SD-WAN Secure Communication Designs and Vulnerabilities

Denis Kolegov
@dnkolegov

DeepSec 2019

whoami

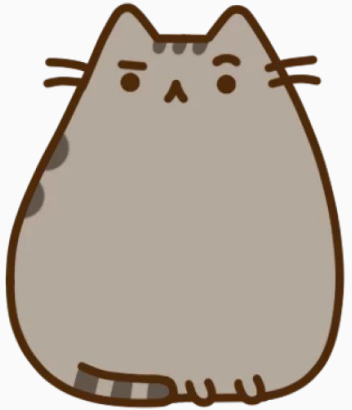
- Ph.d, associate professor at Tomsk State University
- Principal security researcher at BI.Zone LLC
- SD-WAN New Hope and AIsec team member
- https://twitter.com/dnkolegov

Disclaimers

- Please note, this is my personal talk
- I don't speak for my employers
- These thoughts, jokes and opinions are my own
- No SD-WAN were “harmed” in the making of this research
- Some SD-WAN vendors or product names are hidden

Agenda

- SD-WAN New Hop(e) Project
- SD-WAN Essence
- Vulnerabilities
- Secure Design Aspects
- Conclusions




SD WAN
NEW HOPE

SD-WAN New Hop(e) Project

- Citrix / Talari
- Versa
- SilverPeak
- RiverBed
- Fortinet
- Cisco / Viptela
- VMWare / Velocloud
- Viprinet
- Brain4Net

- Checklists
  - SD-WAN Security Assessment
- Tools
  - SD-WAN Harvester
  - SD-WAN Infiltrator
  - Grinder Framework
- Papers
  - SD-WAN Internet Census
  - SD-WAN Threat Landscape
  - SD-WAN Practical Assessment

