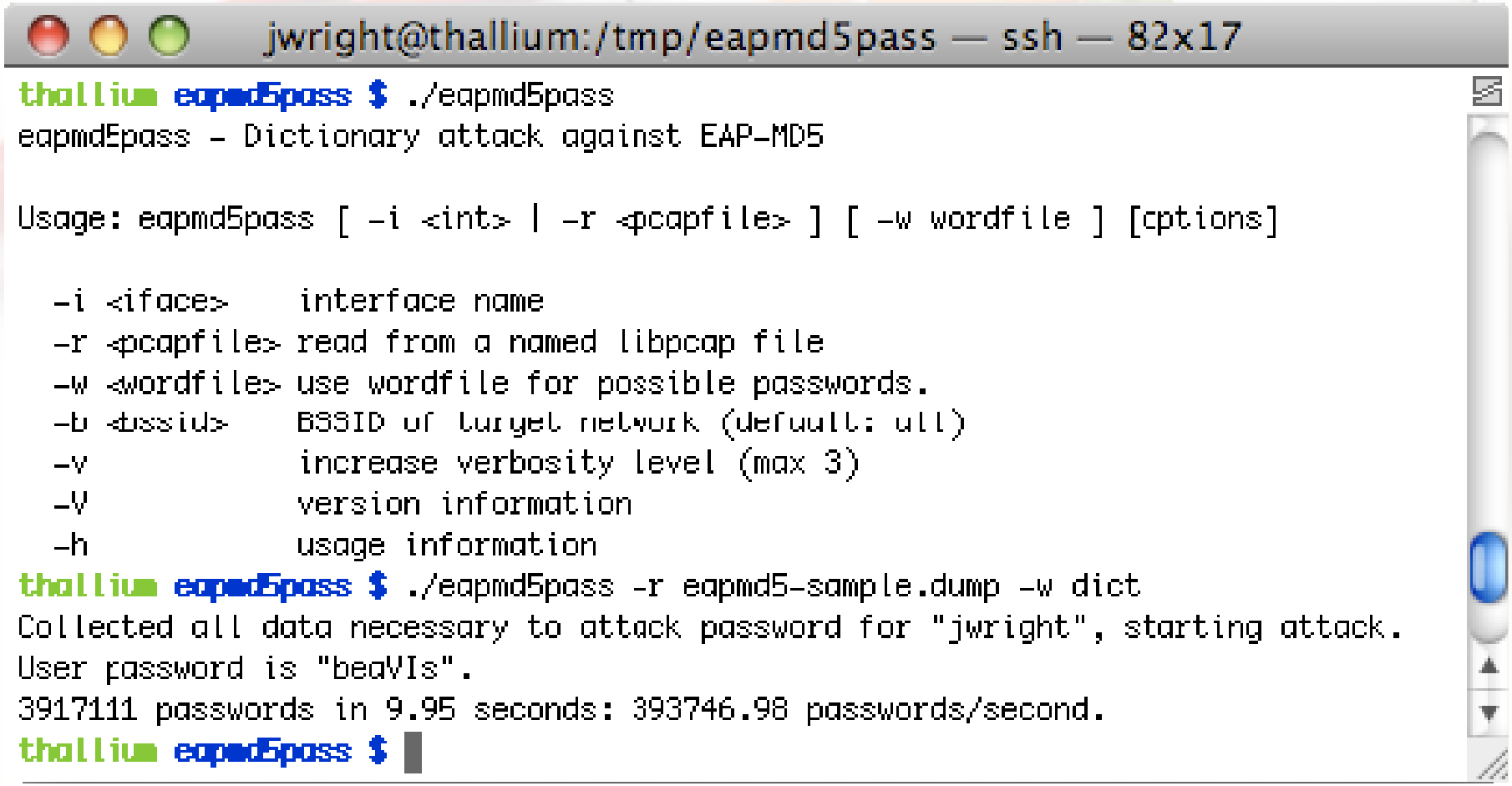
[Client calculates MD5(response ID | password | challenge), returns to RADIUS]

EAP Response

EAP Success/Failure

eapmd5pass

- Simple password auditing tool, GPL
- Read from libpcap file or monitor-mode interface



LEAP

- Security through obscurity with a proprietary protocol
- Uses MS-CHAPv1 challenge-response authentication mechanism
  - 8-byte challenge, 24-byte response
  - Response calculated using 3-DES keys from 16-byte password NTLM/MD4 hash
  - Third DES key is weak, accelerating dictionary attack
- Only available on Cisco AP's, not a compliant EAP type

Asleap

- Offline dictionary attack against LEAP
- Also applies to PPTP, and any MS-CHAPv1 or MS-CHAPv2 challenge/response mechanism
  - Specify challenge and response as command-line parameters
  - Thanks to Jay Beale for this suggestion
- 4 TB limit on precomputed hash lookup files

```
thallium asleap $ ./asleap -C ce:b6:98:85:c6:56:59:0c -R 72:79:f6:5a:a4:
:58:22:c8:9d:cb:dd:73:c1:b8:9d:37:78:44:ca:ea:d4 -f dict.dat -n dict.idx
asleap 2.1 - actively recover LEAP/PPTP passwords. <jwright@hasborg.com>
        hash bytes:         586c
        NT hash:            8846f7eaee8fb117ad06bdd830b7586c
        password:           jaybealehasaposse
thallium asleap $
```

EAP-FAST

- Cisco-developed EAP type following LEAP
  - Designed to be simple but secure
- Leverages Preshared Authentication Credentials (PAC)
  - Effectively a file-based authentication credential
- Challenge is in PAC provisioning
  - Manual option; sneaker-net copy PAC's
  - Automated option; anonymous DH
  - Automated option with validation; RSA

