EAP-TLS THE ROLLS-ROYCE OF EAP METHODS

ENHANCEMENTS IN FREERADIUS 3.0.0-3.2.0

NWS 44

We are the FreeRADIUS experts.

networkRADIUS

WHOAMI

- Arran Cudbard-Bell (@arr2036)
- FreeRADIUS Core Developer
- Mercenary at Network RADIUS
- Director RM-RF LTD
- IETF Note Taker (RADEXT)
- Janet 802.1X SIG member

WHATIS FREERADIUS

- The world's most widely deployed Open Source RADIUS server.
- Glues AAA services to backends e.g. 802.1X/EAP to Active Directory.
- Routes AAA authentication sessions between members of federations like Eduroam.
- Adds intelligence to dumb protocols, using flexible policies.

http://freeradius.surge.sh

TOPICS

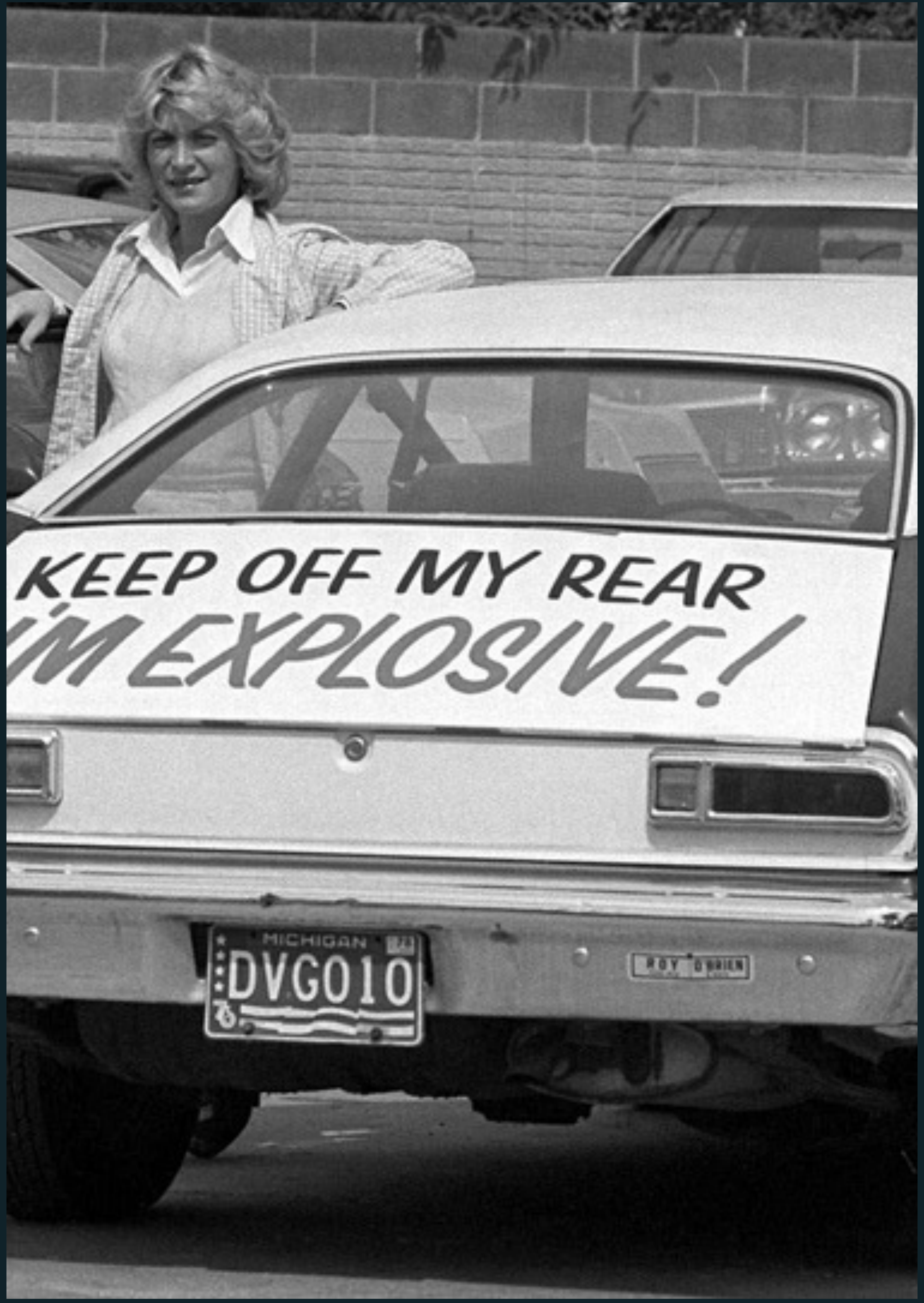
- PEAP - The Ford Pinto of EAP methods.
- EAP-TLS - The Rolls-Royce of EAP methods.
- What's new in v3.0.x / v3.2.0.

PART 1

PEAP - THE FORD PINTO OF EAP METHODS

WHY IS PEAP INSECURE?

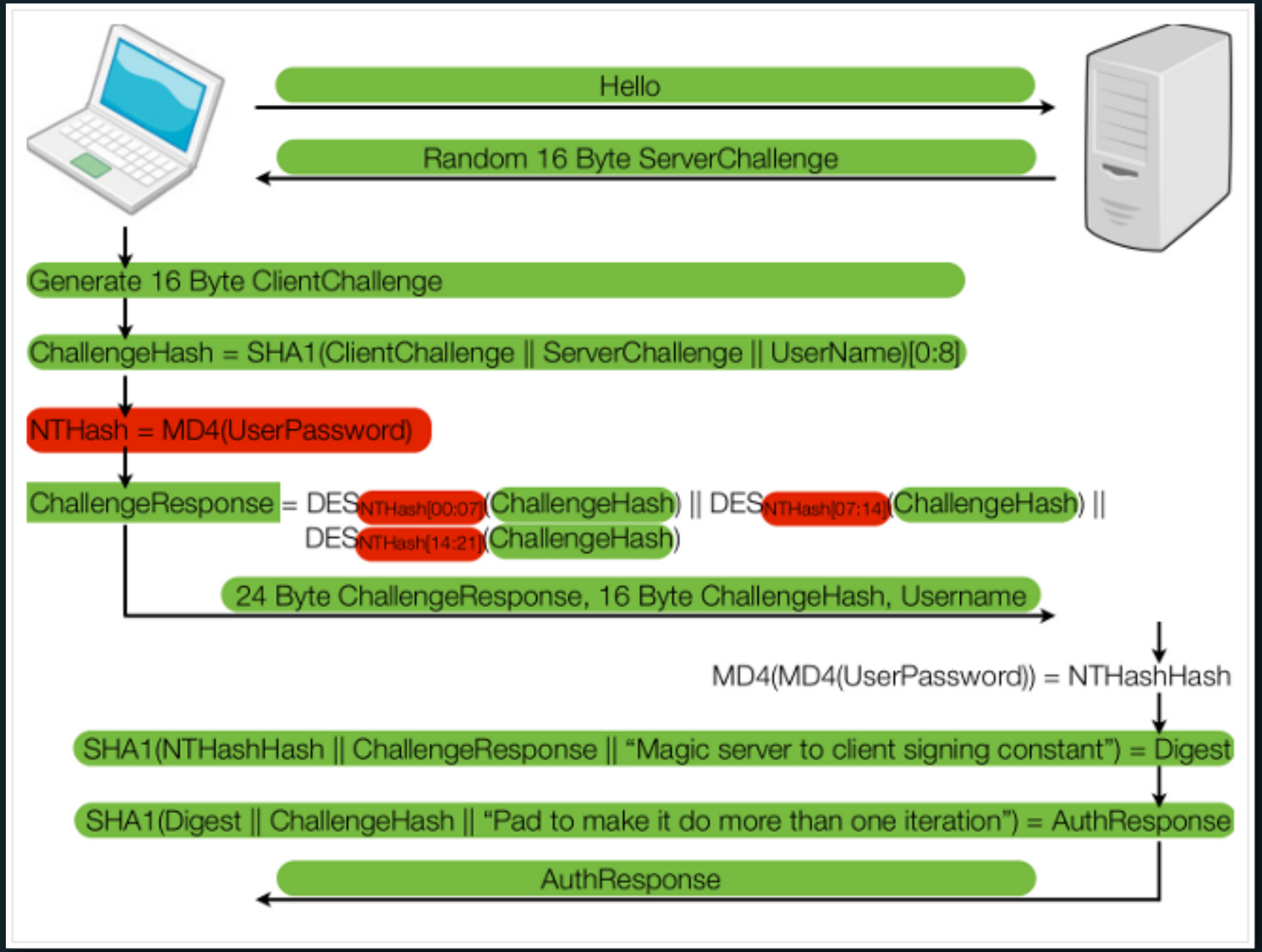
- MSCHAPv2 is broken, must be wrapped in TLS.
- TLS only protects data if peer is not evil.
- Ensuring peer is not evil requires a trust relationship.
- Trust relationship during bootstrap requires PKI savvy users.
- Users are not PKI savvy.



PEAP - THE FORD PINTO OF EAP METHODS

NOT NEWS

- Presented at Defcon 20 (2012) by David Hulton.
- 16 byte MD4 hash is as good as Cleartext for MSCHAPv2 (only thing the server knows).
- We know the ChallengeHash, need to guess the 3 * 7 bytes NT HASH fragments used as DES keys.
- That's a 2^{138} bit keyspace! Eeek!
- Wait... 21 != 16 (the other five are zeros)... so that's only two DES keys we need to find!
- ...and the cipher key is the same for all DES operations, so we can brute force all keys simultaneously.
- Which gives us a 2^{56} bit keyspace, which can be broken by online DES cracking services in < 24 hours.