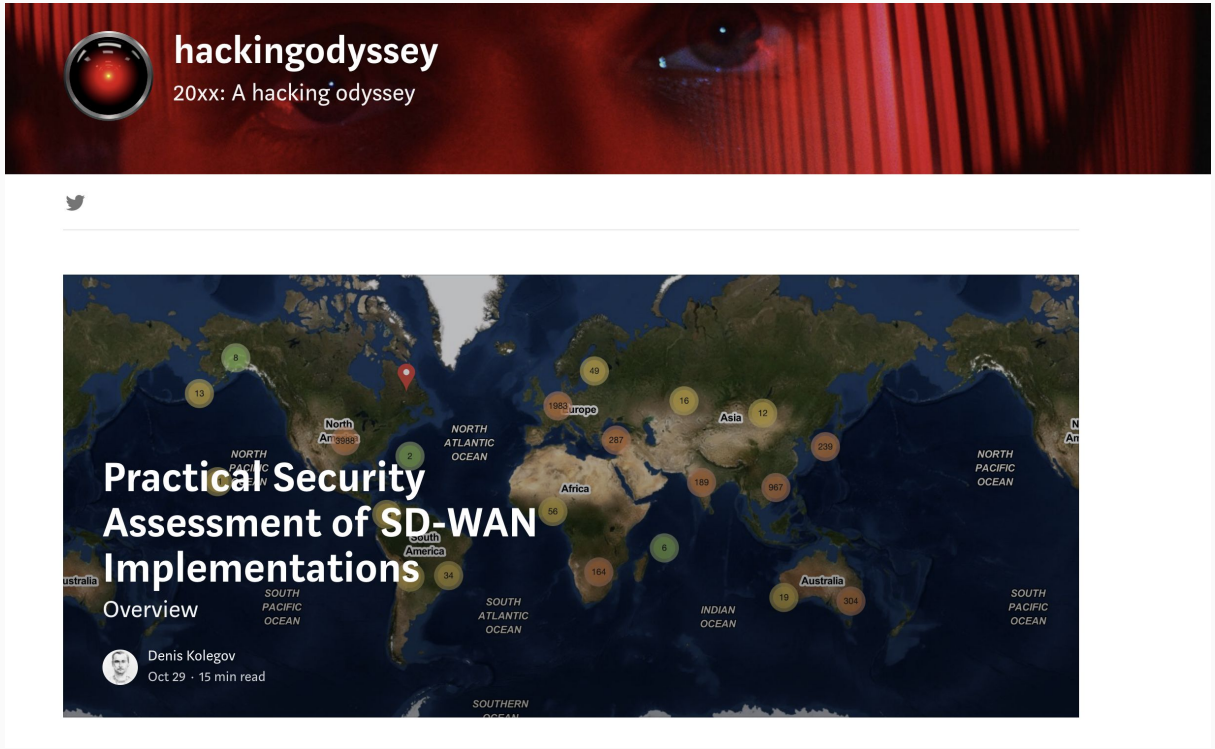
@sdnewhop

SD-WAN New Hop(e) Team

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

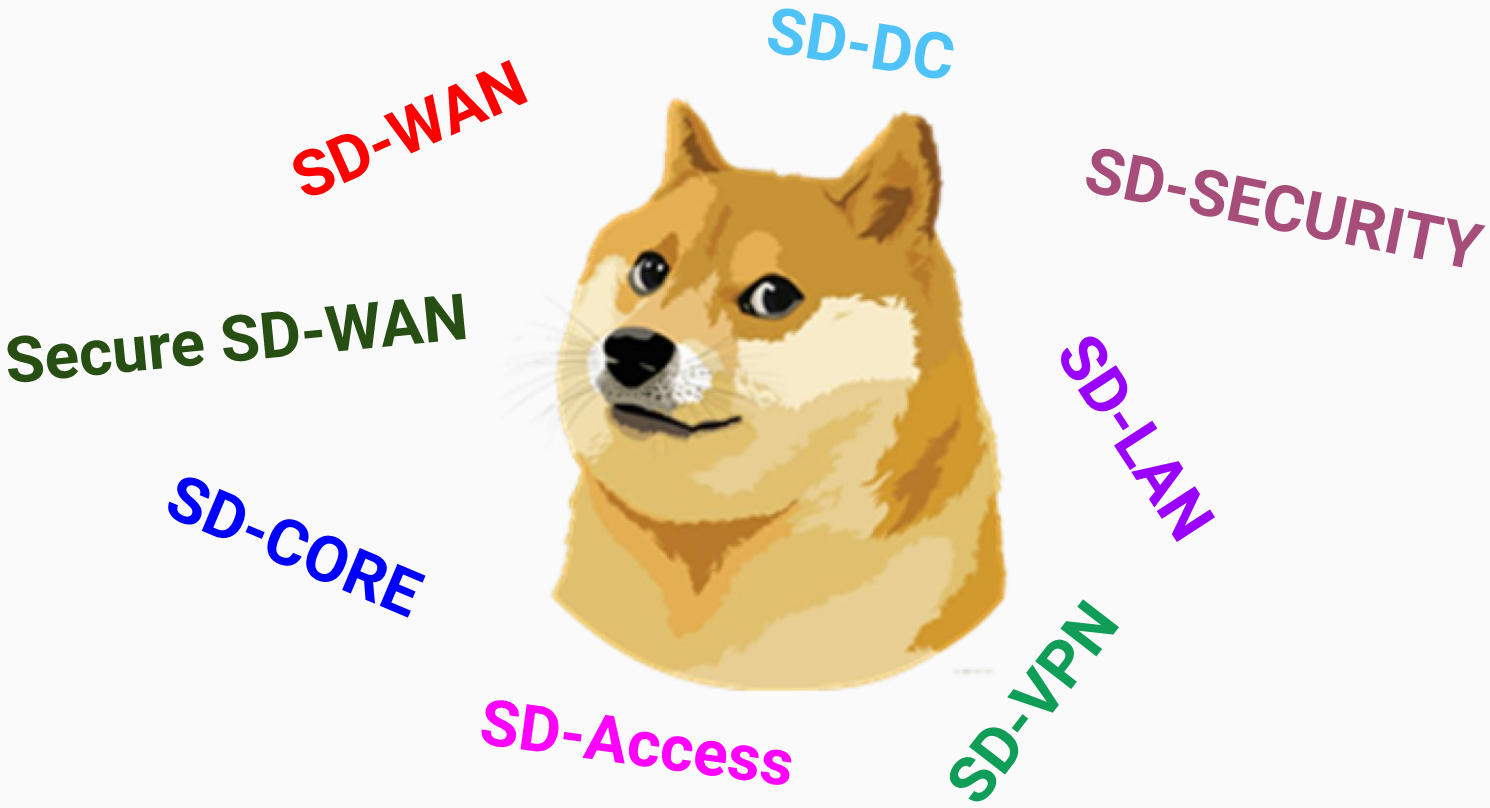
@sdnewhop

Practical Security Assessment of SD-WAN Implementations



https://bit.ly/2rD23kX

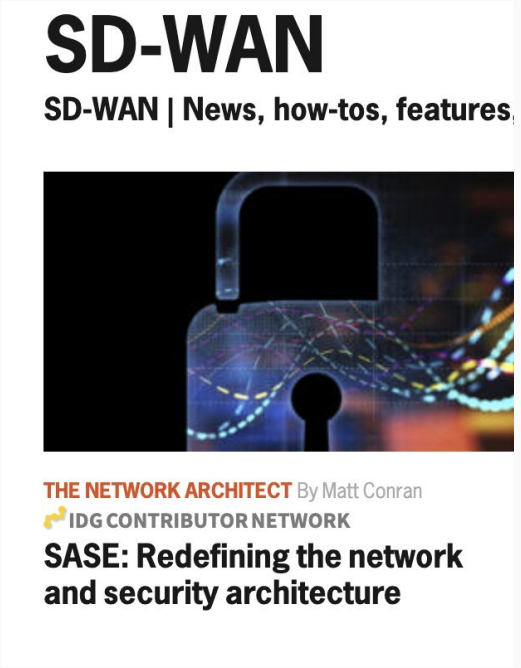
SD Everywhere



SD-WAN News

- Cisco forges tighter SD-WAN links to Microsoft Azure cloud, Office 365
- SD-WAN is evolving into Secure Access Service Edge
- Tight Wi-Fi integration is key to successful SD-Branch
- Performance-Based Routing (PBR) – The gold rush for SD-WAN

Source: https://www.networkworld.com/category/sd-wan/

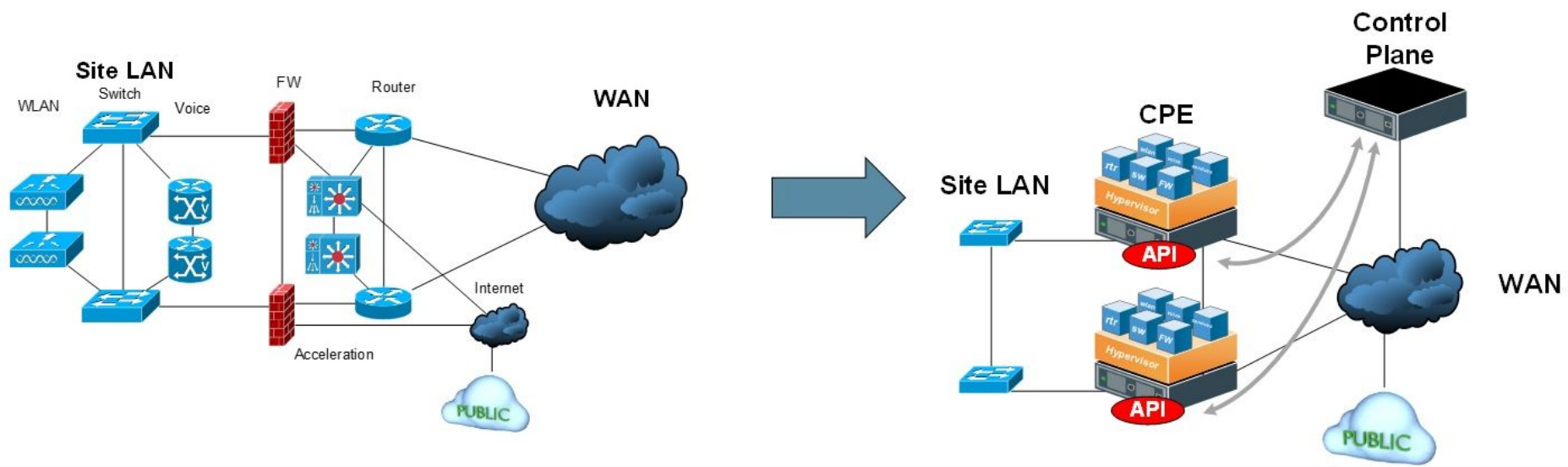


SD-WAN Essence

SDN-NFV/SD-WAN Vocabulary

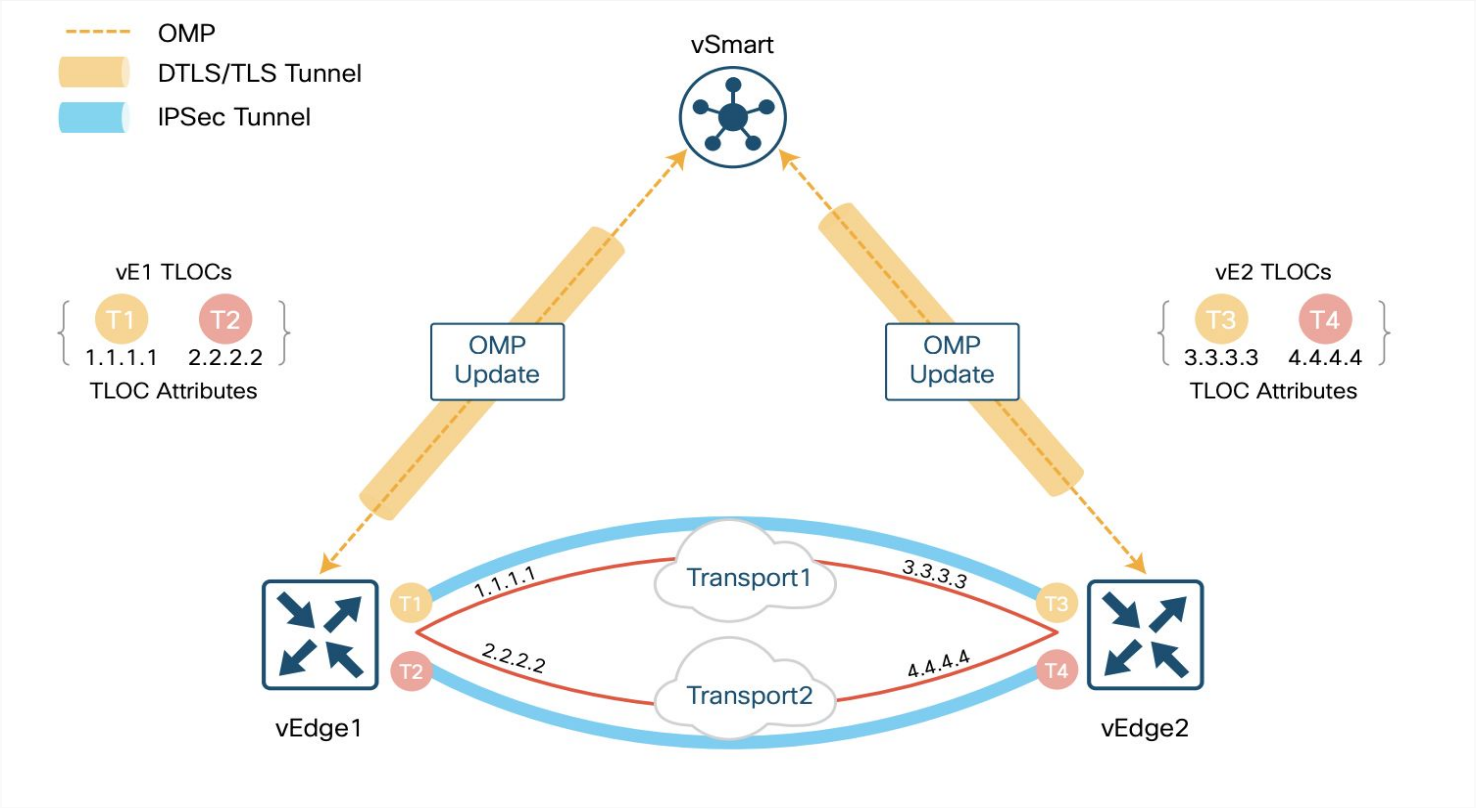
- SDN: principle of physical separation of control plane from data plane
- Network Function (NF): functional block within a network infrastructure that has well-defined external interfaces and functional behavior
- Network Functions Virtualization(NVF): principle of separating network functions from hardware
- Virtualized Network Function(VNF): implementation of an NF that can be deployed using NFVI: DPI, IDPS, WAF, VPN
- **SD-WAN is a specific application of SDN and NFV technologies to WAN connections**

