LURK Protocol for TLS/DTLS1.2

draft-mglt-lurk-tls

D. Migault

16/07/2016- IETF96- Berlin
LURK/TLS Scope

Scope of LURK/TLS:

- Query / response between Edge Server and Key Server
- TLS authentication methods are limited to:
  - RSA
  - ECDHE_RSA
  - ECDHE_DSA
- TLS version 1.2
- Binary protocol
General Design

<table>
<thead>
<tr>
<th>query header</th>
<th>query payload</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>response header</th>
<th>response payload</th>
</tr>
</thead>
</table>
ECDHE

TLS Client           Edge Server           Key Server

ClientHello
  ProtocolVersion server_version
  Random client_random
  Cipher_suite
    TLS_ECDHE_ECDSA_*, TLS_ECDHE_RSA_*, ...
  Extension Supported EC, Supported Point Format

---->

LURKTLS Header (Query)
LURKTLSECDHEInputPayload

1. Generating the signature

---->

LURKTLS Header (Response)
LURKTLSDigitallySignedPayloads
signature

<-------

ServerHello
  ProtocolVersion edge_server_version
  Random server_random
  Cipher_suite=TLS_ECDHE_ECDSA
  Extension Supported EC, Supported Point Format

Certificate
  ECDSA Public Key

ServerKeyExchange
  ecdhe_params
  signature

ServerHelloDone
<-------
ECDHE Security - signing oracle

Exposing the Private Key to first chosen bytes signing attacks:

- TLS exposes to the first 32 byte signing attack by TLS Clients
- LURK 64 byte signing attack by Edge Servers
  - Unlike TLS Clients, Edge Servers are authenticated
  - Unpredictable ECDHE reduces the first chosen to 32
ECDHE Security - cross protocol attacks

The signature is bound to the ClientHello.random, ServerHello.random but not:

- the TLS Version
- the authentication method

As a result, a given signature can be used across different TLS Versions, authentication methods, and signed parameters

- TLS1.3 solves this issue with context

Assumption:

- Key Server provides ECDHE_RSA, ECDHE_ECDSA and RSA
- The Private Key is only hosted on the Key Server
- TLS Version is TLS1.0, TLS1.1, TLS1.2
ECDHE Security - cross protocol attacks

TLS Version
- Authentication methods are identical across version
  - There is little chance to have a vulnerability associated to the TLS Version

Signature Scheme across authentication methods
- RSA signature is used in DHE_RSA and ECDHE_RSA
- ECDSA signature are only used for ECDHE_ECDSA
  - Collision restricted to DHE_RSA and ECDHE_RSA

Signed parameters
- DHE / ECDHE have probability of collisions
  - Rejecting parameters matching other authentication method’s parameters
Security

Authentication credentials generated by the Key Server SHOULD:

- Be restrained to a single TLS session
- Not leak information of the Private Key
- Involve a reasonable amount of resource (cpu / bandwidth)

We believe RSA, ECHDE:

- Exchanges meet this requirements
- Does not (significantly) increase weakness presented by TLS1.2
  - Is it acceptable to consider raising an unauthenticated TLS Client 32 first byte signing attack to an authenticated Edge Server 64 first byte attack?
Security

By centralizing the authentication operations to the Key Server:

- LURK presents a bottleneck architecture
  - Resource provisioning MUST consider this

However, LURK presents significant security advantages:

- Private Key is not shared anymore between running Edge Servers, Edge Servers images, Content Owner, CDNs, ...
- Compromise Edge Server does not leak the Private Key
  - It may only perform authorized cryptographic operations
- Centralizing Private Key operation eases monitoring and detection of malicious behaviors.
Thank you for your attention