EAP-FAST PAC Provisioning

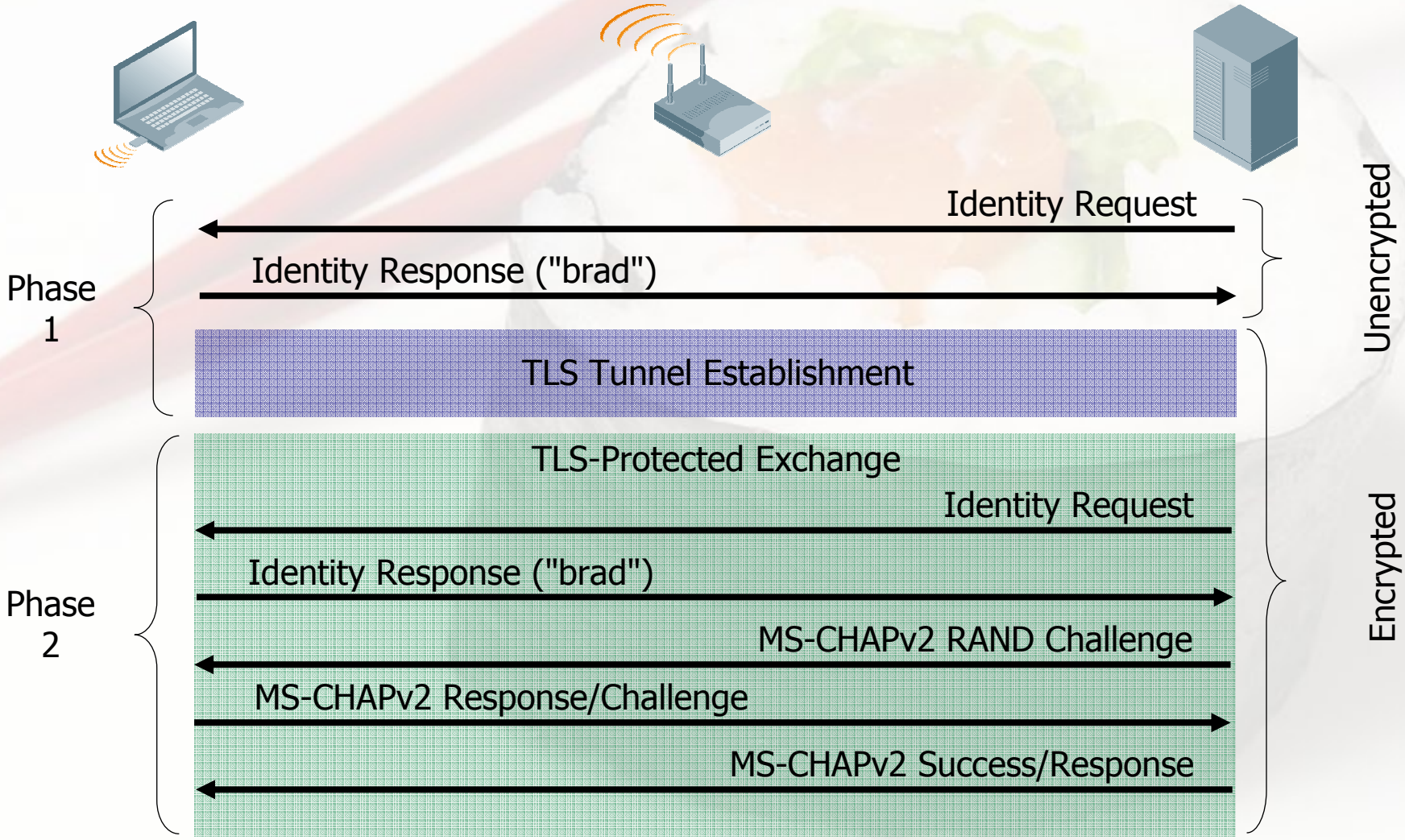
- PAC provisioning is secure, or simple, but not both
- Anonymous DH susceptible to AP impersonation
  - User discloses credentials using inner EAP method (e.g. EAP-MSCHAPv2)
  - Clearly identified in EAP-FAST docs cisco.com
- Fix is to provision a trusted certificate on clients and RADIUS to secure PAC exchange
  - Not simple, requires touching all workstations

Many users leave anonymous provisioning enabled, AP impersonation reveals weak credential exchange for new clients

PEAP and TTLS - Background

- Drafts introduced 2001/2002 leveraging tunneled authentication
  - Inner tunnel leveraging legacy authentication
  - Outer tunnel using TLS, protects inner tunnel
- Satisfies RFC4017 for mutual authentication, MitM attack mitigation, symmetric key derivation
- Requires certificate on RADIUS for STA to validate server identity
- TTLS differs primarily with support for any inner authentication protocol; PEAP=MS-CHAPv2