Hello

Random 16 Byte ServerChallenge

Generate 16 Byte ClientChallenge

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

NTHash = MD4(UserPassword)

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

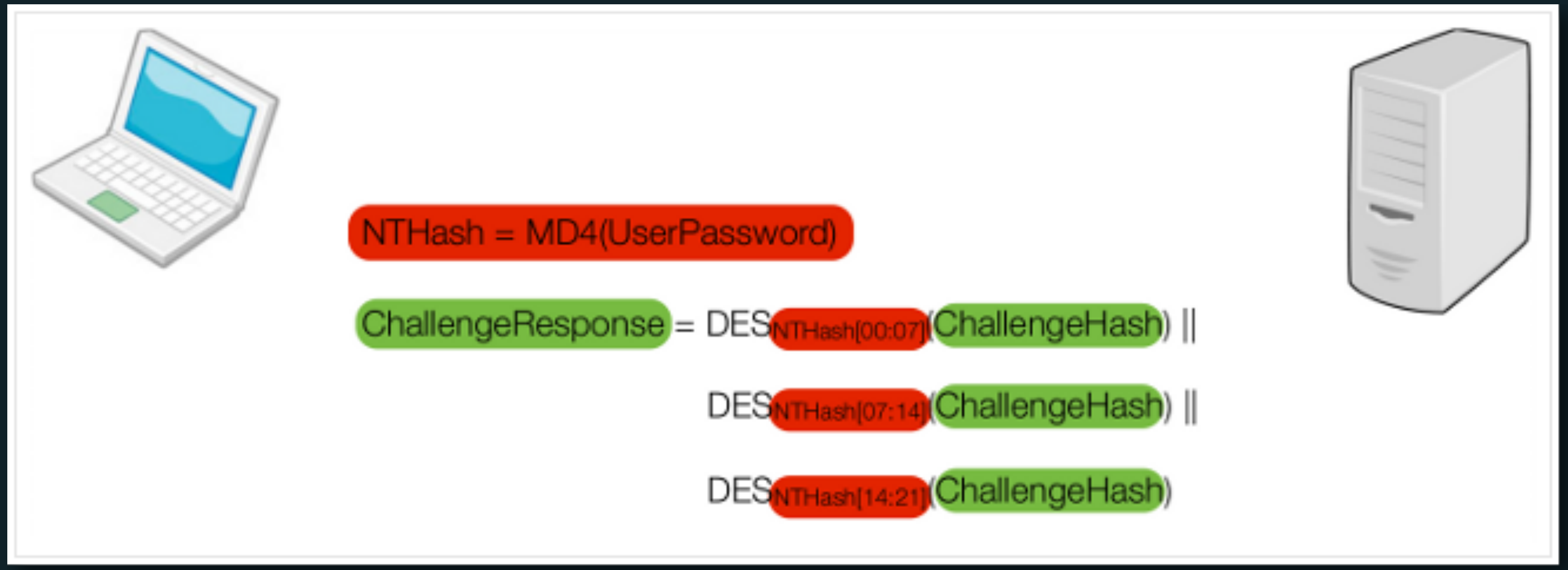
24 Byte ChallengeResponse, 16 Byte ChallengeHash, Username

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



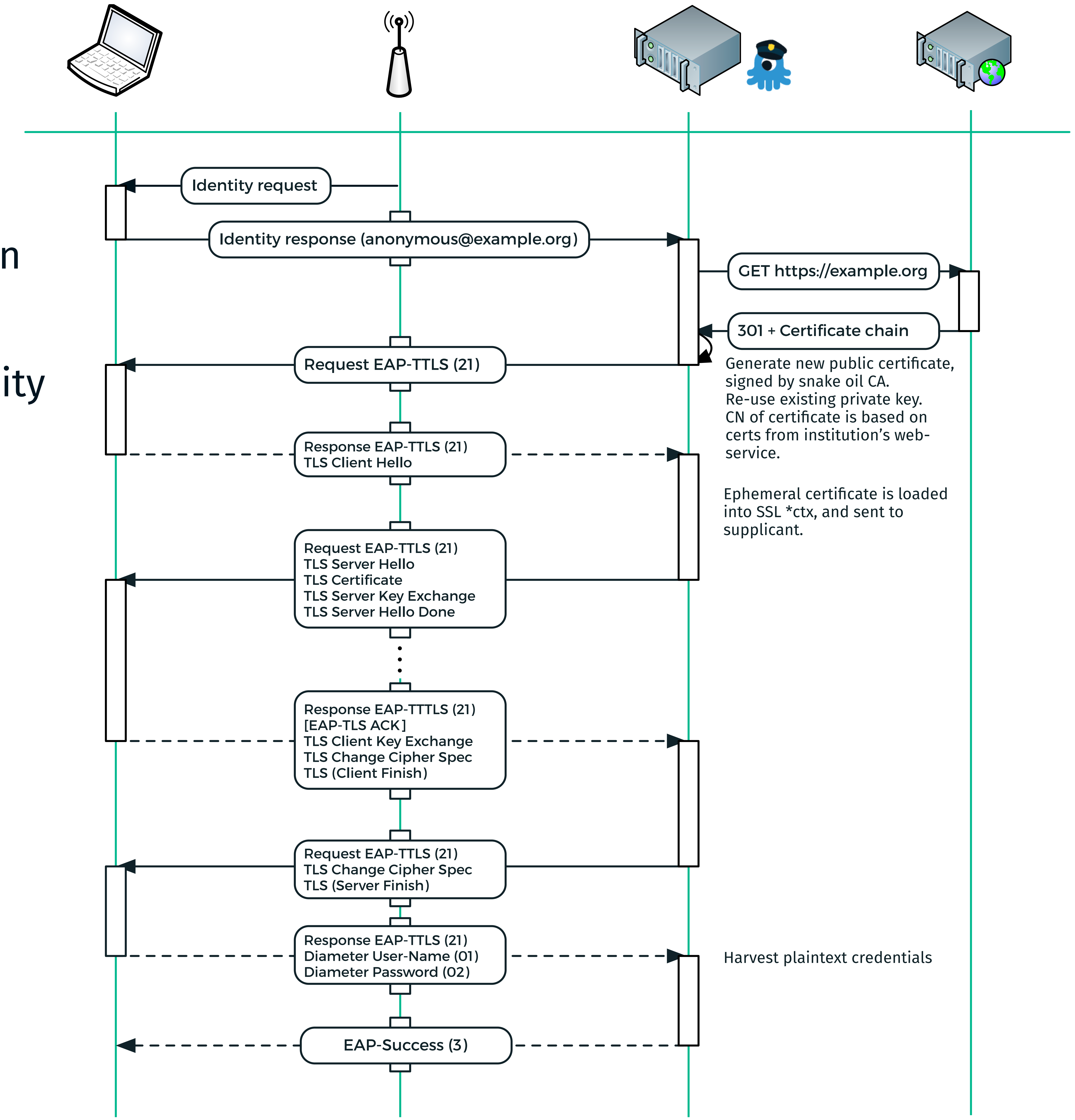
NTHash = MD4(UserPassword)

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

Images by Moxie Marlinspike - Divide and Conquer: Cracking MSCHAPv2 with a 100% success rate
Retrieved from https://www.cloudcracker.com/blog/2012/07/29/cracking-ms-chap-v2/

NOT JUST PEAP

- Anything that relies on MSCHAPv2 for confidentiality is broken e.g. LEAP
- Any insecure inner method that relies on TLS for confidentiality is also broken. e.g.
  - EAP-TTLS-PAP
  - EAP-TTLS-MSCHAPv2
  - EAP-TTLS-GTC
  - PEAPv1-GTC
- For OSX, IOS, and Windows > 8, it's possible to request TTLS-EAP-GTC or TTLS-PAP and get the cleartext password.
- Attacks can be made more convincing by generating certificates on the fly, from the NAI in the identity response.



Identity request

Identity response (anonymous@example.org)

GET https://example.org

301 + Certificate chain

Generate new public certificate, signed by snake oil CA.
Re-use existing private key.
CN of certificate is based on certs from institution's web-service.

Request EAP-TTLS (21)

Response EAP-TTLS (21)
TLS Client Hello

Ephemeral certificate is loaded into SSL *ctx, and sent to supplicant.

Request EAP-TTLS (21)
TLS Server Hello
TLS Certificate
TLS Server Key Exchange
TLS Server Hello Done

Response EAP-TTTLS (21)
[EAP-TLS ACK]
TLS Client Key Exchange
TLS Change Cipher Spec
TLS (Client Finish)

Request EAP-TTLS (21)
TLS Change Cipher Spec
TLS (Server Finish)

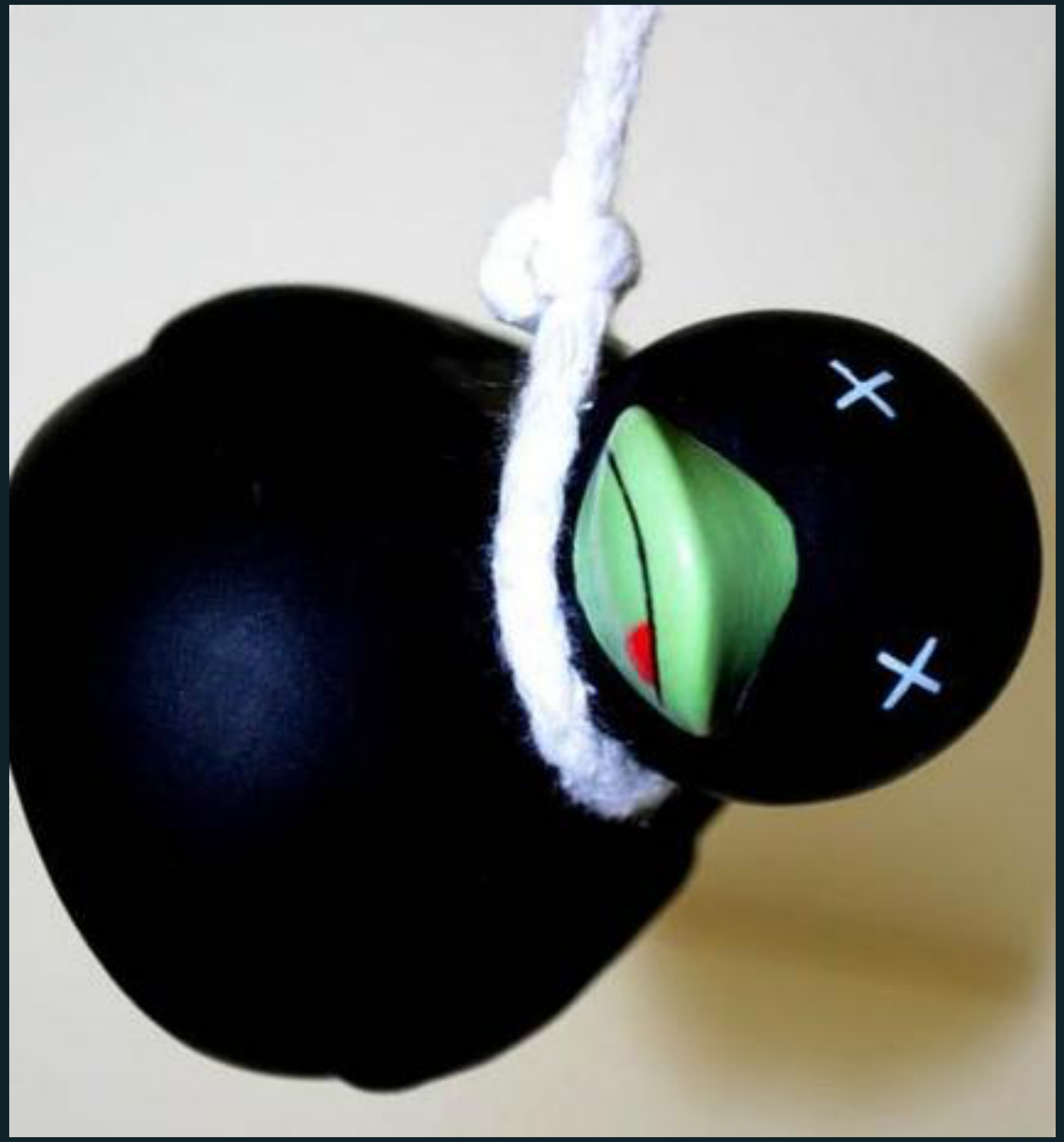
Response EAP-TTLS (21)
Diameter User-Name (01)
Diameter Password (02)