Traditional WAN vs Software-defined WAN



Source: http://www.abusedbits.com/2017/01/modern-network-areas-in-software-defined.html

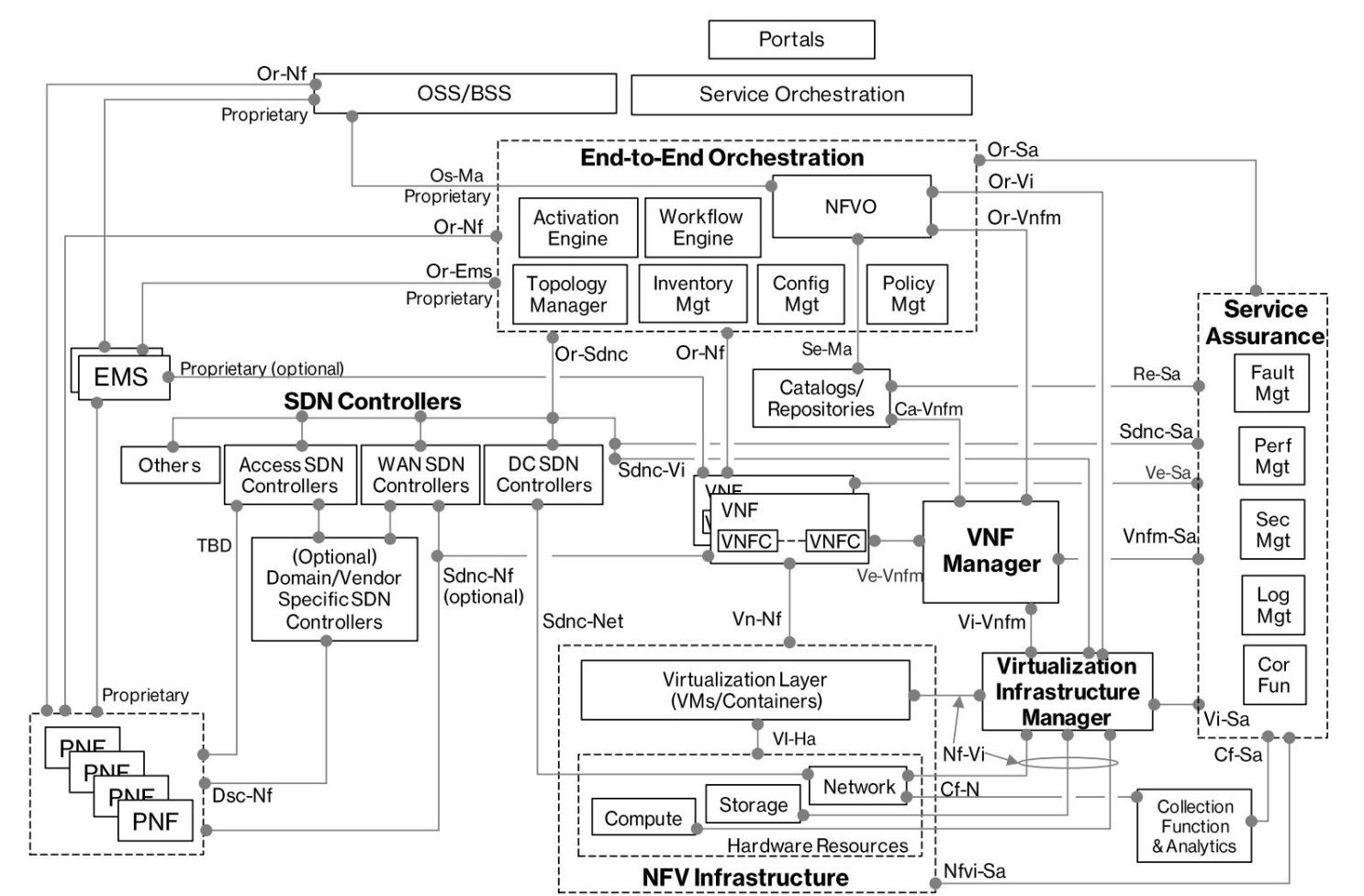
Cisco Viptela SD-WAN Design



Source: Cisco SD-WAN Design Guide

SDN vs SD-WAN

- While they share the same concept, they are two completely different usage environments
- SDN started out in datacenters (internal use), whereas SD-WAN is external use
- Different use, different requirements, especially for security
- This also has an impact on network security (underlay network and control plane)
- Networks are best protected at the lowest layer possible



Verizon SDN-NFV reference architecture

Vulnerabilities

Zero Touch Provisioning

Zero Touch Provisioning

- ZTP requires a known provisioning server
- If a management portal (UI) is cloud-based and vendor-controlled, it requires full trust to vendor
- Approaches
  - One-time tokens
  - Challenge-response protocols
  - Password-based authentication
  - Secret-based authentication (e.g., chassis serial numbers)
- Mutual authentication
  - An orchestrator authenticates an edge router
  - The edge router authenticates the orchestrator
- One of requirements is automated process for managing keys and certificates

Versa ZTP Bootstrapping with Hardcoded Password

```
(function () {
    'use strict';
     angular.module('[redacted].services')
    .service('BootstrapLoadConfigService', function ($window, $q, $http,  $rootScope, $cookieStore, $,  Base64Service, [redacted]) {

            var self = this;
            self.loadMergeConfig = loadMergeConfig;
            self.counter = 1;

            var authdata = Base64Service.encode('admin' + ':' + '[redacted]');

            function loadMergeConfig( params ) {
                var deferred = $q.defer();

                $http({
                    method: 'POST',
                    url: '/load[redacted]',
                    data: params,
                    headers: {
                            'Content-Type': 'application/[redacted]',
                            'Accept': 'application/[redacted]',
                            'Authorization': "Basic "+authdata,
                            'url':[redacted].apiHost+':'+[redacted].apiPort + '[redacted].apiConfig +
'/system:system/configuration/_operations/load-merge'
                        }
```

Arista ZTP

- ZTPServer provides a bootstrap environment for Arista EOS based products
- Sources
  - https://github.com/arista-eosplus/ztpserver
  - https://ztpserver.readthedocs.io/en/master/index.html
- It is recommended to use Apache (mod_wsgi)
  - When do you say Apache, do you mean Slow HTTP DoS attacks?

Arista Zero Security Provisioning

20.3 DHCP Service for Zero Touch Provisioning (ZTP) Setup

The ZTP process relies on a DHCP server to get devices registered with CVP. The DHCP server can be on the CVP, but is more commonly an external DHCP server.

Step 1 Ensure the DHCP server is installed (it is installed by default in CVP).