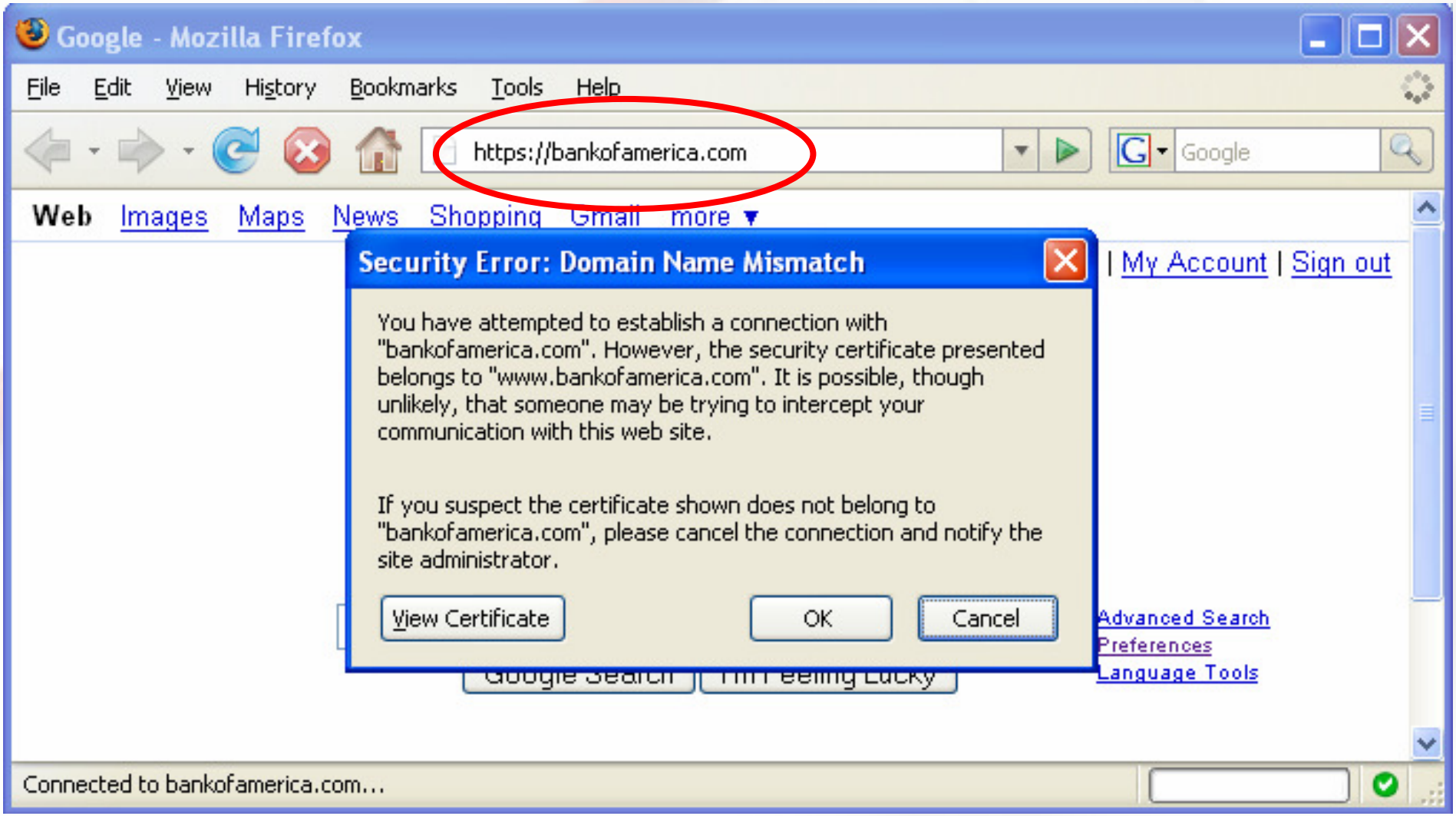
PEAP Transaction



Server Validation

- TLS provides authenticator validation
- Supplicant retrieves certificate from authenticator
  - Identifies signing authority
  - Validates as trusted CA
  - Compares CN of certificate to trusted RADIUS hostname
- Authentication server authenticates supplicant with inner authentication method

HTTP TLS Validation



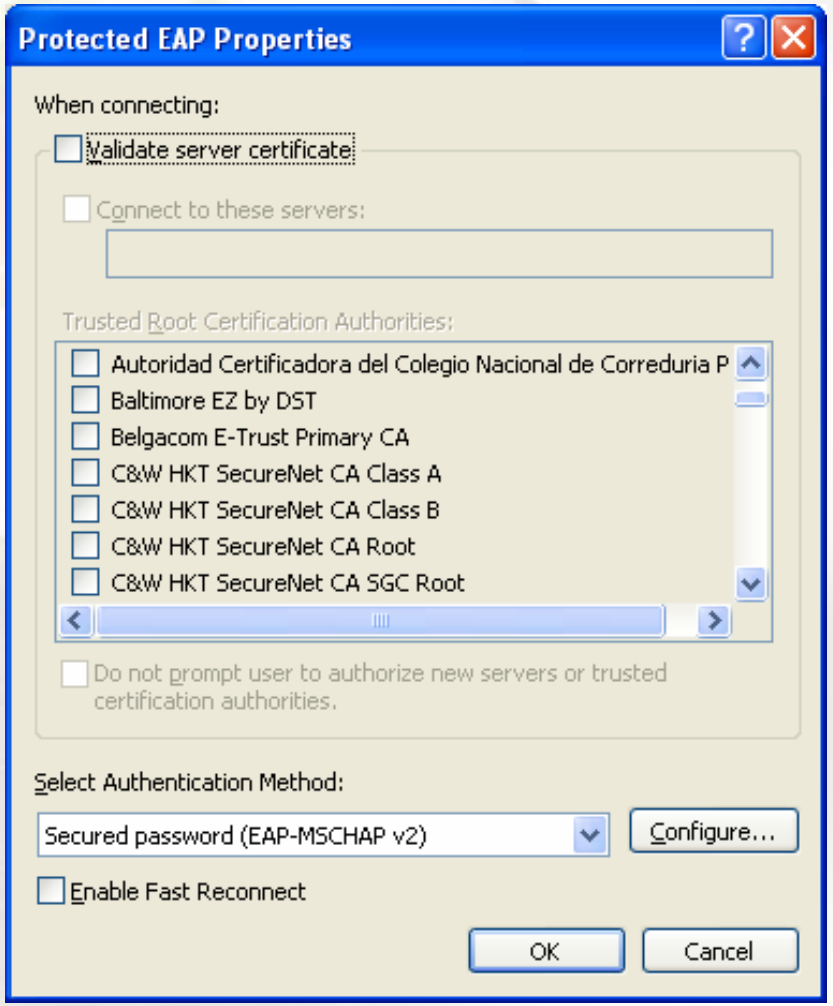
What happens when Joe User clicks "OK"?