Harvest plaintext credentials

EAP-Success (3)

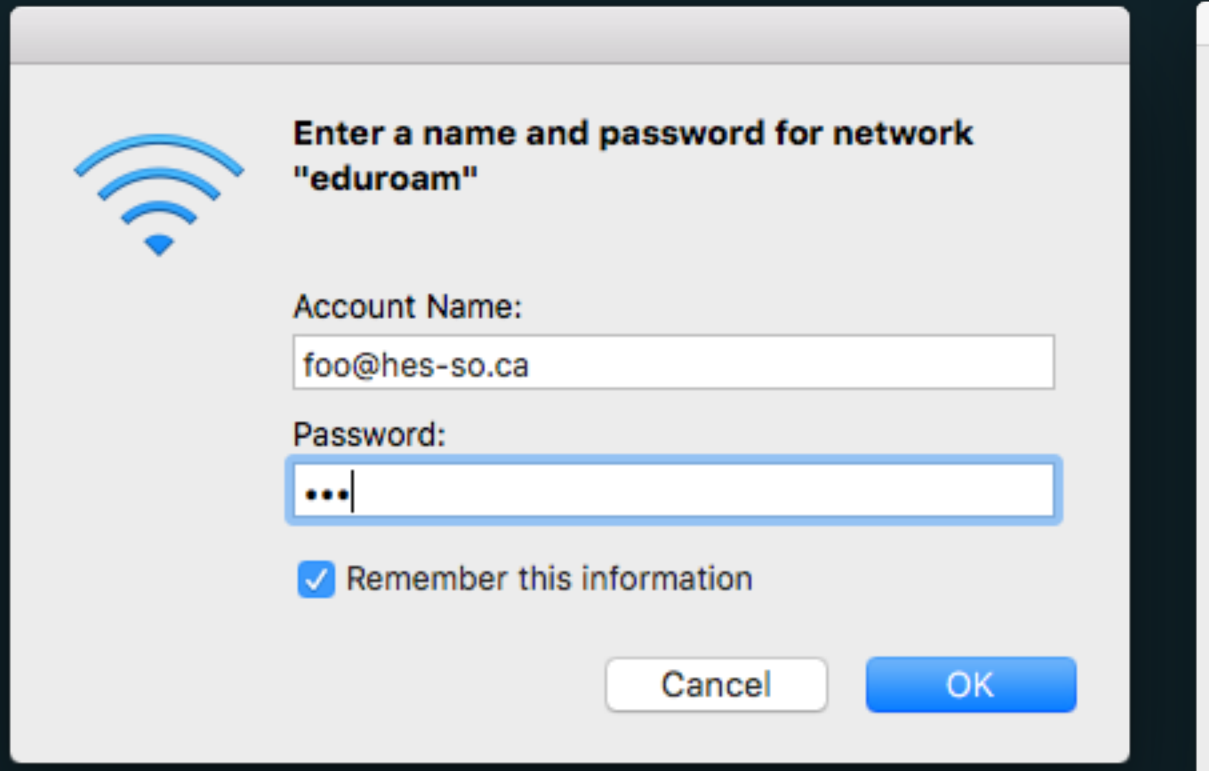
FAILURE OF THE DUCK TEST

- Only method of authenticating wireless network is SSID (which isn't really authentication).
- Only method of authenticating EAP server is by presented certificate (fingerprint, CN and signing CA).
- Supplicants give users too much and too little power.



FOXING FERRETS ON OSX EL-CAPITAN

- IOS/OSX supplicants prompt for User-Name/Password before negotiating the EAP method.
- No option to select personal certificate.
- No option to manually configure supplicant profiles.
- Only CN of certificate shown in UI (can expand to see full details)
- Trivial to click past certificate verification dialogues, but you at least need to be able to change trust preferences.
- When TTLS is requested, supplicant will send EAP-Identity and trigger EAP negotiation, allowing negotiation of EAP-GTC.
- Unless network/supplicant settings were defined by a profile, cached credentials will be re-used on networks with the same name, but presenting a different cert.

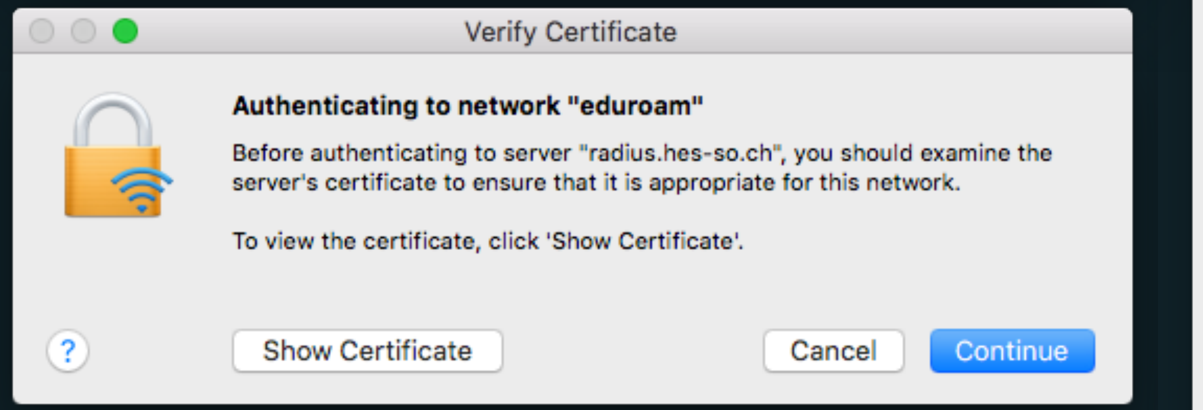


Enter a name and password for network "eduroam"

Account Name: foo@hes-so.ca

Password: •••

☑ Remember this information

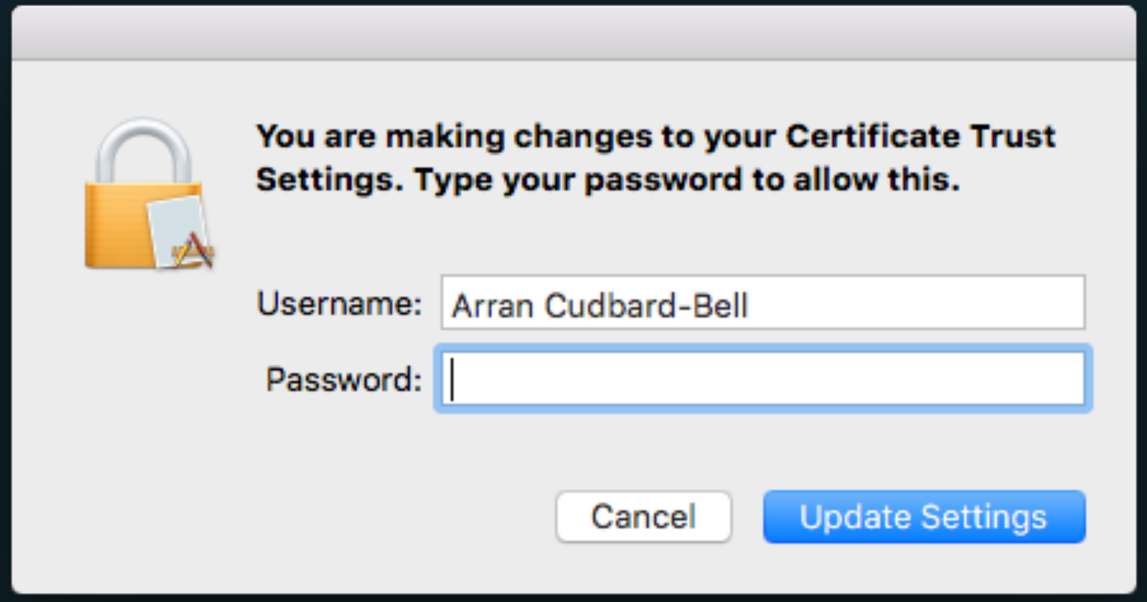


Verify Certificate

Authenticating to network "eduroam"

Before authenticating to server "radius.hes-so.ch", you should examine the server's certificate to ensure that it is appropriate for this network.

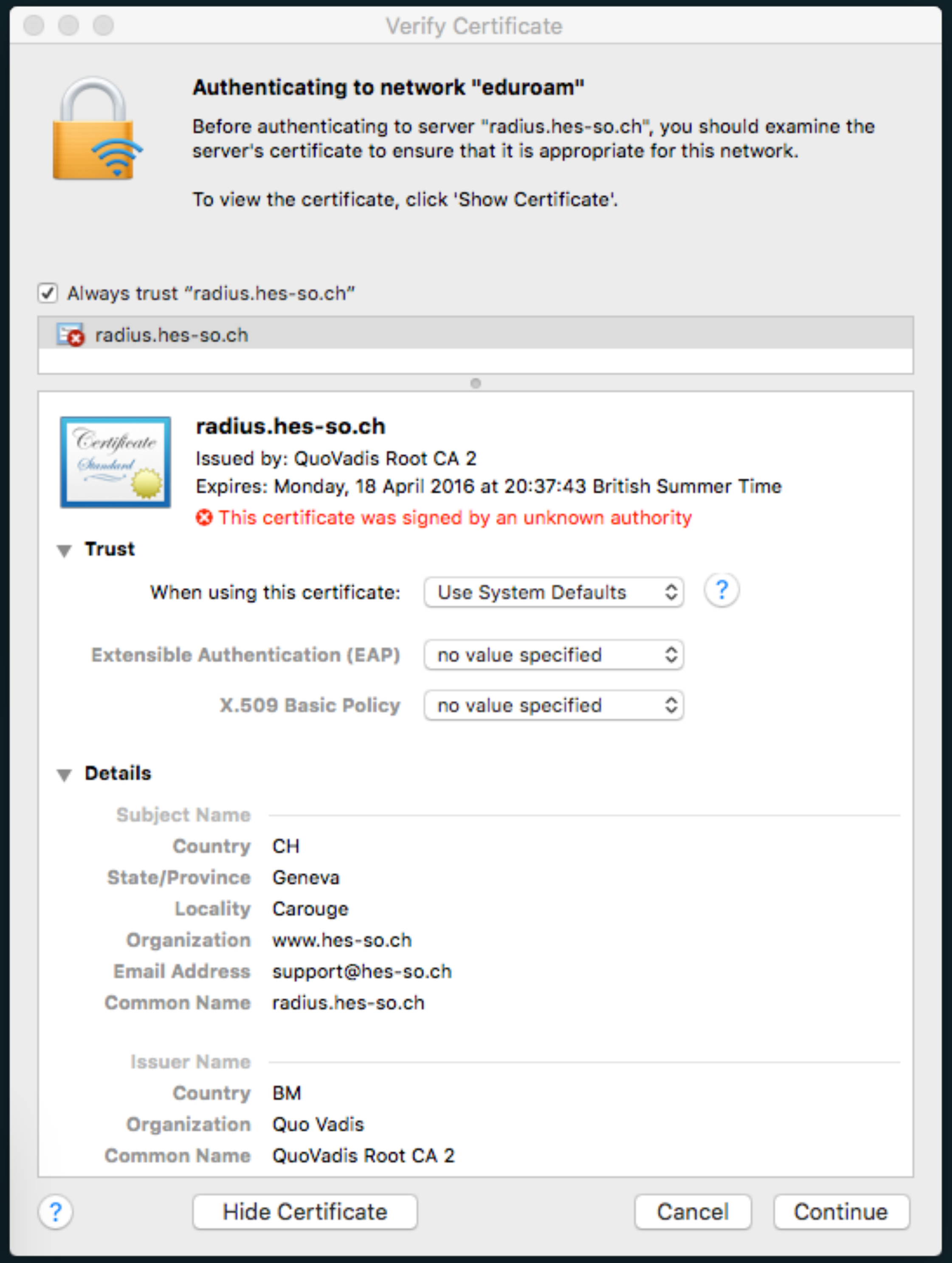
To view the certificate, click 'Show Certificate'.



