Lightboard Lesson: Perfect Forward Secrecy Inspection Visibility

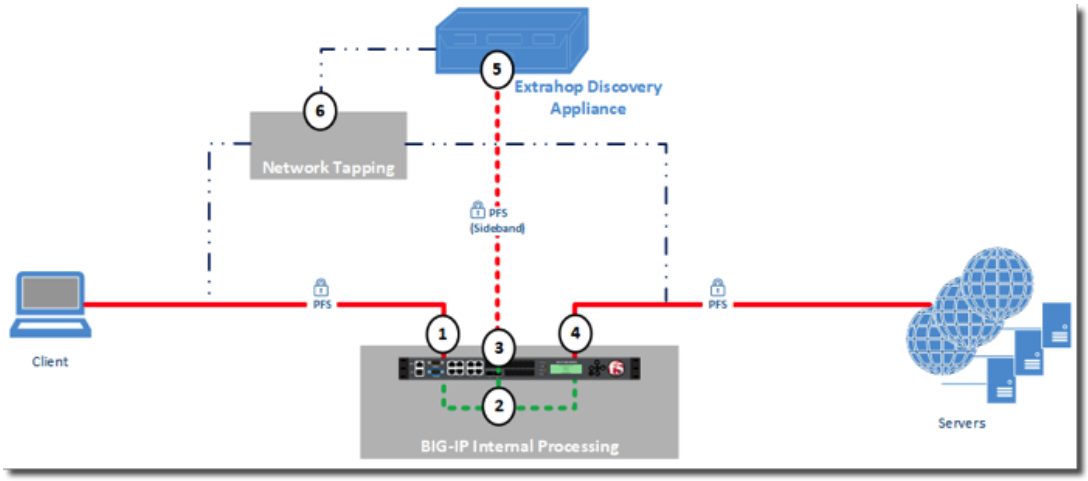
Jason Rahm, 2018-14-11

This time last year, we released a Lightboard Lesson recorded by David Holmes featuring his insights on the ultimate passive inspection architecture. Whereas the solution he proposed in that video stepped perfect forward secrecy (pfs) traffic down to RSA just for the inspection zone before stepping back up to pfs outbound to the server, the solution I'll cover today doesn't require passing the traffic payload to the inspection zone at all, and further, nothing has to leave the BIG-IP outside of forward secrecy protection.

Solution Overview

Good friend, former colleague, and community stalwart Colin Walker, reached out to share this solution for whom he credits his Extrahop colleagues Stephen DeSanto and Kanen Clement. The Lightboard Lesson version of the solution is below, followed by the detailed solution notes.

The Extrahop Discovery Appliance (EDA) has a feature that allows administrators to receive the session keys from a pfs session and together with the server private key, decrypt each session's traffic. This solution utilizes the functionality in iRules to collect those session keys and deliver them to the appliance via a sideband connection.



1. A client connects to the BIG-IP via TLS with a PFS cipher. The external virtual server has a clientssl profile to decrypt traffic for inspection.
2. While traffic is inspected, an iRule on the external virtual server collects session secret keys and then makes a sideband connection to an internal virtual server with the EDA as a pool member. This virtual server has a serverssl

profile to ensure the traffic between BIG-IP and the EDA is protected since session keys are going to be passed.

3. The internal virtual server has an iRule applied that manages the protocol requirements between BIG-IP and the EDA. Oneconnect is utilized and session keys will be sent on existing connections if available to reduce the connection setup overhead with the EDA.
4. The external virtual server also has a serverssl profile to re-encrypt the traffic back to the server farm.
5. The EDA receives the keys from BIG-IP, and together with the configured server private keys, is available to decrypt traffic.
6. Payload is not sent from BIG-IP to the EDA, this is received either from port mirroring configurations or a network tap infrastructure as shown in the image.

BIG-IP Configuration

The BIG-IP configuration objects required for this solution include:

- iRule for session key copying and sideband communication
- iRule for protocol management
- Cipher Rule/Group for the shared secret receiver
- Serverssl profile for the sideband connection to the EDA
- Node/Pool/Virtual Server for the EDA