PEAP Weakness

- Validation of RADIUS server based on certificate validation
  - Trusted issuing authority, matching CN
- Many PEAP deployments fail to properly deploy
- Malicious RADIUS server grants access to inner authentication methods
  - PEAP: MS-CHAPv2
  - TTLS: MS-CHAPv2, CHAP, PAP, etc.

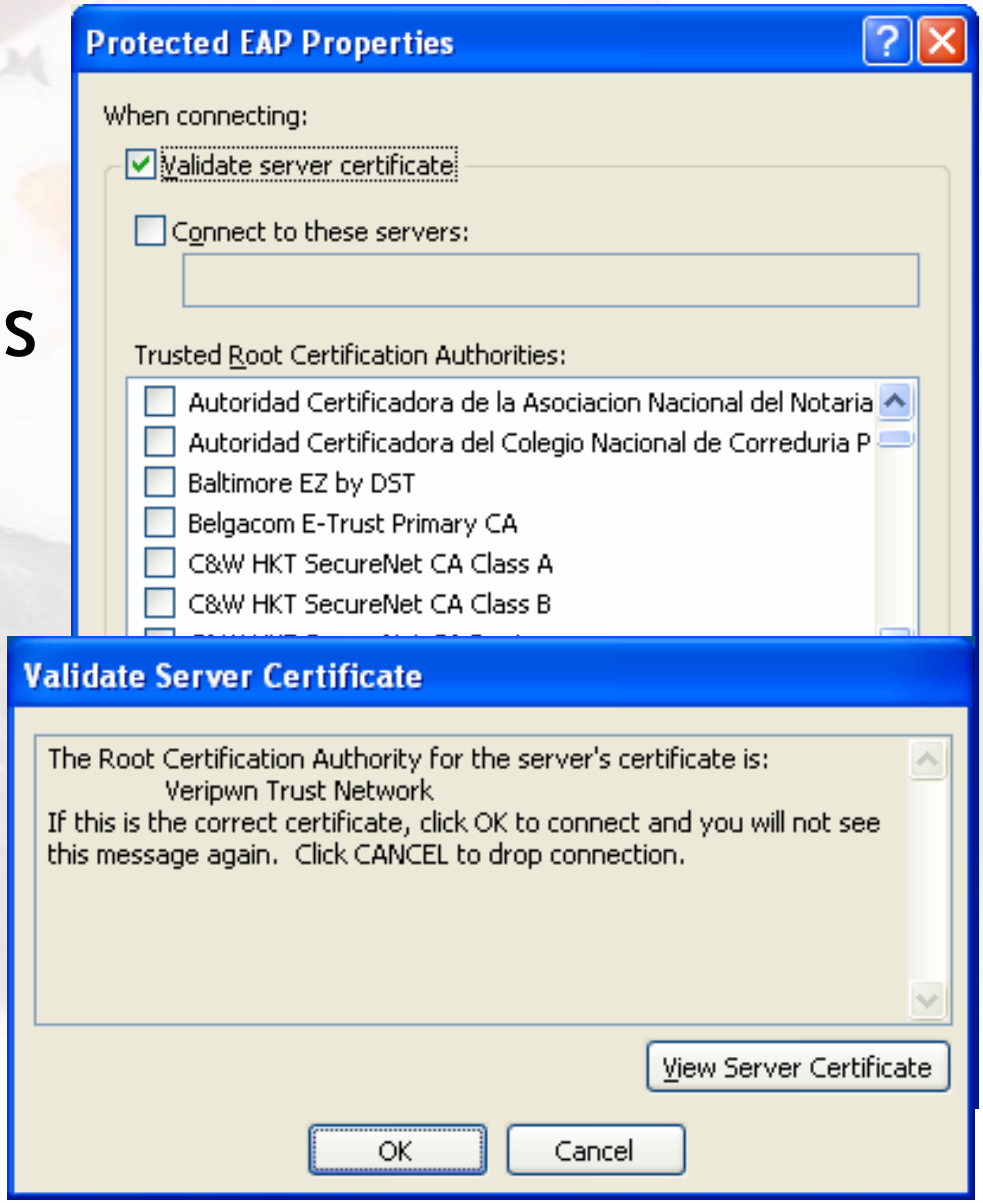
Windows WZC (1)

- Many users disable server certificate validation altogether
- Anyone can impersonate the RADIUS server
- Simple Pwnage, easily attributed to client configuration failure