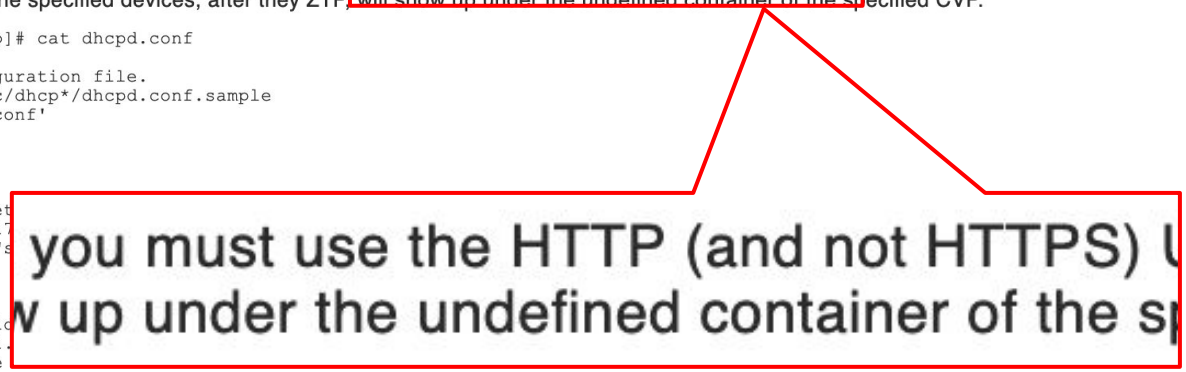
```
rpm -qa | grep dhcp
dhcp-common-4.1.1-43.P1.el6.x86_64
dhcp-4.1.1-43.P1.el6.x86_64
```

Step 2 Edit the **/etc/dhcp/dhcpd.conf** file to include the option **bootfile-name**, which provides the location of the script that starts the ZTP process between CVP and the device.

In this example, DHCP is serving the 172.31.0.0/16 subnet.

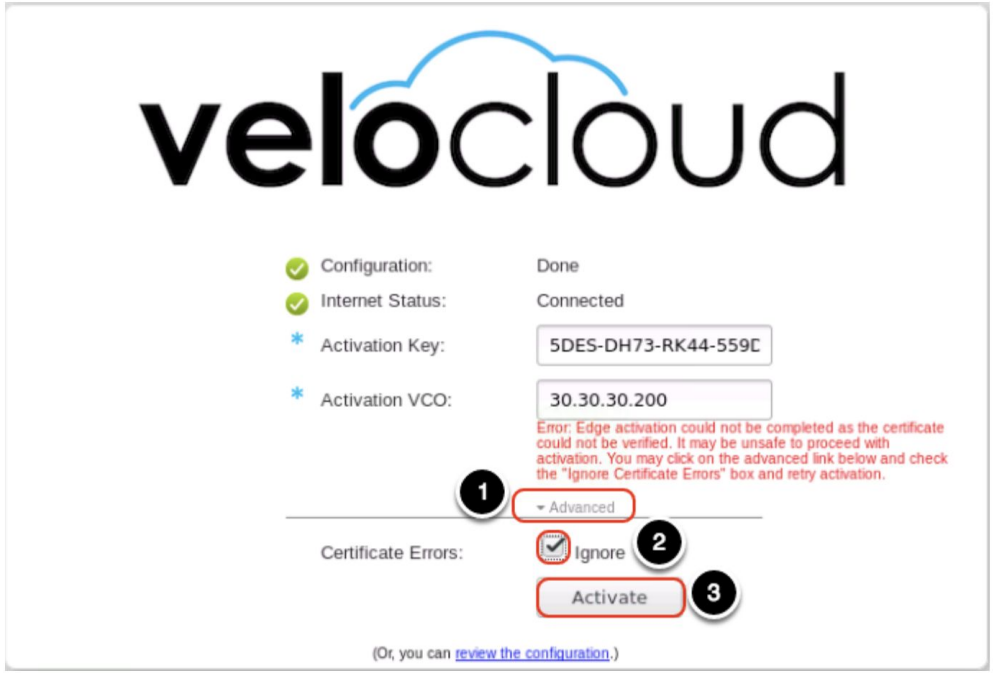
Note The 172.31.5.60 is the IP address of a CVP node, and that you must use the HTTP (and not HTTPS) URL to the bootstrap file. This ensures that the specified devices, after they ZTP, will show up under the undefined container of the specified CVP.

```
[root@cvp1-dhcp dhcp]# cat dhcpd.conf
#
# DHCP Server Configuration file.
# see /usr/share/doc/dhcp*/dhcpd.conf.sample
# see 'man 5 dhcpd.conf'
#



subnet 172.31.0.0 net
range 172.31.3.212 17
option domain-name "s
}

host esx21-vm20 {
option dhcp-client-id
fixed-address 172.31.
option bootfile-name
}

host esx21-vm22 {
option dhcp-client-identifier 00:0c:29:d1:64:e1;
fixed-address 172.31.3.213;
option bootfile-name "http://172.31.5.60/ztp/bootstrap";
}
```



Velocloud Activation Rollback

Client Activation



As this is a lab environment, Certificate Error should be ignored.

1. Click on **Advanced**
2. Click the **Ignore checkbox** for Certificate Error.
3. Click **Activate**

Insecure Bootstrapping

1. A connected router establishes a secure channel with a controller over TLS
2. The router generates a public/private key pair and a CSR and send the CSR to the controller CA over TLS channel
3. The CA issues the certificate
4. The router uses the certificate on the control plane

CSR Generation on an Edge Device

```
NAME=$2
SUBJ=/C=[redacted]/L=[redacted]/O=[redacted]/OU=[redacted]/CN=${NAME}.[redacted]/emailAddress=${NAME}@[redacted]

case $1 in
    gen)
        openssl genrsa -out ${NAME}.key 2048
        openssl req -new -key ${NAME}.key -nodes -out ${NAME}.csr -subj "${SUBJ}"
        ;;
    p12)
        openssl pkcs12 -export -inkey ${NAME}.key \
                -in ${NAME}.crt -out ${NAME}.p12 -passout pass:
        ;;
    *) usage;;
esac
```

Certificate Generation on a CA server

```
if content is not None:
    MASTER = 'ctl'
    DAYS = ["-days", "365"]
    SERIAL = ["-CAcreateserial", "-CAserial", "server/ca.seq"]

    args = [
        'openssl', 'x509', '-req'
    ] + DAYS + [
        '-CA', 'server/{}.ca.crt'.format(MASTER),
        '-CAkey', 'server/{}.key'.format(MASTER)
    ] + SERIAL
    [illegible]
    # openssl x509 -req -days 365 -CA server/ctl.ca.crt -CAkey server/ctl.key -CAcreateserial -CAserial server/ca.seq
    [illegible]
    proc = Popen(args, stdout=PIPE, stderr=PIPE, stdin=PIPE)
    outs, errs = proc.communicate(content, timeout=15)
    proc.wait(3)
    if int(proc.returncode) != 0:
        print('error')
        [illegible]
    else:
        [illegible]
```

ZTP URL Padding Oracle

link = “$(echo "ztp?ip=1.1.1.10&m=24&token=c28ds340df82g317402&dns=8.8.8.8" | openssl enc -e -aes-256-cbc -pbkdf2 -k PrettyGoodPreSharedKey -nosalt | base64 -w0”)

curl https://orchestrator/activate?$link

The activation script replies HTTP 500, if the encrypted link cannot be decrypted