iRule #1 - Session Secrets & Sideband Connection Initiation

```
when RULE_INIT {
  # Here, you must define the name of the sideband virtual server to send secrets
  set static::virtual_server "extrahop_shared_secret_sideband"
}
when CLIENTSSL_HANDSHAKE {
  if { [catch {call sendSecret [SSL::clientrandom] [SSL::sessionsecret]} ] } {
    log local0. "ExtraHop Sideband: sideband vip unavailable"
    return
  }
}
when SERVERSSL_HANDSHAKE {
  if { [catch {call sendSecret [SSL::clientrandom] [SSL::sessionsecret]} ] } {
    log local0. "ExtraHop Sideband: sideband vip unavailable"
    return
  }
}
proc sendSecret {client_rand secret} {
  set client_rand [SSL::clientrandom]
  set secret [SSL::sessionsecret]
  set cmp_unit [TMM::cmp_unit]
  set session_secret [binary format H* $client_rand$secret]
  set length [string length $session_secret]
  if { $length != 80 }{
    return
  }
  # key for session table
  set key "${cmp_unit}_conn_${static::virtual_server}"
  # conn = session table data for $key for this dest_addr
  set conn [session lookup dest_addr $key]
  if { $conn eq "" }{
    set conn [connect -timeout 1000 -idle 300 -status conn_status $static::virtual_server]
    if { $conn_status ne "connected" }{
      return
    }
    session add dest_addr $key "$conn" 300
  } else {
```

```
    } else {
        # Attempt sideband connection re-use
        set conn_info [connect info -status $conn]
        set conn_state [lindex [lindex $conn_info 0] 0]
        if { $conn_state ne "connected" }{
          set conn [connect -timeout 1000 -idle 300 -status conn_status $static::virtual_server]
          if { $conn_status ne "connected" }{
            return
          }
          session add dest_addr $key "$conn" 300
        }
    }
    # Send secret message
    set secret_message [binary format H* "f35000$client_rand$secret"]
    set send_bytes [send -timeout 1000 -status send_status $conn $secret_message]
    recv -timeout 1 $conn
}
```

iRule #2 - Sideband Connection Management to EDA

```
when CLIENT_ACCEPTED {
  TCP::collect 80
}
when SERVERSSL_HANDSHAKE {
  # Send an initial hello
  SSL::respond [binary format H* "f00800dec0de0100000000"]
}
when CLIENT_DATA {
  set msg_header [TCP::payload 3]
  binary scan $msg_header H* header_hex
  if { $header_hex eq "" }{
    return
  }
  if { $header_hex eq "f00800" } {
    # Connection hello
    TCP::release
    TCP::notify request
  } elseif { $header_hex eq "f35000" } {
      # Shared secret message
       TCP::release
       TCP::notify request
  } else {
      TCP::release
      log local0. "ExtraHop Sideband: Unknown message type/header: $header_hex"
  }
  TCP::collect 80
}
when USER_REQUEST {
  # We don't expect a response, so let's just signal one
  # and detach to make oneconnect happy (message-based)
  TCP::notify response
  LB::detach
}
```

Cipher Rule/Group

```
ltm cipher rule extrahop_shared_secret_cipher_rule {
```

```
    cipher ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDH
}
ltm cipher group extrahop_shared_secret_cipher_group {
    allow {
        extrahop_shared_secret_cipher_rule { }
    }
}
```

Serverssl Profile

```
ltm profile server-ssl extrahop_shared_secret_ssl_profile {
    app-service none
    cipher-group extrahop_shared_secret_cipher_group
    ciphers none
    defaults-from serverssl
}
```

Node

```
ltm node node_extrahop {
    address 10.0.2.25
    monitor icmp
    session monitor-enabled
    state down
}
```

Pool

```
ltm pool pool_extrahop_secret_receiver {
    members {
        node_extrahop:4873 {
            address 10.0.2.25
            session monitor-enabled
            state checking
        }
    }
    monitor tcp
}
```

Virtual Server

```
ltm virtual extrahop_shared_secret_sideband {
    destination 10.0.0.1:4873
    ip-protocol tcp
    mask 255.255.255.255
    pool pool_extrahop_secret_receiver
    profiles {
        extrahop_shared_secret_ssl_profile {
            context serverside
        }
        oneconnect { }
        tcp { }
    }
    rules {
        extrahop_shared_secret_proto
```

```
    }
    source 0.0.0.0/0
    source-address-translation {
        type automap
    }
    translate-address enabled
    translate-port enabled
    vs-index 2
}
```

Note: iRule #1 (extrahop_shared_secret_export) should be applied to whatever client/server payload virtual server(s) requiring pfs decryption on the EDA.

Extrahop Configuration

The EDA configuration steps are available here on Extrahop's website (login required.) Thanks again to the Extrahop team for an excellent solution utilizing the power of BIG-IP to equip organizations to inspect forward secret traffic.

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc. Corporate Headquarters info@f5.com	F5 Networks Asia-Pacific apacinfo@f5.com	F5 Networks Ltd. Europe/Middle-East/Africa emeainfo@f5.com	F5 Networks Japan K.K. f5j-info@f5.com