You are making changes to your Certificate Trust Settings. Type your password to allow this.

Username: Arran Cudbard-Bell

Password:



Verify Certificate

Authenticating to network "eduroam"

Before authenticating to server "radius.hes-so.ch", you should examine the server's certificate to ensure that it is appropriate for this network.

To view the certificate, click 'Show Certificate'.

☑ Always trust "radius.hes-so.ch"

radius.hes-so.ch

radius.hes-so.ch
Issued by: QuoVadis Root CA 2
Expires: Monday, 18 April 2016 at 20:37:43 British Summer Time
☒ This certificate was signed by an unknown authority

Trust

When using this certificate: Use System Defaults

Extensible Authentication (EAP): no value specified

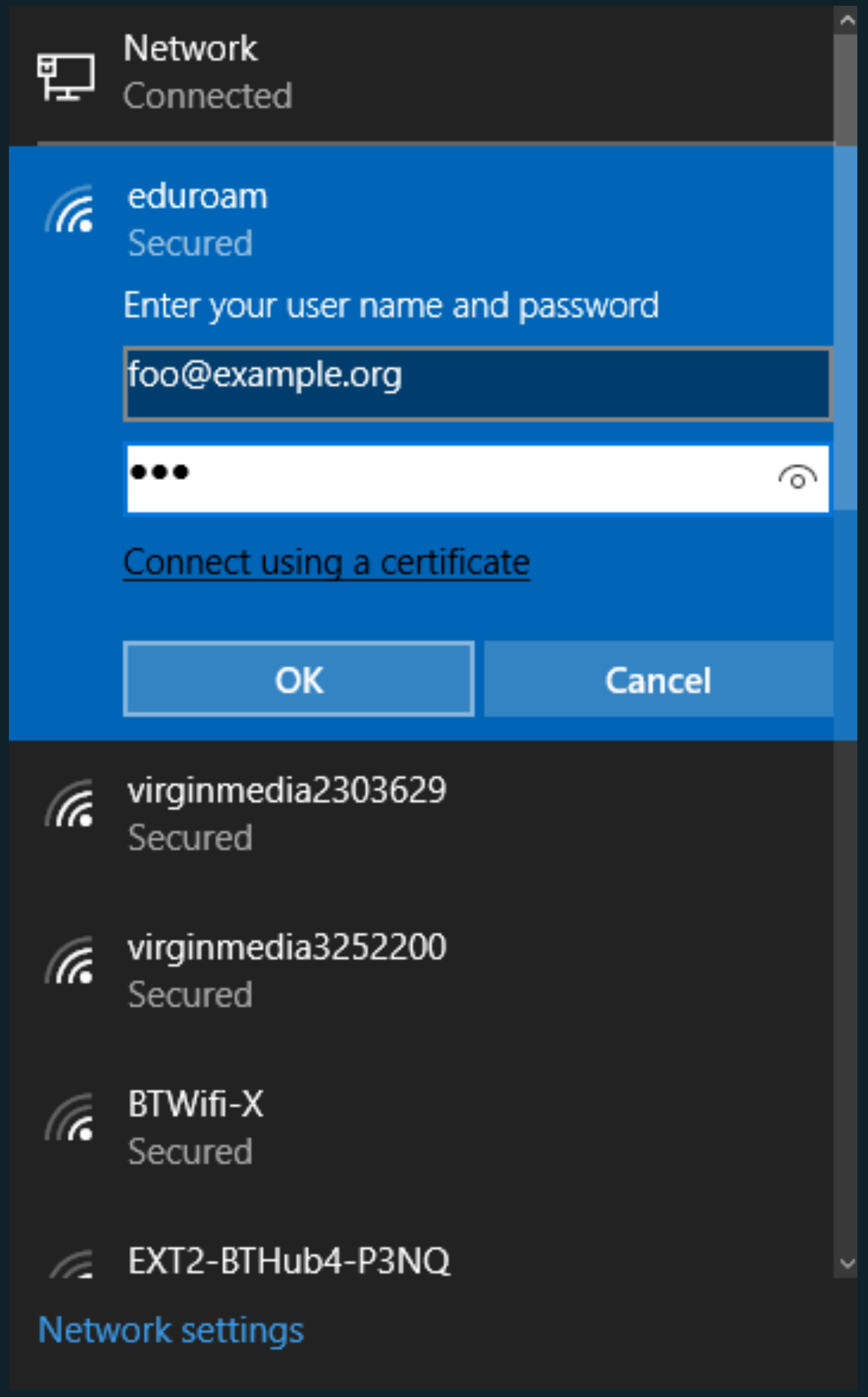
X.509 Basic Policy: no value specified

Details

Subject Name	
Country	CH
State/Province	Geneva
Locality	Carouge
Organization	www.hes-so.ch
Email Address	support@hes-so.ch
Common Name	radius.hes-so.ch
Issuer Name	
Country	BM
Organization	Quo Vadis
Common Name	QuoVadis Root CA 2

WAYLAYING WEASELS ON WINDOWS 10

- For unknown networks Windows 10 supplicants auto discover WPA-Enterprise, and prompt for User-Name/Password even before negotiating the EAP method.
- No way to see certificate CN or issuer, only available detail is fingerprint.
- When TTLS is requested, supplicant will perform EAP-TTLS-PAP by default. Can't negotiate EAP unless explicitly configured.
- Does allow manual configuration of supplicant, but exceedingly well hidden.