Oracle

ZTP URL Padding Oracle

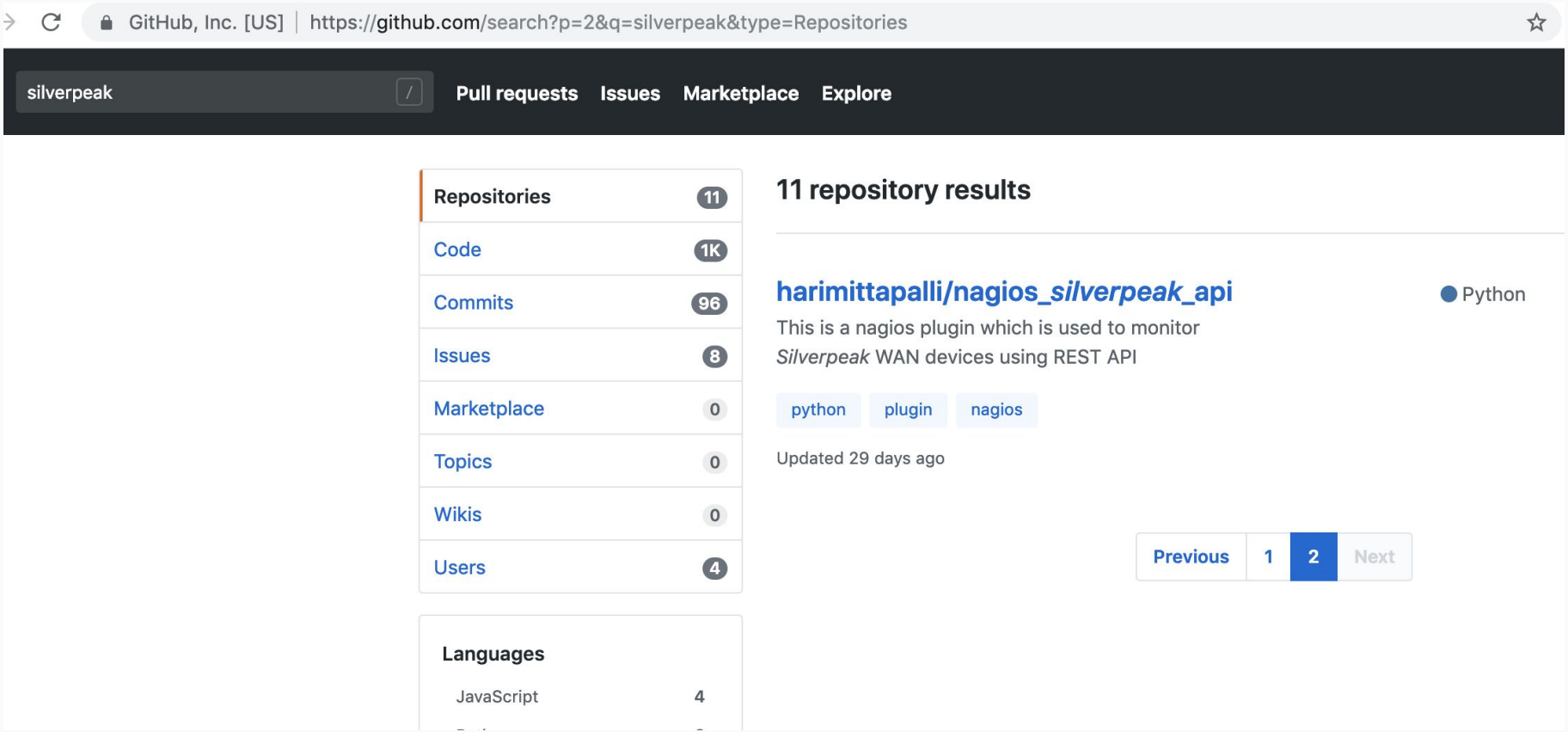
- Vulnerability to padding oracle attack
  - If an attacker has an encrypted ZTP link and access to ZTP service (oracle) he will recover the cleartext
- Malleability
  - There is no any authentication, an attacker, in theory, can change the encrypted ZTP URL so the new cleartext will contain a malicious DNS server address
- Solution
  - Use AEAD primitives (AES-GCM, ChaCha20-Poly1305, etc.)

SilverPeak Crypto Case

SilverPeak Crypto

- SilverPeak uses Racoon as an IPsec library
- No AEAD ciphers for data plane
- It uses TLS on the control and orchestration planes
- The main protocol is self-invented IKE-less IPsec over UDP
- Self-invented 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

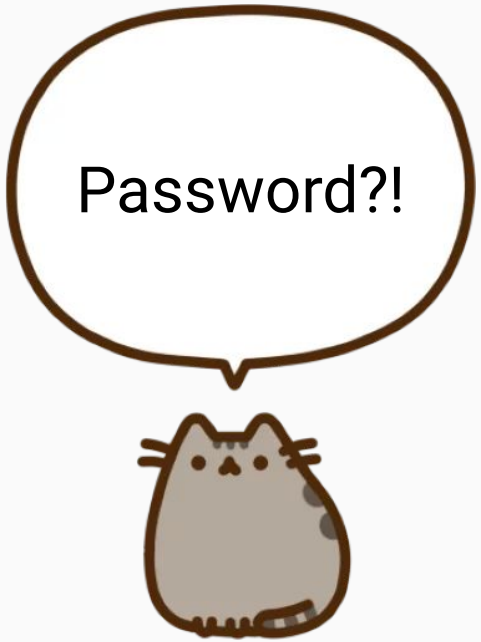
```
                    memory / swap / alarms / tunnels / nexthops / vrrp / diskinfo
```

-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 ''
```

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

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

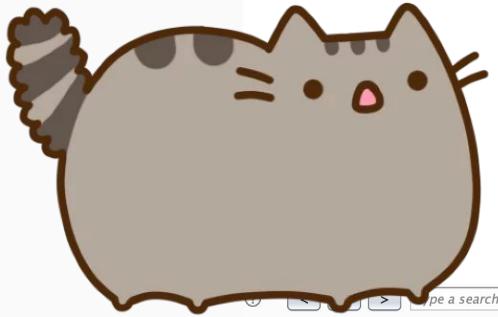
tunnelsConfigAndStates API Result

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

Request

Raw | Params | Headers | Hex

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



Type a search term | 0 matches

Response

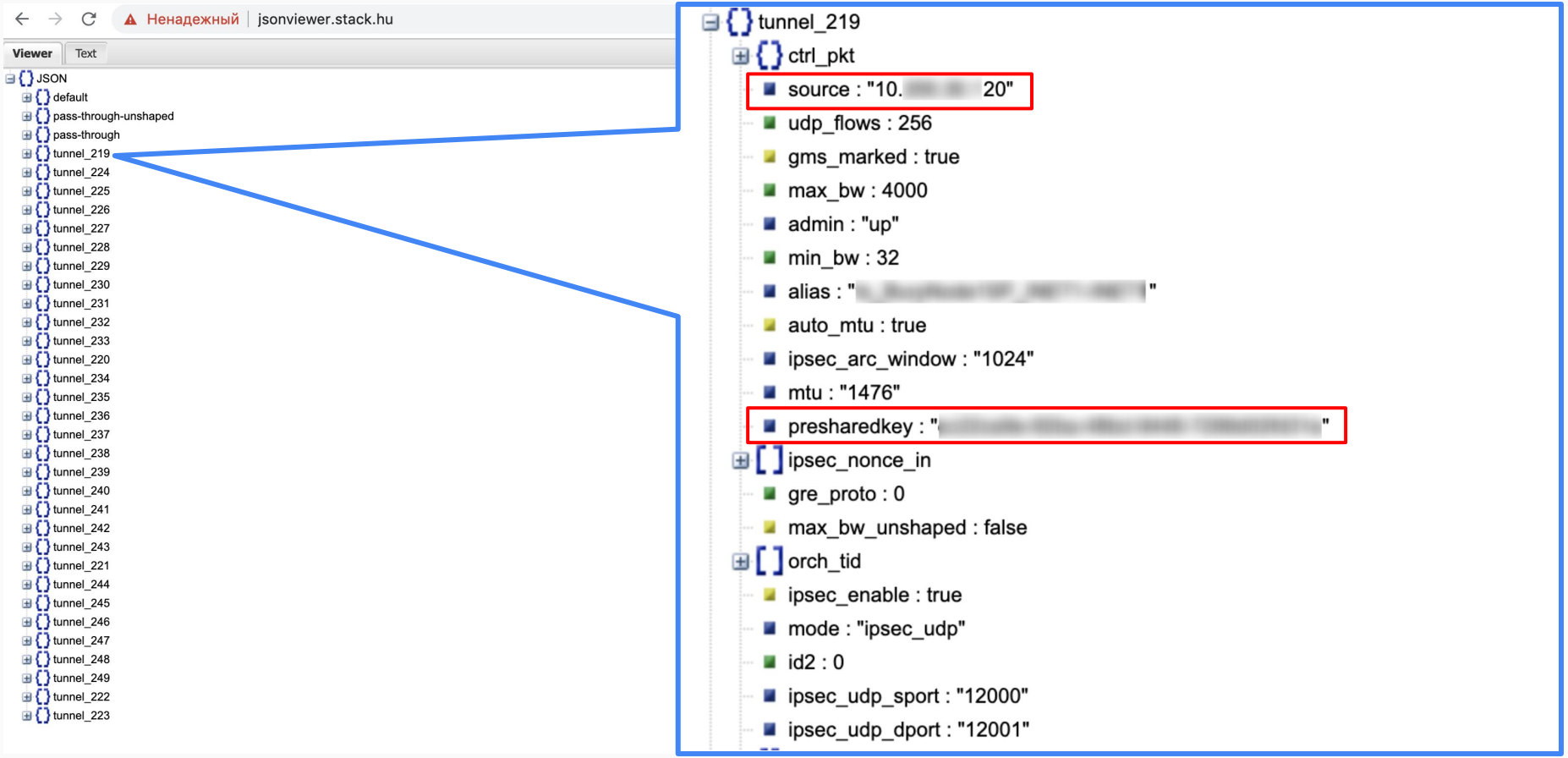
Raw | Headers | Hex