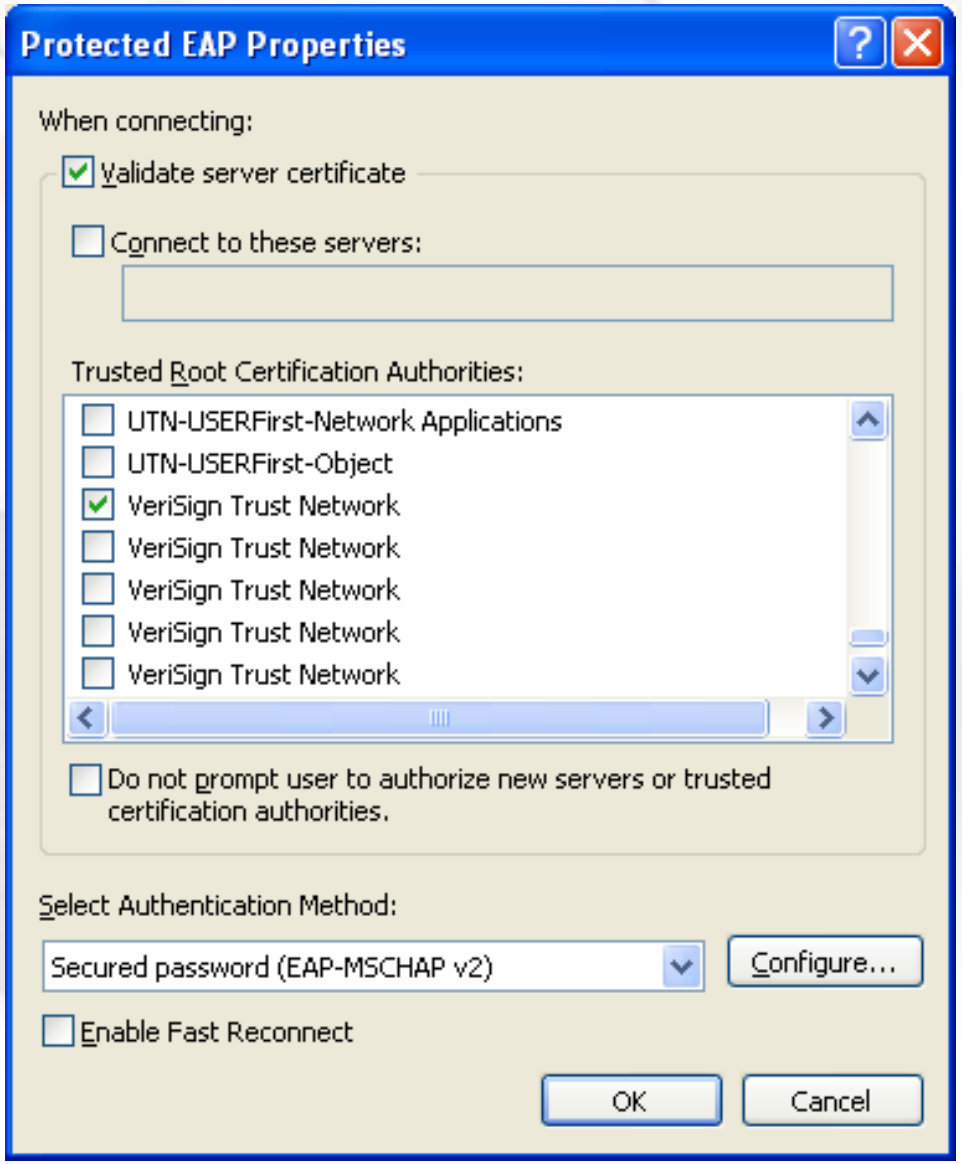
Windows WZC (2)

- Default WZC configuration
- Server certificate is validated
- WZC prompts user to validate server certificate
- Only signing authority is shown in dialog



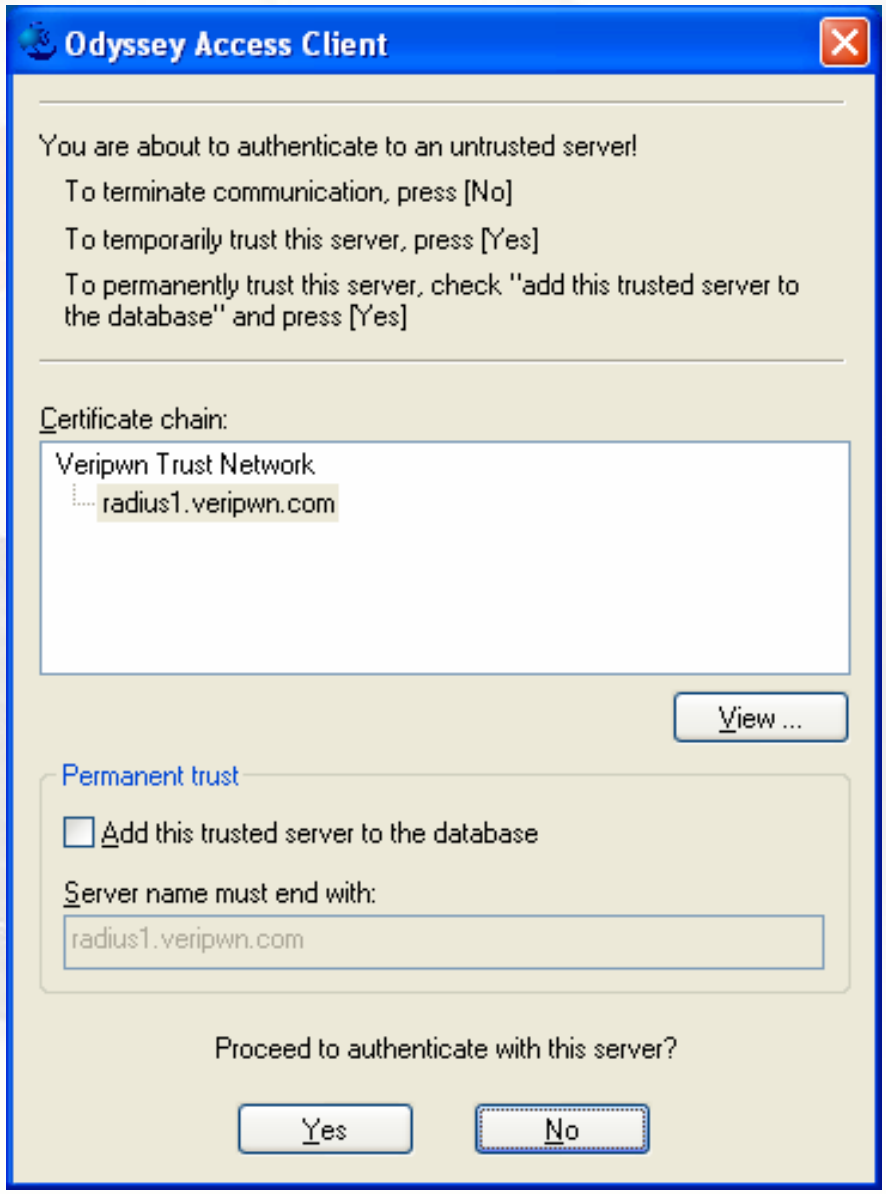
Windows WZC (3)

- Worst possible "valid" configuration for WZC
- Any certificate matching the selected CA is trusted
  - Regardless of CN
- Trivial for attacker to sniff login and identify trusted CA
- Attacker buys cert from trusted CA for any CN