Network
Connected

eduroam
Secured

Enter your user name and password

foo@example.org

•••

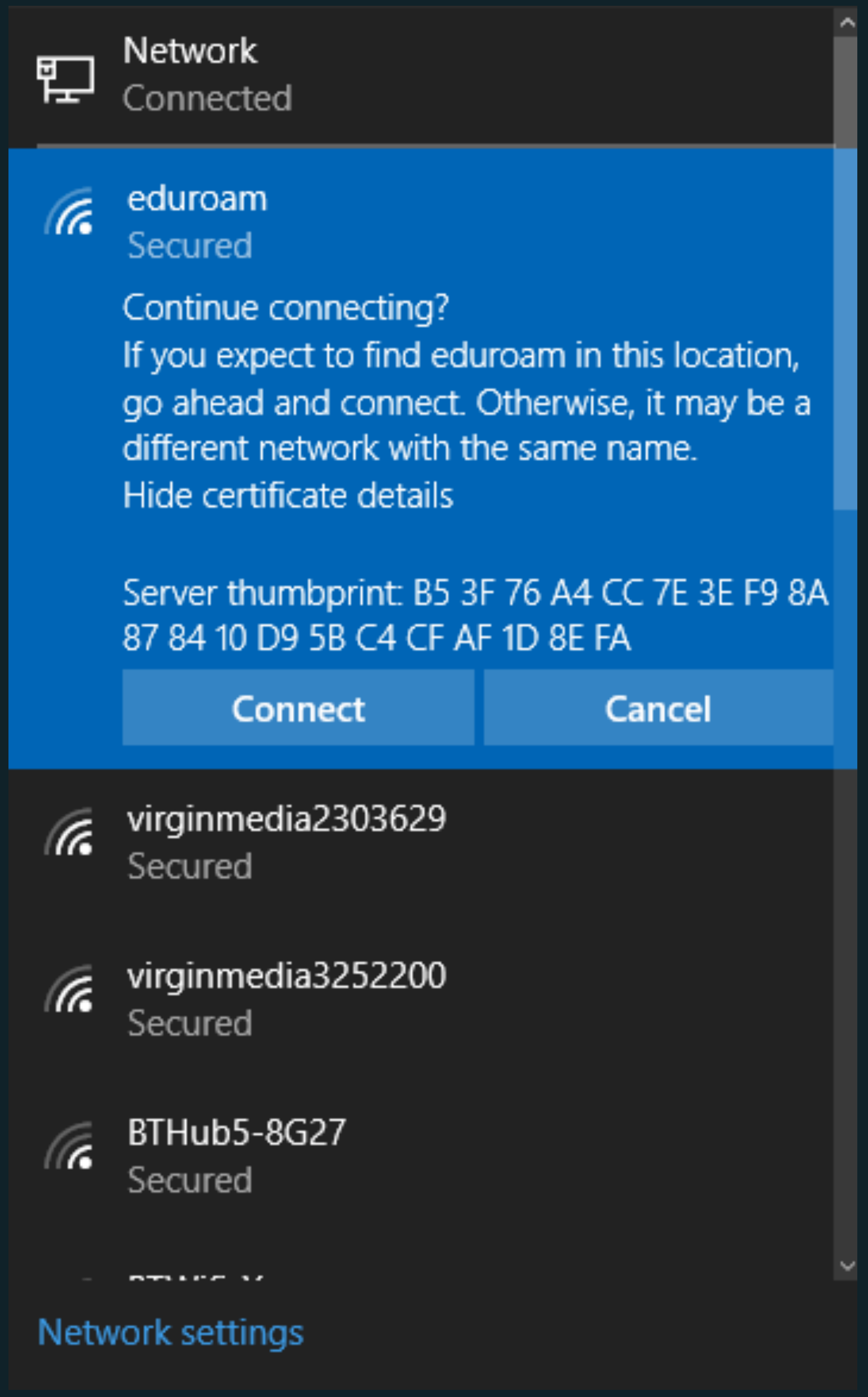
Connect using a certificate

virginmedia2303629
Secured

virginmedia3252200
Secured

BTWifi-X
Secured

EXT2-BTHub4-P3NQ



Network
Connected

eduroam
Secured

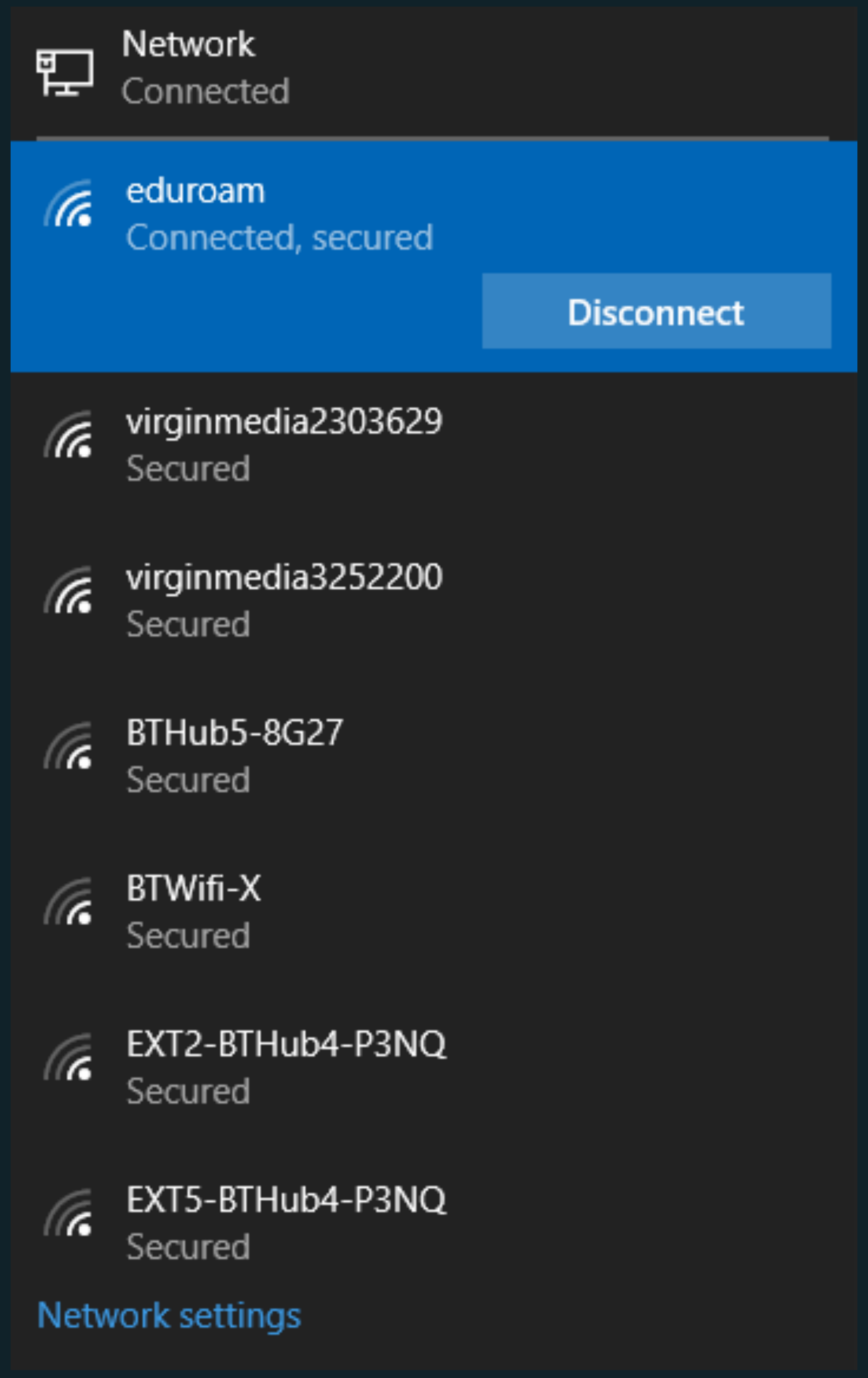
Continue connecting?
If you expect to find eduroam in this location, go ahead and connect. Otherwise, it may be a different network with the same name.
Hide certificate details

Server thumbprint: B5 3F 76 A4 CC 7E 3E F9 8A 87 84 10 D9 5B C4 CF AF 1D 8E FA

virginmedia2303629
Secured

virginmedia3252200
Secured

BTHub5-8G27
Secured



Network
Connected

eduroam
Connected, secured

virginmedia2303629
Secured

virginmedia3252200
Secured

BTHub5-8G27
Secured

BTWifi-X
Secured

EXT2-BTHub4-P3NQ
Secured

EXT5-BTHub4-P3NQ
Secured

PREVENTING MISIDENTIFICATION OF WATERFOWL

- Eduroam sites
  - Transition to EAP-TLS (please?).
  - Consider deploying Eduroam CAT or similar.
  - Adopt HotSpot 2.0 R2. Register interest in OSU (Online Signup Server) certs provided by central authority (GÉANT/Jisc).
  - Pre/post-flight checks (verify supplicants behave correctly).
- OS/supplicant vendors
  - Should never involve users in PKI validity checks.
  - Failing that - Cert fingerprint MUST be consistent when re-using cached credentials for AD-Hoc 802.1X profiles.
- IETF/standards bodies
  - Define strongly worded guidelines for supplicant implementors (http://geant3plus.archive.geant.net/Resources/Open_Call_deliverables/Documents/SENSE_final_report.pdf)

WHY MOVE TO EAP-TLS?

- Extremely secure - Reduces chance of user's credentials being exposed.
- Efficient - Almost half the round trips compared to PEAP.
- Robust - No need to query external oracle (AD, LDAP, SQL) for authenticating users.
- Scales horizontally.
- Well supported - Windows, OSX, IOS, Android, Even HP printers (and better supported in future with over the air certificate deployment).
- Should allow for proper 2FA (in future). Linux (wpa_supplicant) already supports client certs for PEAP/TTLS. Feature request made to Microsoft.
- PEAP-EAP-TLS allows for identity privacy (and SoH).

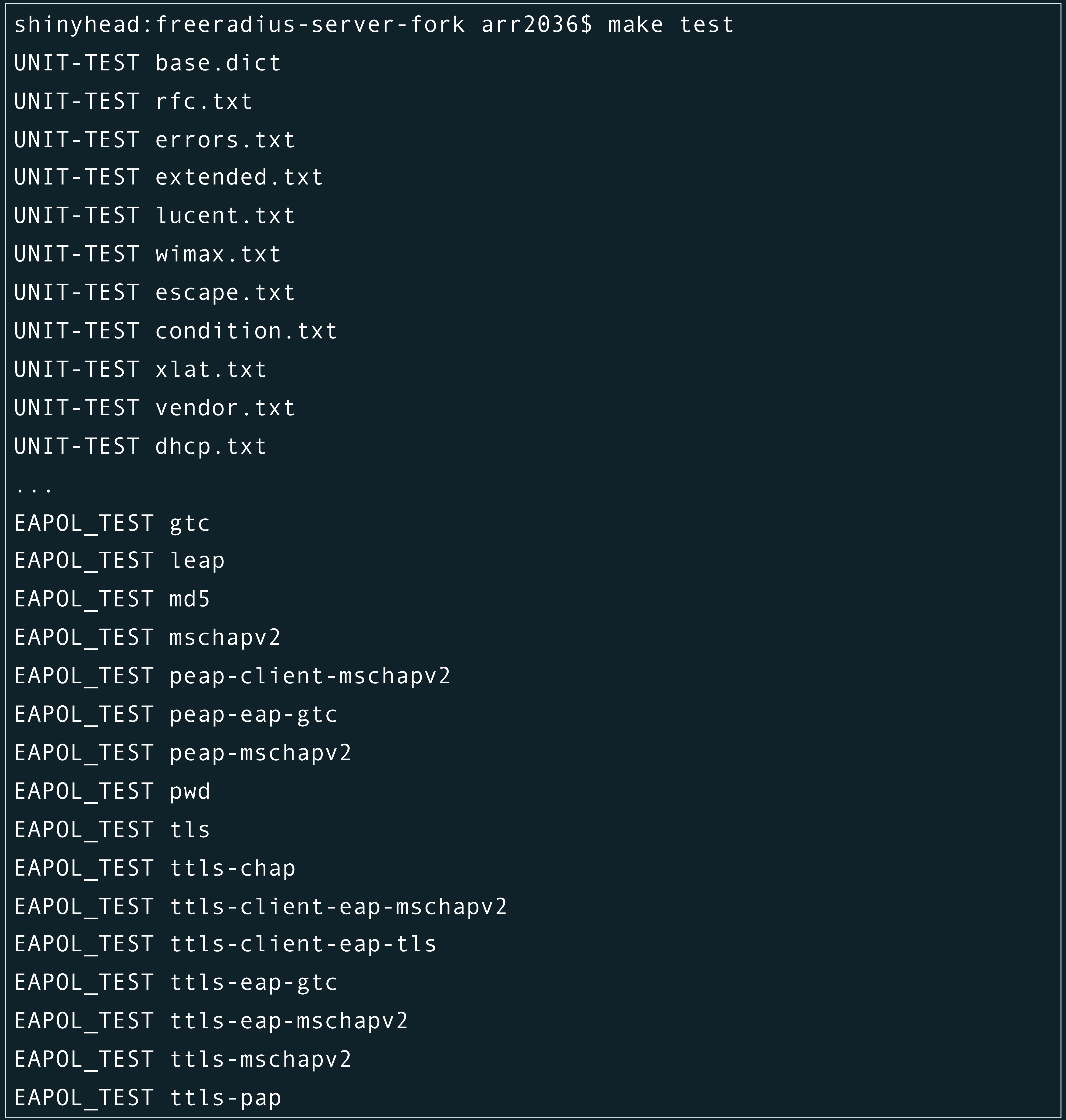
EAP-TLS PKI WHAT ARE THE OPTIONS?

- Two established commercial solutions for managing client PKI
  - Cloudpath
  - Clearpass (FreeRADIUS FTW!)
- EST (Enrolment Over Secure Transport) RFC 7030 - Fairly simple to integrate, but only reference implementations available.
- SCEP (Secure Certificate Enrolment Protocol) now revived as https://tools.ietf.org/html/draft-gutmann-scep-02.
  - Many implementations OpenSCEP, EJCBA and Dogtag most well known.
  - Good support from Apple (iOS/OSX) they provide full reference code for Ruby based SCEP server.
- Eduroam CAT - Local PKI (as in local to the CAT server) on roadmap, considering RFC 7030/SCEP support.