```
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"
[redacted]
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"

Nonce - number only used once?

ipsec_nonce_out:

0:	185
1:	254
2:	208
3:	161
4:	75
5:	11
6:	98
7:	18
8:	247
9:	231
10:	181
11:	137
12:	240
13:	159
14:	177
15:	112
16:	56
17:	143
18:	31
19:	101
20:	209
21:	178
22:	159
23:	49
24:	208
25:	79
26:	88
27:	138
28:	45
29:	81
30:	199
31:	162

ipsec_nonce_in:

0:	210
1:	151
2:	181
3:	240
4:	176
5:	26
6:	213
7:	170
8:	189
9:	230
10:	165
11:	121
12:	42
13:	189
14:	83
15:	54
16:	213
17:	54
18:	152
19:	175
20:	16
21:	254
22:	51
23:	16
24:	255
25:	23
26:	146
27:	148
28:	197
29:	50
30:	87
31:	87

orch_tid:

0:	208
1:	3
2:	2
3:	52
4:	126
5:	108
6:	27
7:	151
8:	168
9:	45
10:	158
11:	49
12:	225
13:	219
14:	195
15:	170

Nonce - number only used once?

ipsec_nonce_out:

0:	185
1:	254
2:	208
3:	161
4:	75
5:	11
6:	98
7:	18
8:	247
9:	231
10:	181
11:	137
12:	240
13:	159
14:	177
15:	112
16:	56
17:	143
18:	31
19:	101
20:	209
21:	178
22:	159
23:	49
24:	208
25:	79
26:	88
27:	138
28:	45
29:	81
30:	199
31:	162

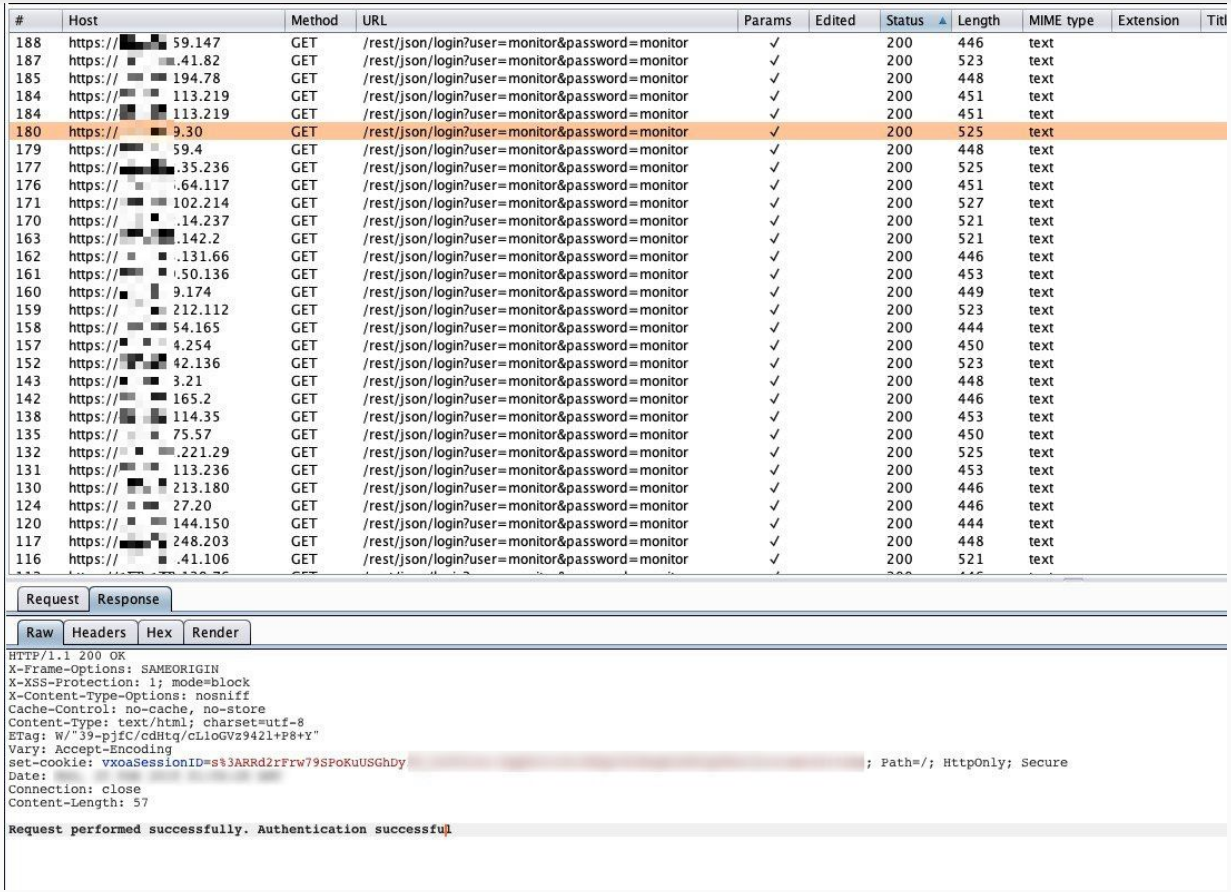
ipsec_nonce_in:

0:	210
1:	151
2:	181
3:	240
4:	176
5:	26
6:	213
7:	170
8:	189
9:	230
10:	165
11:	121
12:	42
13:	189
14:	83
15:	54
16:	213
17:	54
18:	152
19:	175
20:	16
21:	254
22:	51
23:	16
24:	255
25:	23
26:	146
27:	148
28:	197
29:	50
30:	87
31:	87