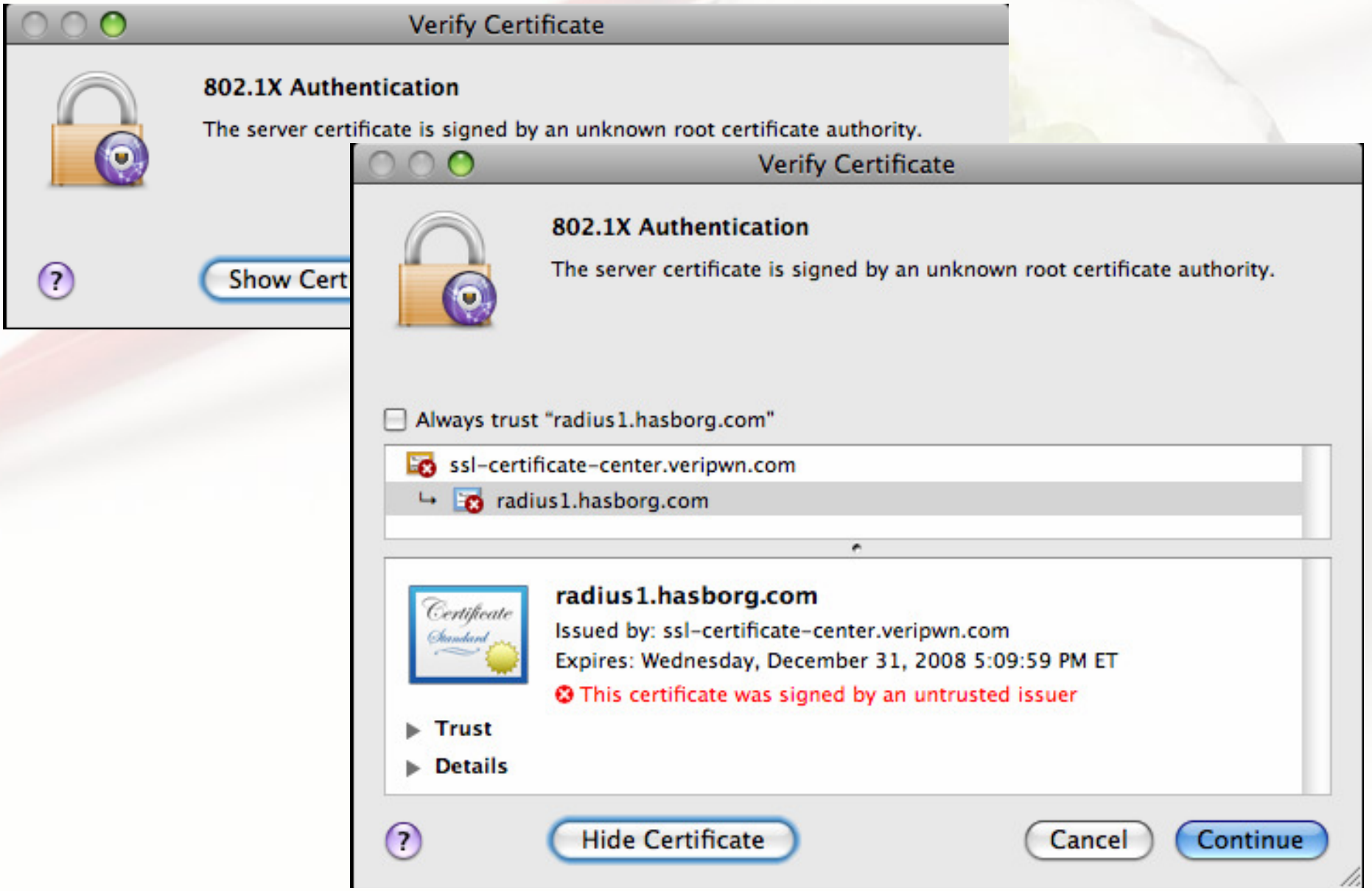


Juniper (Funk) Odyssey

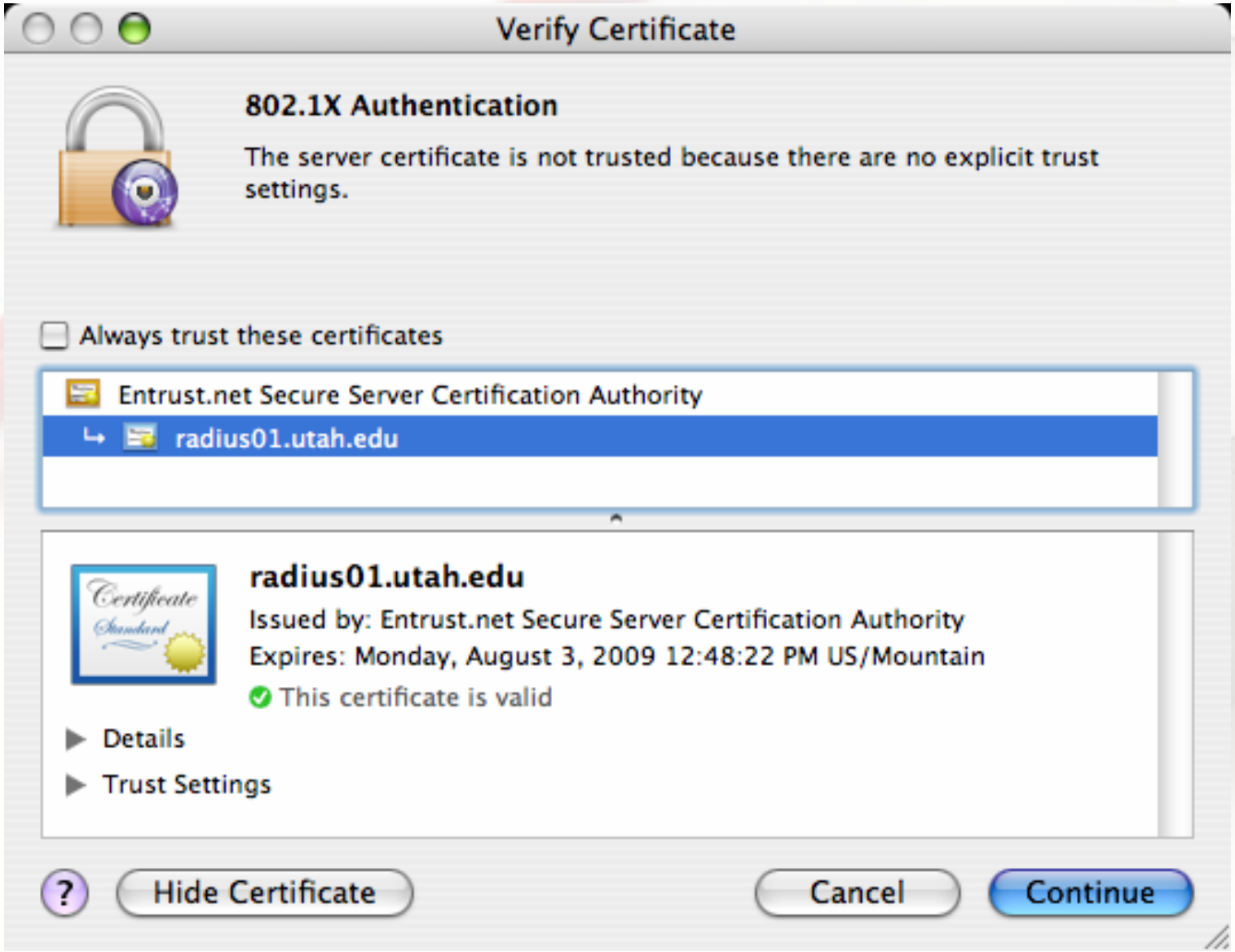
- Does not ship with any trusted CA's
- Administrator must preconfigure trust, or allow users to select trusted/not-trusted
- Prompted each time, or added to stored trust



OSX Supplicant (1)



OSX Supplicant (2)



Attacking PEAP Deployments

- Users often left with decision to trust/reject network
  - "Security in the hands of the end-user"
- Attacker impersonates SSID
  - Untrusted certificate, user decides
  - Trusted certificate in WZC silently accept in some configurations
- Supplicant performs inner authentication with attacker; grants access to exchange

Attacker's RADIUS Server

1. Returns success for any authentication request (to continue authentication exchange)
2. Emulates victim network following authentication (e.g. KARMA)
3. Logs authentication credentials (challenge/response, password, username)
4. Potential to accelerates credential cracking with fixed challenge

freeradius-wpe

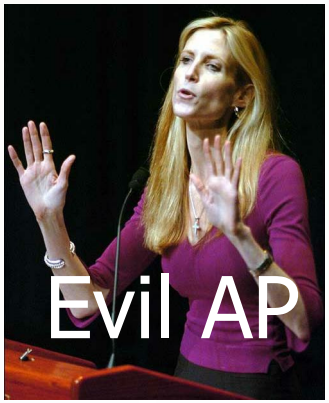
- Patch for FreeRADIUS 2.0.2
- Adds logging for authentication credentials
  - TTLS/PAP: Username/password
  - TTLS/CHAP: Challenge/response
  - PEAP/MS-CHAPv2: Challenge/response
  - A few others
- Returns success for any credentials where possible