PART 2

WHATS NEW V3.0.X

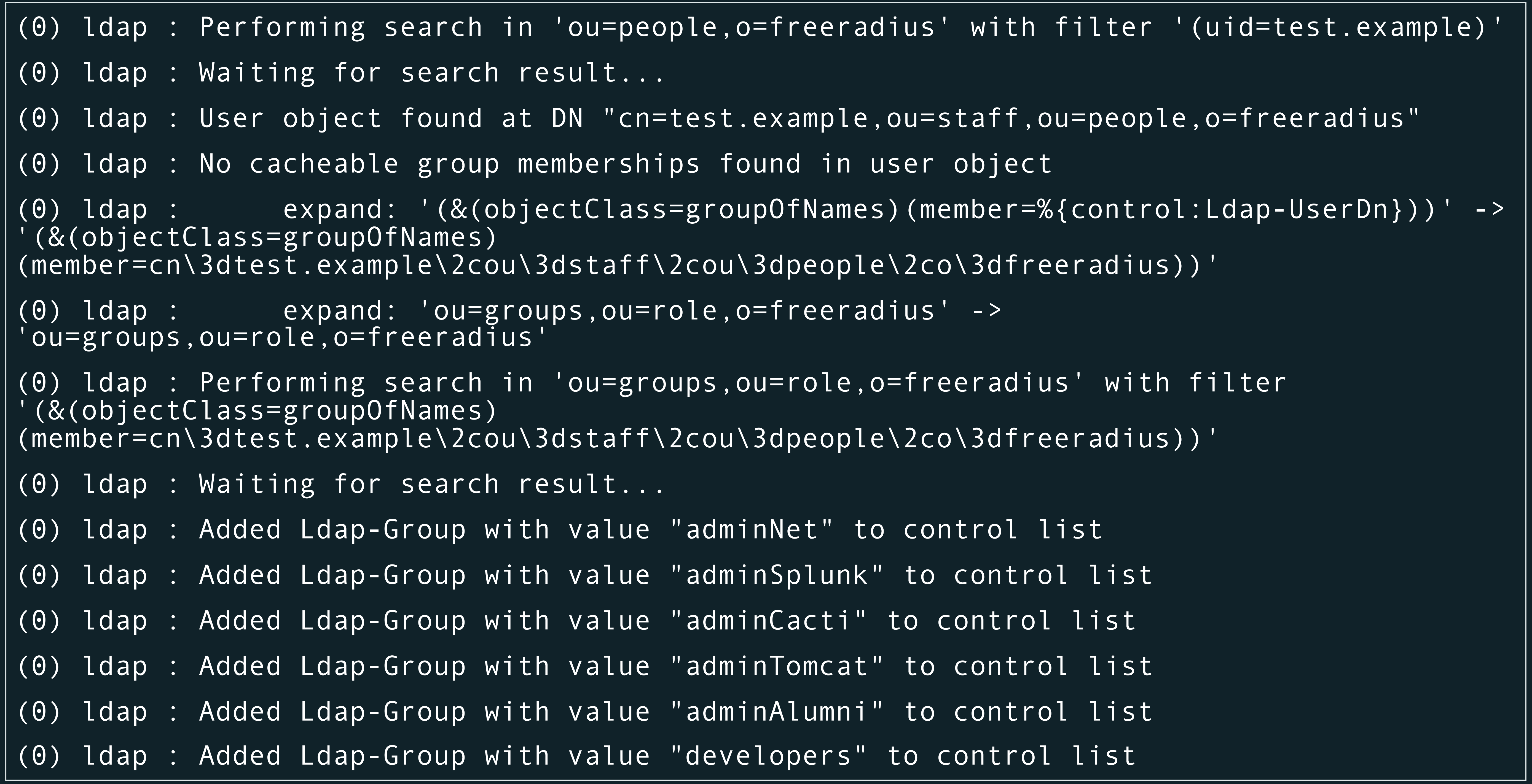
1. RADSEC (RADIUS over TLS) + DTLS.
2. Password change support (MSCHAPv2).
3. Connection pools and connection pool sharing.
4. Proper regular expressions (libpcre) + Pre-Compilation + named captures groups + 32 numbered capture groups for all regex flavours.
5. Tests, tests and more tests (Policy language, modules etc...).
6. COLOURS (Colourised log output for errors/warnings/info).
7. IP address comparisons e.g. "<cidr>127.0.0.1 < 12.0.0.0/24".
8. 64Bit Integers and IP prefix types.
9. SNMP Traps
10. Multivalued comparisons e.g. if (&Groups[*] == &User-Name)



```
shinyhead:freeradius-server-fork arr2036$ make test
UNIT-TEST base.dict
UNIT-TEST rfc.txt
UNIT-TEST errors.txt
UNIT-TEST extended.txt
UNIT-TEST lucent.txt
UNIT-TEST wimax.txt
UNIT-TEST escape.txt
UNIT-TEST condition.txt
UNIT-TEST xlat.txt
UNIT-TEST vendor.txt
UNIT-TEST dhcp.txt
...
EAPOL_TEST gtc
EAPOL_TEST leap
EAPOL_TEST md5
EAPOL_TEST mschapv2
EAPOL_TEST peap-client-mschapv2
EAPOL_TEST peap-eap-gtc
EAPOL_TEST peap-mschapv2
EAPOL_TEST pwd
EAPOL_TEST tls
EAPOL_TEST ttls-chap
EAPOL_TEST ttls-client-eap-mschapv2
EAPOL_TEST ttls-client-eap-tls
EAPOL_TEST ttls-eap-gtc
EAPOL_TEST ttls-eap-mschapv2
EAPOL_TEST ttls-mschapv2
EAPOL_TEST ttls-pap
```

WHATS NEW V3.0.X CONT…

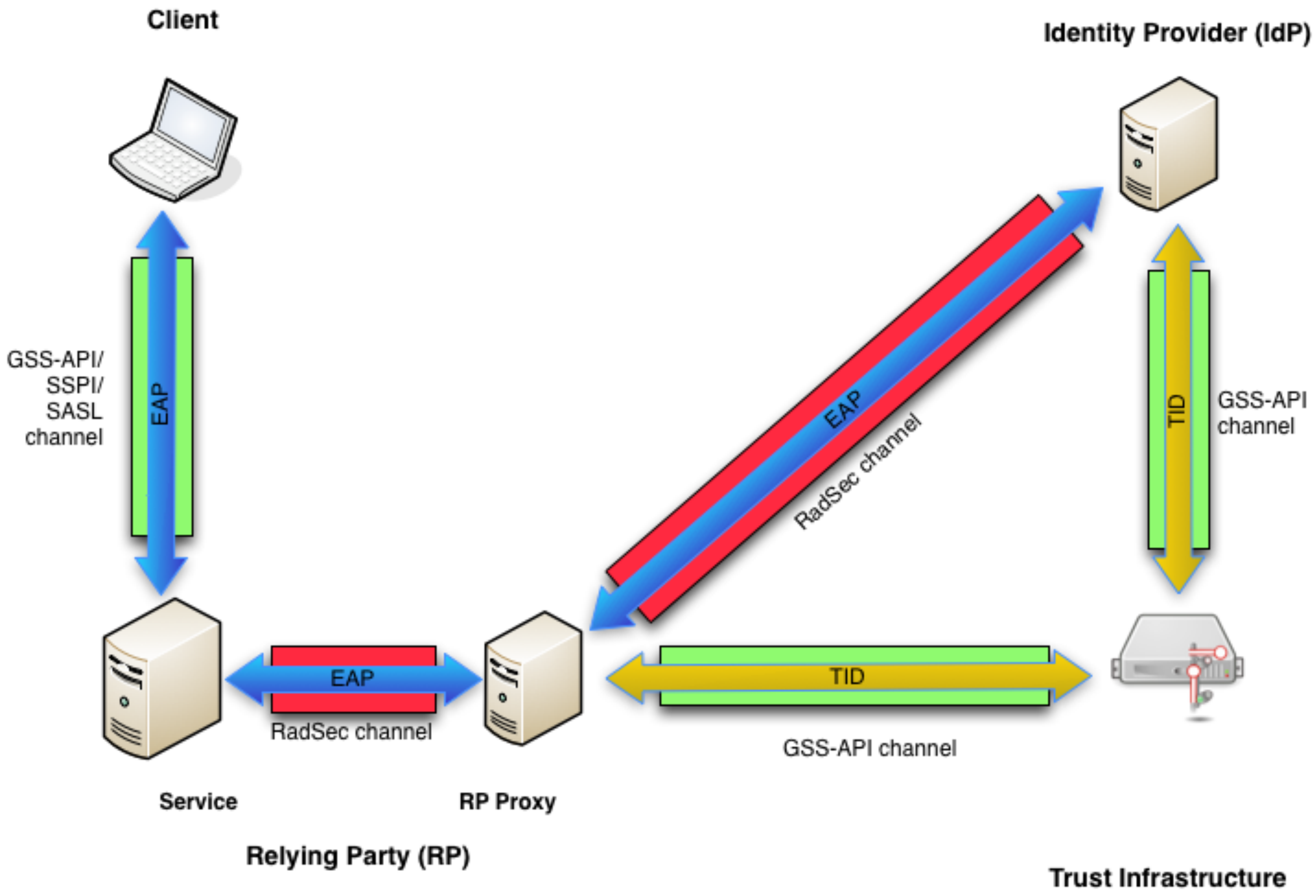
11. LDAP Group caching.
12. LDAP Dynamic clients.
13. REST API client.
14. Arbitrary client attributes.
15. SHA2 support.
16. Significantly improved detail reader performance (200%).
17. Shared cache support (currently memcached, Redis).
18. Startup checks for xlat and unlang syntax.
19. Pre-compilation of regular expressions
20. Functional user specific debugging



```
(0) ldap : Performing search in 'ou=people,o=freeradius' with filter '(uid=test.example)'
(0) ldap : Waiting for search result...
(0) ldap : User object found at DN "cn=test.example,ou=staff,ou=people,o=freeradius"
(0) ldap : No cacheable group memberships found in user object
(0) ldap :      expand: '(&(objectClass=groupOfNames)(member=%{control:Ldap-UserDn}))' ->
'(&(objectClass=groupOfNames)
(member=cn\3dtest.example\2cou\3dstaff\2cou\3dpeople\2co\3dfreeradius))'
(0) ldap :      expand: 'ou=groups,ou=role,o=freeradius' ->
'ou=groups,ou=role,o=freeradius'
(0) ldap : Performing search in 'ou=groups,ou=role,o=freeradius' with filter
'(&(objectClass=groupOfNames)
(member=cn\3dtest.example\2cou\3dstaff\2cou\3dpeople\2co\3dfreeradius))'
(0) ldap : Waiting for search result...
(0) ldap : Added Ldap-Group with value "adminNet" to control list
(0) ldap : Added Ldap-Group with value "adminSplunk" to control list
(0) ldap : Added Ldap-Group with value "adminCacti" to control list
(0) ldap : Added Ldap-Group with value "adminTomcat" to control list
(0) ldap : Added Ldap-Group with value "adminAlumni" to control list
(0) ldap : Added Ldap-Group with value "developers" to control list
```

NEW IN V3.0.X - TRUST ROUTER

- Next generation trust/ introduction service developed as part of the Moonshot project (now the Assent service).
- Allows COIs (Communities Of Interest) to operate across multiple federations.
- X509v3 could technically achieve the same trust relationships. But admin would be extremely difficult.
- Trust router implements two services
  - Trust router protocol - Distributes information about available IdPs (Identity providers) and the realms they serve to members of a COI.
  - Trust path query/Temporary ID protocol. Allows service provider (SP) to retrieve a TID (Temporary ID) for communicating with IdP.
- TIDs allow for lower latency, higher reliability, communication between SPs and IdPs.



https://wiki.moonshot.ja.net/display/Moonshot/The+Architecture+and+Protocol+Flows+of+Moonshot

NEW IN V3.0.X - TRUST ROUTER (THE FREERADIUS BIT)