PoC

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

PoC on Burp Intruder



#	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

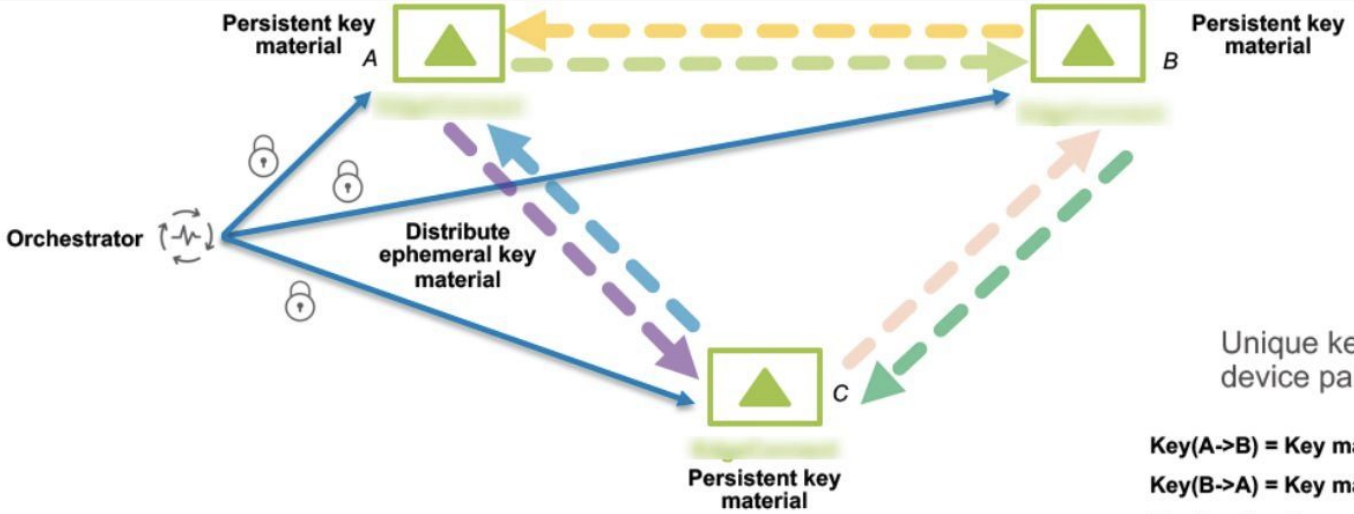
Raw | Headers | Hex | Render

```
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

- November 2019
  - 954 SilverPeak devices
  - 490 alive
  - 168 devices have monitor:monitor user
  - 15 devices have admin:admin user
- November 2018
  - 571 SilverPeak devices
  - 380 alive
  - 150 devices have monitor:monitor user
  - 3 devices have admin:admin user
- May 2019
  - 601 SilverPeak devices
  - 396 alive
  - 184 devices have monitor:monitor user
  - 3 devices have admin:admin user

SilverPeak's 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)

Key Management Black Box Analysis

- Pre-shared keys are generated by the orchestrator
  - It is not possible to view, set or change a PSK using the WebUI
- PSK are the same on all tunnels within a domain
  - A spoke with more than 20 tunnels has the same PSK
  - 5d30a54c-3233-434e-8481-8bf6ac5efa5c
- If A and B are IPsec peers then A's ipsec_nonce_in is equal to B's ipsec_nonce_out
- "Nonces" are the same
- We did not see that PSK or nonces are changed

Hard-coded Credentials

Fortinet Hardcoded Keys

FG-IR-18-100: Hard-coded keys in FortiGuard

At a glance:

IR Number	FG-IR-18-100
Date	Nov 20, 2019
Risk	●●●○○
Impact	Information disclosure
CVE ID	CVE-2018-9195
CVRF	Download

PSIRT Advisory

Hardcoded cryptographic key in the FortiGuard services communication protocol

Summary

Use of a hardcoded cryptographic key in the FortiGuard services communication protocol may allow a Man in the middle with knowledge of the key to eavesdrop on and modify information (URL/SPAM services in FortiOS 5.6, and URL/SPAM/AV services in FortiOS 6.0.; URL rating in FortiClient) sent and received from Fortiguard severs by decrypting these messages.

Impact

Information disclosure

Affected Products

FortiOS 6.0.6 and below
FortiClientWindows 6.0.6 and below
FortiClientMac 6.2.1 and below

FG-IR-18-100: Hard-coded keys in FortiGuard

- SecConsult report
- Fortinet products, including FortiGate and Forticlient regularly send information to Fortinet servers (DNS: guard.fortinet.com) on
  - UDP ports 53, 8888 and
  - TCP port 80 (HTTP POST /fgdsvc)
- The messages are encrypted using XOR “encryption” with a static key
- The protocol messages contain the following types of information:
  - Serial number of the Fortinet product installation
  - Full HTTP URLs of users web surfing activity
  - Unspecified email data
  - Unspecified AntiVirus data

FG-IR-19-007: Hard-coded keys in Fortinet SD-WAN

At a glance:

IR Number	FG-IR-19-007
Date	Nov 19, 2019
Risk	●●○○○
Impact	Information Disclosure
CVE ID	CVE-2019-6693
CVRF	Download

PSIRT Advisory

Use of a hard-coded cryptographic key to cipher sensitive data in configuration backup files

Summary

Use of a hard-coded cryptographic key to cipher sensitive data in FortiOS configuration backup file may allow an attacker with access to the backup file to decipher the sensitive data, via knowledge of the hard-coded key.

The aforementioned sensitive data includes users' passwords (except the administrator's password), private keys' passphrases and High Availability password (when set).

Impact

Information Disclosure

Affected Products

5.6.10 and below
6.0.6 and below
6.2.0

FG-IR-19-007: Hard-coded keys in Fortinet SD-WAN

- FortiGate and FortiManager store passwords in encrypted format. The following command sets a password “test” for the admin user

```
config system admin user
    edit "admin"
        set password ENC
NzIyMjg3MTg2MTI1MjQ0MVdSZNNjo34BASXf0rFqWojteb6vF0dHmhzcDAsWzUzEpLcE35aMZx+7z16mdyra/eSco3TgN3CF0/8agm00Ve
12mBsMyQFqu2KRAJWOv8opm9laO2/t/c79al9OO4ANDjnzqONY3XYo682U7oFCsX7vlfs2
```

- It’s base64 encoding of IV and encrypted password