FreeRADIUS WPE

- Setting up rogue RADIUS in 8 easy steps
- Setup AP using RFC1918 address, RADIUS shared secret of "test"
- Logging in /usr/local/var/log/radius/freeradius-server-wpe.log

```
$ tar xvfj freeradius-server-2.0.2.tar.bz2
$ cd freeradius-server-2.0.2/
$ patch -p1 < ../freeradius-wpe-2.0.2.patch
$ ./configure && make && sudo make install && sudo ldconfig
# cd /usr/local/etc/raddb/certs
# ./bootstrap
# radiusd
# tail -f /usr/local/var/log/radius/freeradius-server-wpe.log
```

Combining Tools




```
polonium radius # tail -f freeradius-server-wpe.log
mschap: Sat Feb  2 22:10:08 2008

        username: hrollins
        challenge: 08:92:54:d7:3c:33:c7:b7
        response: bb:6e:8f:4f:57:c8:da:71:3e:e4:91:a7:
        58:79:ac:5a:a9:53:36:05:ba
```

I love you Annie

Unsuspecting victim

```
jwright@polonium ~/asleap-2.1 $ ./asleap -f dict.dat -n dic
t.idx -C 08:92:54:d7:3c:33:c7:b7 -R bb:6e:8f:4f:57:c8:da:71
:3e:e4:91:a7:dd:40:df:58:79:ac:5a:a9:53:36:05:ba
asleap 2.1 - actively recover LEAP/PPTP passwords. <jwright
@hasborg.com>
        hash bytes:             00cc
        NT hash:                ac8e657f83df82beea5d43bdaf7800cc
        password:               anncoulter
jwright@polonium ~/asleap-2.1 $
```

DEMO

Are PEAP and TTLS Broken?