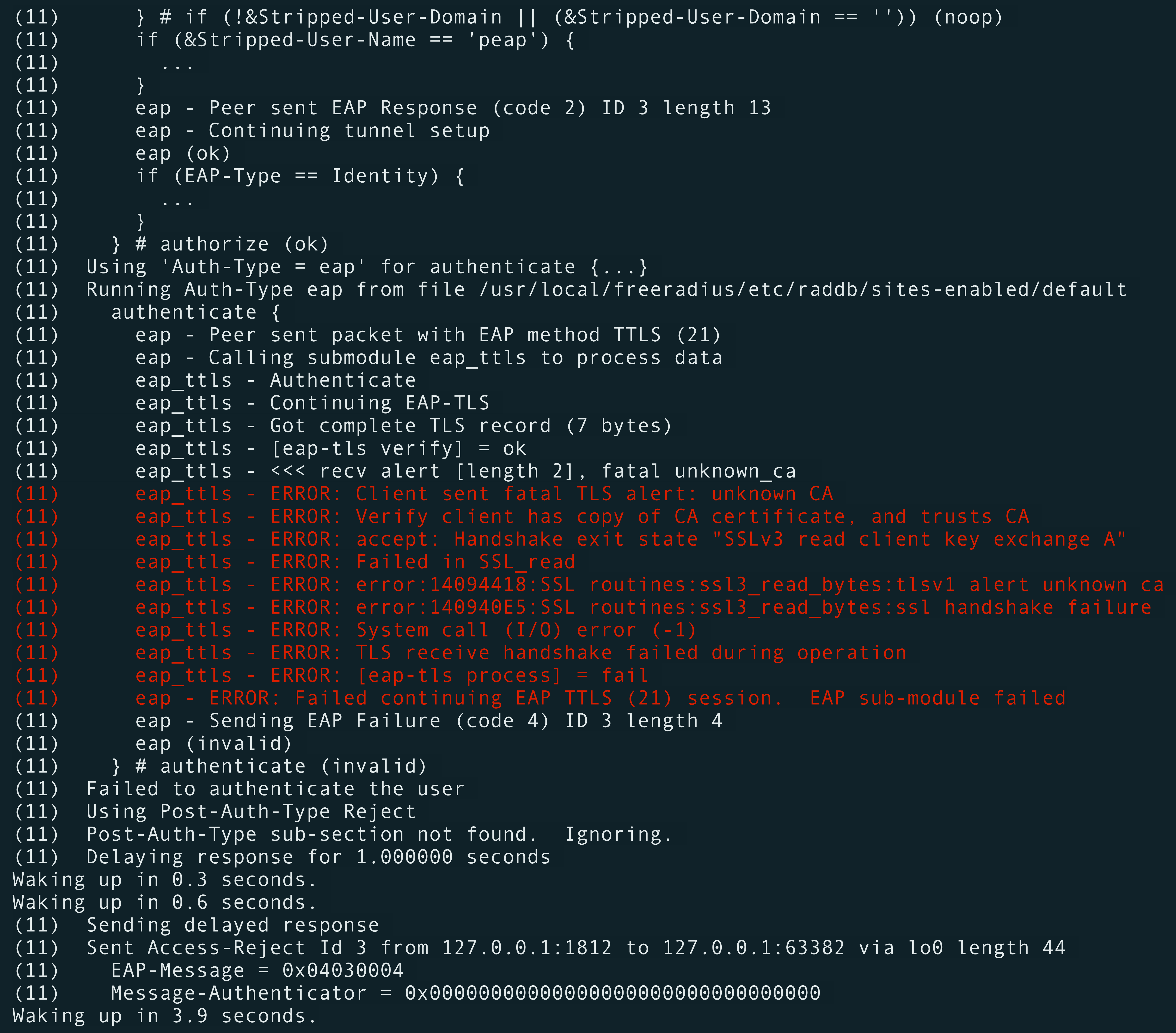
- SP - On call to rlm_realm, on discovering an unknown realm (or realm which requires update).
  - Queries local TR for IdP information and TID, providing DH Params (first half of DH exchange).
  - If positive response - retrieves list of IdP servers, the DH exchange completed for each IdP (second half of DH exchange). Computes symmetric keys for each home server.
  - Creates/Inserts new realm entry with realm->pool->home_server structures.
  - Establishes outbound TCP connection to IdP.
  - Performs TLS-PSK handshake, with the key identifier (unique ID for the temporary credentials), and computed PSK.
- IdP - On request from INADDR_ANY client
  - Uses key identity from TLS-PSK to query TR to get previously generated PSK.

WHY IS FREERADIUS A GOOD CANDIDATE FOR TRUST ROUTER INTEGRATION?

- FreeRADIUS already had support for RADSEC.
- Fairly minor modifications needed.
- Pluggable architecture, easy to accommodate changes or enhancements to Trust Router.
- Dynamic debugging (watch live authentications).
- Scalable - 30,000 PPS when proxying on modest hardware.

WHATS NEW V3.2.X

1. Redis 3.0.x cluster support.
2. High performance > (10,000 alloc/s per cluster node) Redis IPv4/IPv6 allocation.
3. Significantly improved EAP debugging + Improved EAP performance
4. Certificateless EAP-TLS (like HTTPS).
5. DHCPv4 'just works' (improved auto-discovery of interface configuration).



```
(11)      } # if (!&Stripped-User-Domain || (&Stripped-User-Domain == '')) (noop)
(11)      if (&Stripped-User-Name == 'peap') {
(11)        ...
(11)      }
(11)      eap - Peer sent EAP Response (code 2) ID 3 length 13
(11)      eap - Continuing tunnel setup
(11)      eap (ok)
(11)      if (EAP-Type == Identity) {
(11)        ...
(11)      }
(11)    } # authorize (ok)
(11)  Using 'Auth-Type = eap' for authenticate {...}
(11)  Running Auth-Type eap from file /usr/local/freeradius/etc/raddb/sites-enabled/default
(11)    authenticate {
(11)      eap - Peer sent packet with EAP method TTLS (21)
(11)      eap - Calling submodule eap_ttls to process data
(11)      eap_ttls - Authenticate
(11)      eap_ttls - Continuing EAP-TLS
(11)      eap_ttls - Got complete TLS record (7 bytes)
(11)      eap_ttls - [eap-tls verify] = ok
(11)      eap_ttls - <<< recv alert [length 2], fatal unknown_ca
(11)      eap_ttls - ERROR: Client sent fatal TLS alert: unknown CA
(11)      eap_ttls - ERROR: Verify client has copy of CA certificate, and trusts CA
(11)      eap_ttls - ERROR: accept: Handshake exit state "SSLv3 read client key exchange A"
(11)      eap_ttls - ERROR: Failed in SSL_read
(11)      eap_ttls - ERROR: error:14094418:SSL routines:ssl3_read_bytes:tlsv1 alert unknown ca
(11)      eap_ttls - ERROR: error:140940E5:SSL routines:ssl3_read_bytes:ssl handshake failure
(11)      eap_ttls - ERROR: System call (I/O) error (-1)
(11)      eap_ttls - ERROR: TLS receive handshake failed during operation
(11)      eap_ttls - ERROR: [eap-tls process] = fail
(11)      eap - ERROR: Failed continuing EAP TTLS (21) session.  EAP sub-module failed
(11)      eap - Sending EAP Failure (code 4) ID 3 length 4
(11)      eap (invalid)
(11)    } # authenticate (invalid)
(11)  Failed to authenticate the user
(11)  Using Post-Auth-Type Reject
(11)  Post-Auth-Type sub-section not found.  Ignoring.
(11)  Delaying response for 1.000000 seconds
Waking up in 0.3 seconds.
Waking up in 0.6 seconds.
(11)  Sending delayed response
(11)  Sent Access-Reject Id 3 from 127.0.0.1:1812 to 127.0.0.1:63382 via lo0 length 44
(11)    EAP-Message = 0x04030004
(11)    Message-Authenticator = 0x00000000000000000000000000000000
Waking up in 3.9 seconds.
```

WHATS NEW V3.2.X CONT…

6. Infinitely nested TLV support.

7. SNMP support (will also work over proxy chains). Translates OIDs into TLV structures in FR extended dictionary space. Calls client using Net-SNMP pass persist.

8. libwbclient support (30% performance improvement over ntlm_auth for Active Directory authentication).

9. map { } syntax to support retrieving multiple values per query from LDAP and SQL.

10. JSON parsing and simple query syntax

```
radsnmp (debug): read: get
radsnmp (debug): read: .1.3.6.1.2.1.67.1.1.1.1.1.0
Sent Status-Server Id 3 from (null):0 to 127.0.0.1:1812 length 78
  Radius-Auth-Serv-Ident = "\000"
  FreeRADIUS-SNMP-Operation = get
  Message-Authenticator = 0x00
Received Access-Accept Id 3 from 127.0.0.1:1812 to 127.0.0.1:49548 via (null) length 75
  Radius-Auth-Serv-Ident = "FreeRADIUS 3.1.0"
  FreeRADIUS-SNMP-Type = string
radsnmp (debug): said: 1.3.6.1.2.1.67.1.1.1.1.1
radsnmp (debug): said: string
radsnmp (debug): said: FreeRADIUS 3.1.0
radsnmp (debug): Returned 1 varbind responses
```

```
map sql "SELECT group, reply_message FROM users WHERE user = '%{User-Name}'" {
   &control:Group := 'group'
   &reply:Reply-Message := 'reply_message'
}
```

```
map json "%{rest:http://www.example.org/user/%{User-Name}" {
   &control:Group := 'user.group'
   &reply:Reply-Message := 'user.message'
}
```

NEW IN V3.2.0 - DISTRIBUTED SESSION RESUMPTION

- Server side session resumption part of the TLS standard (not an extension). Not RFC 5077.
- Hashes MSK (Master session key) from previous session with random data from client + server to generate new MSK.
- Hashing cheap compared to performing full RSA calculations to generate keying material for MSK.
- Real wall clock saving comes from not having to exchange big certificate chains.
- With PEAP/TTLS, resuming a session allows you to skip the inner method (no hits on AD or LDAP).