- No, PEAP and TTLS can be secure when deployed carefully
- Caution in configuring supplicants
  - Distribute private CA certificate, or buy from a public CA
  - Always validate server certificate
  - Manually identify CN's of authorized RADIUS servers
- Is my supplicant secure?
  - Supplicants must include a feature to reject (not prompt) RADIUS CN's that do not match
  - Odyssey, WZC accommodate this today

Proper WZC Supplicant Config

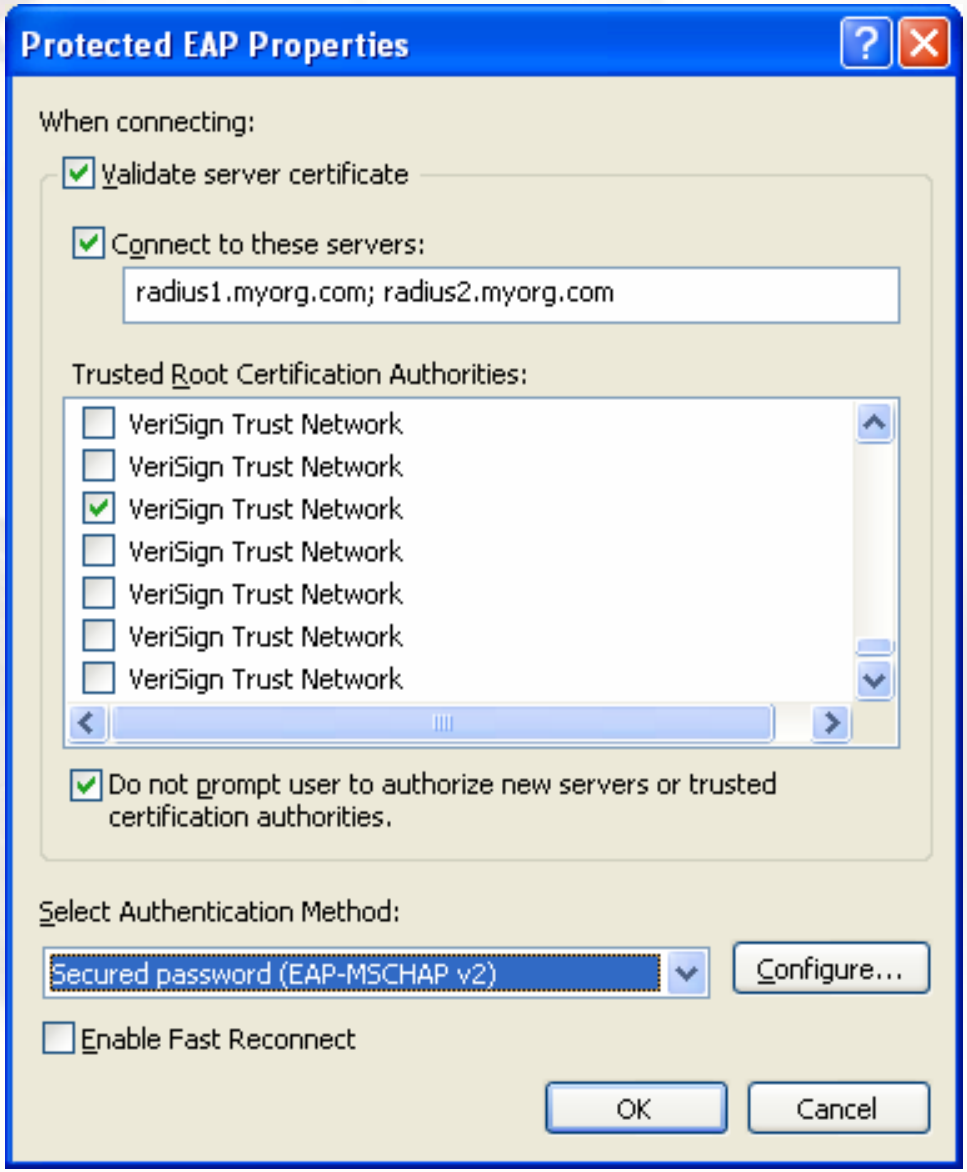
Always validate certificate

Specify CN on certificate(s)

Specify trusted CA

Forbid user from adding new trusted RADIUS servers

Microsoft KB941123: "How to configure PEAPv0 to reduce potential risks against man-in-the-middle attacks and against password-based attacks when you use authentication servers in Windows Vista or in Windows Server 2008"



Summary

- Evolution of WLAN security relies on strong EAP types for authentication
- EAP-MD5, LEAP should not be used
- EAP-FAST suffers from complexity or weak security in PAC provisioning
- Common PEAP/TTLS deployments are secure
  - Can be fixed with careful deployment steps
- Tools/patches at willhackforsushi.com

Knowledge helps us all to defend our networks

Questions?

ShmooCon 2008
Joshua Wright, jwright@willhackforsushi.com
Brad Antoniewicz, Brad.Antoniewicz@foundstone.com

Code at www.willhackforsushi.com/offensive.html (Monday)
Brad's Paper at www.foundstone.com

Extra Stuff

Stuff we moved to the end of the presentation for time consideration

MS-CHAPv1 Challenged

- Normal MS-CHAPv1 behavior:
  1. RADIUS→STA: 8-byte challenge
  2. STA→RADIUS: DES(challenge) *3, return 24-byte response
  3. RADIUS compares observed response to calculated response
- Attacker knows challenge and response, challenge acts as a "salt"
- Pwned MS-CHAPv1 behavior:
  1. RADIUS→STA: Fixed challenge "00000000"

Removing random challenge allows attacker to implement a precomputed lookup table of responses for a given hash

LEAP or TTLS/MS-CHAP Attack

- Fixed challenge from attacker removes uniqueness ("salt") from exchange
- Accommodates RainbowTable attack using challenge/response

```
$ ./rcrack mschap_loweralpha#8-8_1_256x10000_mschap.rt -h
9bb1789e3e1224c563bab42517dd097d3dd4de4498d3d3a1
searching for 1 hash...
plaintext of 9bb1789e3e1224c563bab42517dd097d3dd4de4498d3d3a1 is
pjpxwijt
cryptanalysis time: 0.00 s
statistics
----------------------------------------------------------
plaintext found:           1 of 1 (100.00%)
total disk access time:    0.00 s
total chain walk step:     36
----------------------------------------------------------
9bb1789e3e1224c563bab42517dd097d3dd4de4498d3d3a1  pjpxwijt
hex:706a707877696a74
```