Client hello

```
struct {
    ProtocolVersion client_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suites;
    CompressionMethod compression_methods;
    union extensions_present;
} ClientHello;
```

Server hello

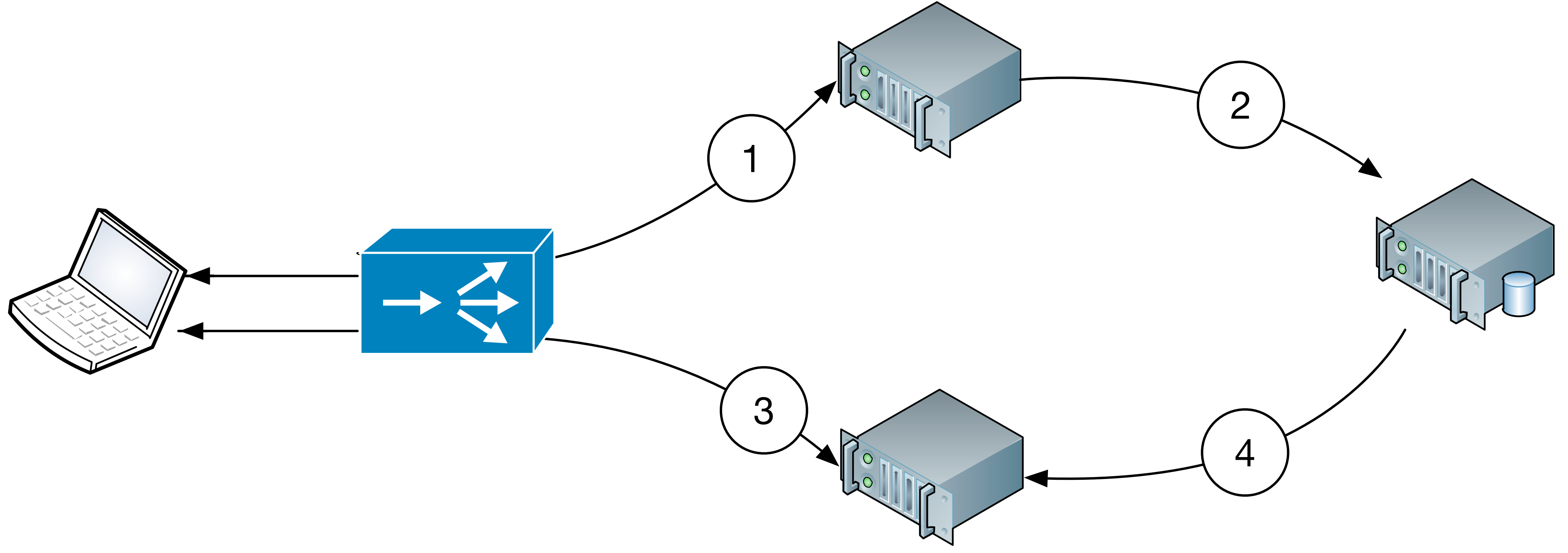
```
struct {
    ProtocolVersion server_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suite;
    CompressionMethod compression_method;
     union extensions_present;
} ServerHello;
```

1

```
Client                       Server

ClientHello         >>>

                             ServerHello
                             Certificate
                             ServerKeyExchange
                             CertificateRequest
                    <<<      ServerHelloDone

Certificate
ClientKeyExchange
CertificateVerify
[ChangeCipherSpec]
Finished            >>>

                             [ChangeCipherSpec]
                    <<<      Finished

Application Data    <->      Application Data
```

2

```
Session ID
<opaque session blob>
&session-state;[*]
```



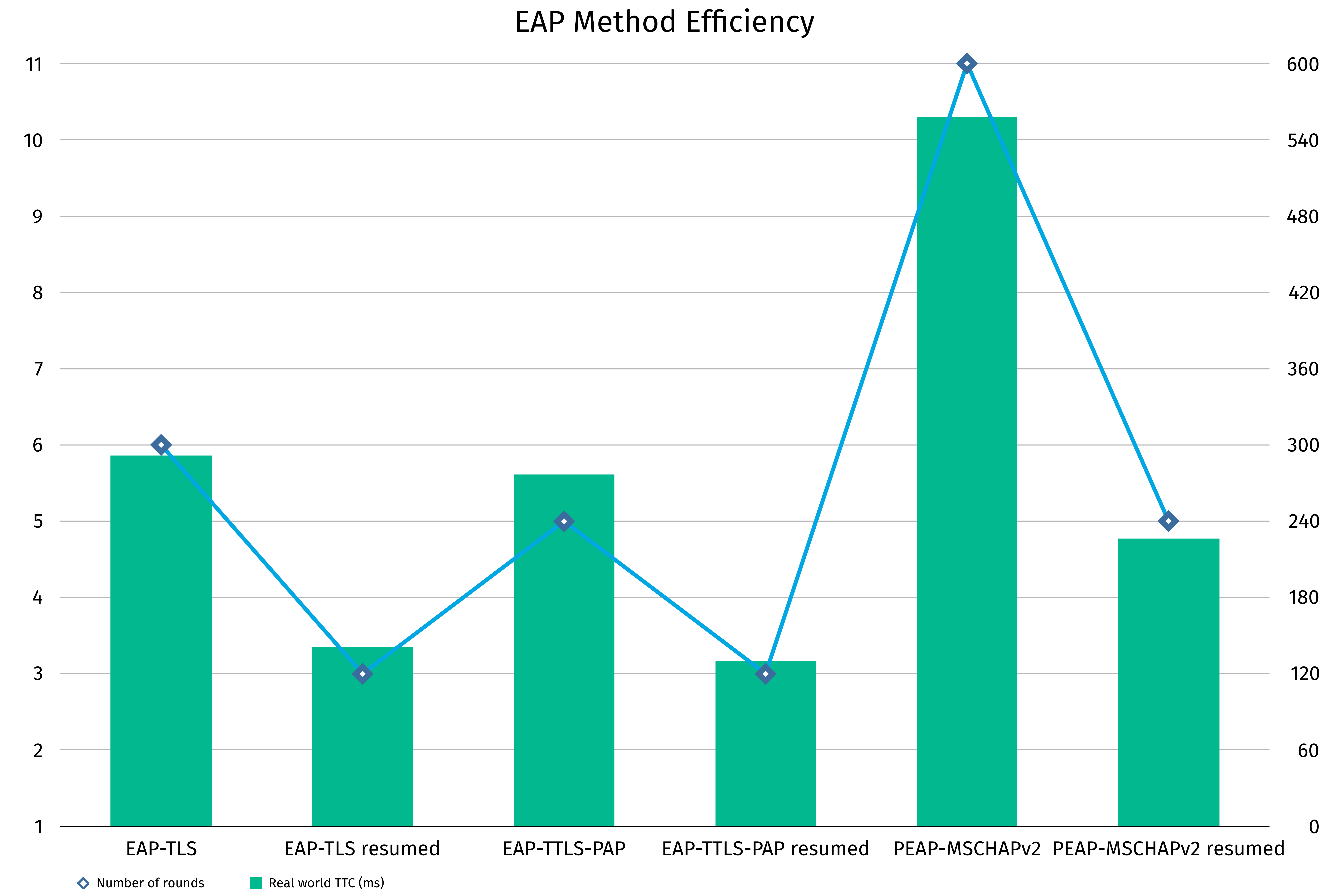
3

```
Client                          Server

ClientHello            >>>
                                ServerHello
                                [ChangeCipherSpec]

                       <<<      Finished

[ChangeCipherSpec]
Finished               >>>

Application Data       <->      Application Data
```

4

```
Session ID
<opaque session blob>
&session-state;[*]
```

ENHANCEMENTS IN FREERADIUS 3.0.0-3.2.0



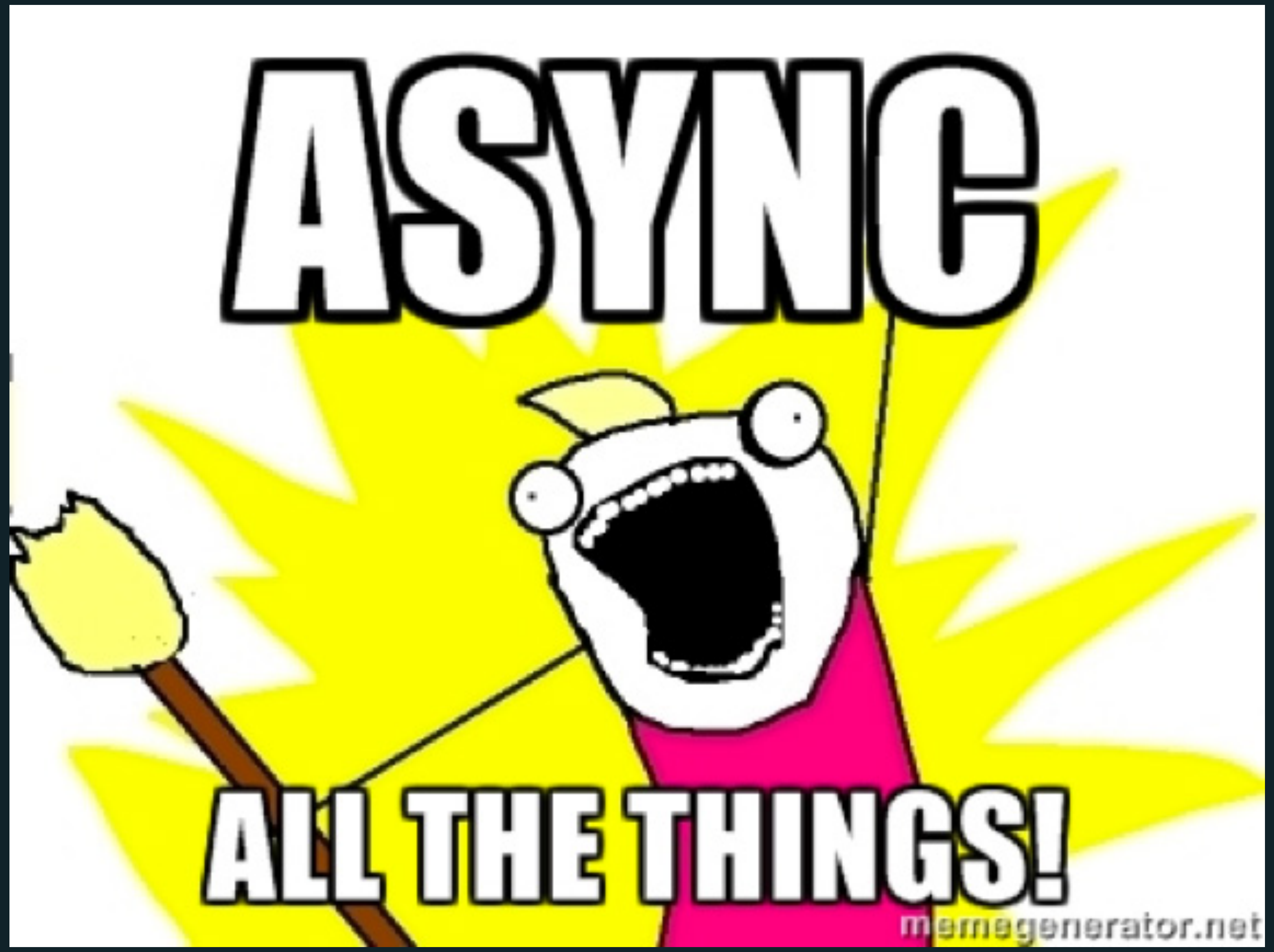
- 2 fragments per client/ server certificate chain.
- Real world Time To Completion (TTC) assumes 50ms credential lookup (where relevant) + 40ms Round Trip Time (RTT).
- Session blobs 138b EAP-TTLS-PAP
- Session blobs 1863b EAP-TLS.

WASN'T THIS AVAILABLE BEFORE?

- Yes we've supported this for a while...
- The innovation is exporting the opaque session blob as an attribute.
- Couldn't be done in v2.x.x because max value length was 253 bytes.
- Couldn't be done usefully (between multiple servers) in v3.0.x because no shared cache module.
- Finally in v3.2.0 we have all the components needed to support this properly, including a new &session-state: list to simplify re-authorization.

FUTURE LOOKING STATEMENTS

- Should become a generic platform for implementing network protocols, with re-useable and flexible policy logic.
- Move to asynchronous I/O.
  - Process has begun - Iterative (as opposed to recursive) interpreter for unlang introduced in v3.1.x
- Diversified protocol support
  - New internal 'proto' API introduced in v3.1.x, should allow for:
    - DHCPv6.
    - Diameter.
    - maybe ANQP (when someone creates a standard for transporting it over IP).
- SQL statement caching.
- Better interpreted language support.



?