```
7222871861252441WRd\xd3c\xa3~\x01\x01%\xdf\xd2\xb1jZ\x88\xedy\xbe\xaf\x17GG\x9a\x1c\xdc\x0c\x0b\x16\xcdL\x
c4\xa4\xb7\x04\xdf\x96\x8cg\x1f\xbb\xcf^\xa6w*\xda\xfd\xe4\x9c\xa3t\xe07p\x85\xd3\xff\x1a\x82m4U\xedv\x98\
x1b\x0c\xc9\x01j\xbbb\x91\x00\x95\x8e\xbf\xca)\x9b\xd9Z;o\xed\xfd\xce\xfdj_N;\x80\r\x0e9\xf3\xa8\xe3X\xddv
(\xeb\xcd\x94\xee\x81B\xb1~\xef\x95\xfb6
```

- The key used to encrypt the password is the same for all devices
- So it makes possible to decrypt a password from any configuration file if an attacker has one

Citrix Hard-coded RSA Keys

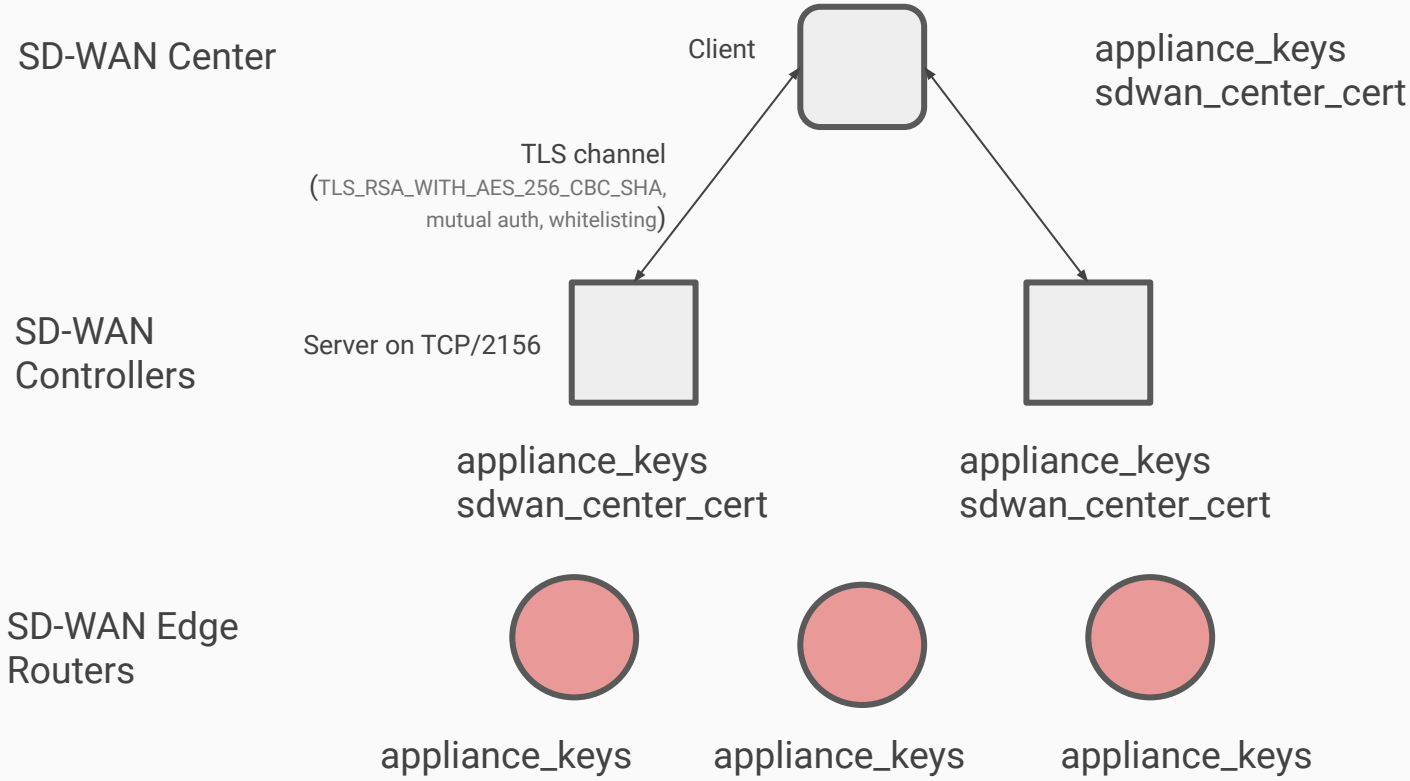
Overview

- **All** Citrix NetScaler SD-WAN appliances used **the same pre-installed** RSA key pair and the corresponding self-signed certificate
- This certificate was used in Controller - Orchestrator communication protocol
- 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

Controller - Orchestrator Protocol



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`)

Versa Hardcoded Passwords

Why do versa devops use “versa123”?

```
from fabric.api import sudo
from fabric.api import env
from fabric.api import run

env.user = "Administrator"
env.host_string = '10.192.28.176'
env.password = "versa123"


def test():
    sudo('ls -lrt')
    sudo("sudo sed -i '/singh/ s/$/anythin/' /tmp/pompina")

test()
```

joshuap-cfy / frontier-versa-sdwan-poc-0117

forked from Cloudify-PS/cloudify-versa-plugin

Code | Pull requests 0 | Projects 0 | Wiki | Insights

187 lines (175 sloc) 5.64 KB

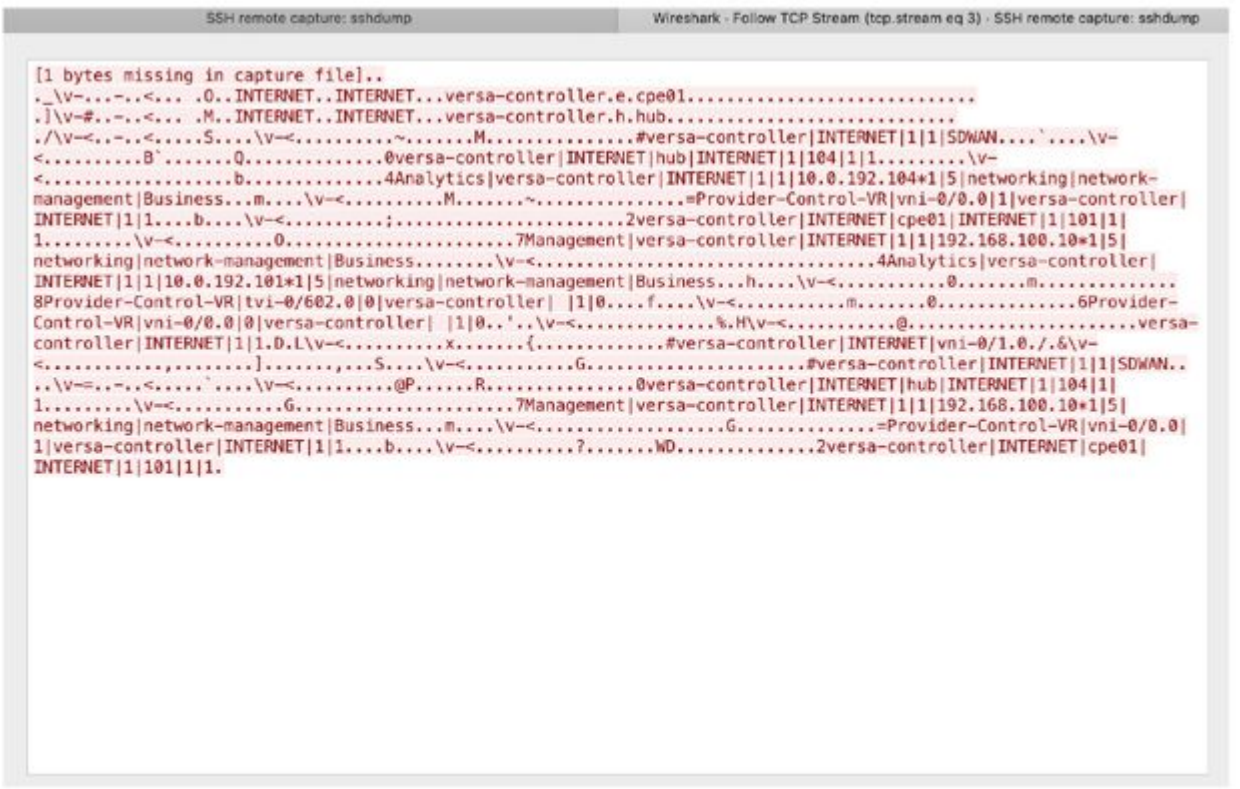
```
1  #Add and configure network with DHCP,DNS,Firewall to exsistent organization
2  #Organization must have one free interface
3  tosca_definitions_version: cloudify_dsl_1_3
4
5  imports:
6    - imports.yaml
7
8  inputs:
9      versa_url:
10         default: "https://172.19.0.210:9183"
11     client_id:
12         default: "voae_rest"
13     client_secret:
14         default: "asrevnet_123"
15     username:
16         default: "Administrator"
17     password:
18         default: "versa123"
```

Versa Hard-coded Passwords

- Versa Analytics Driver REST API (/opt/versa/bin/versa-analytics-driver) uses the hardcoded credentials located at the /opt/versa/var/van-app/properties/application.properties file
- The credentials are used to perform HTTP Basic Authentication
- The credentials are equal to vanclient:88347b9e8s6$90d9f31te366&d5be77 and they are the same for all Versa Analytics deployments

Cleartext Communications

Versa Analytics TCP 1234 Service Cleartext Communications



B4N SD-WAN Secure Communications

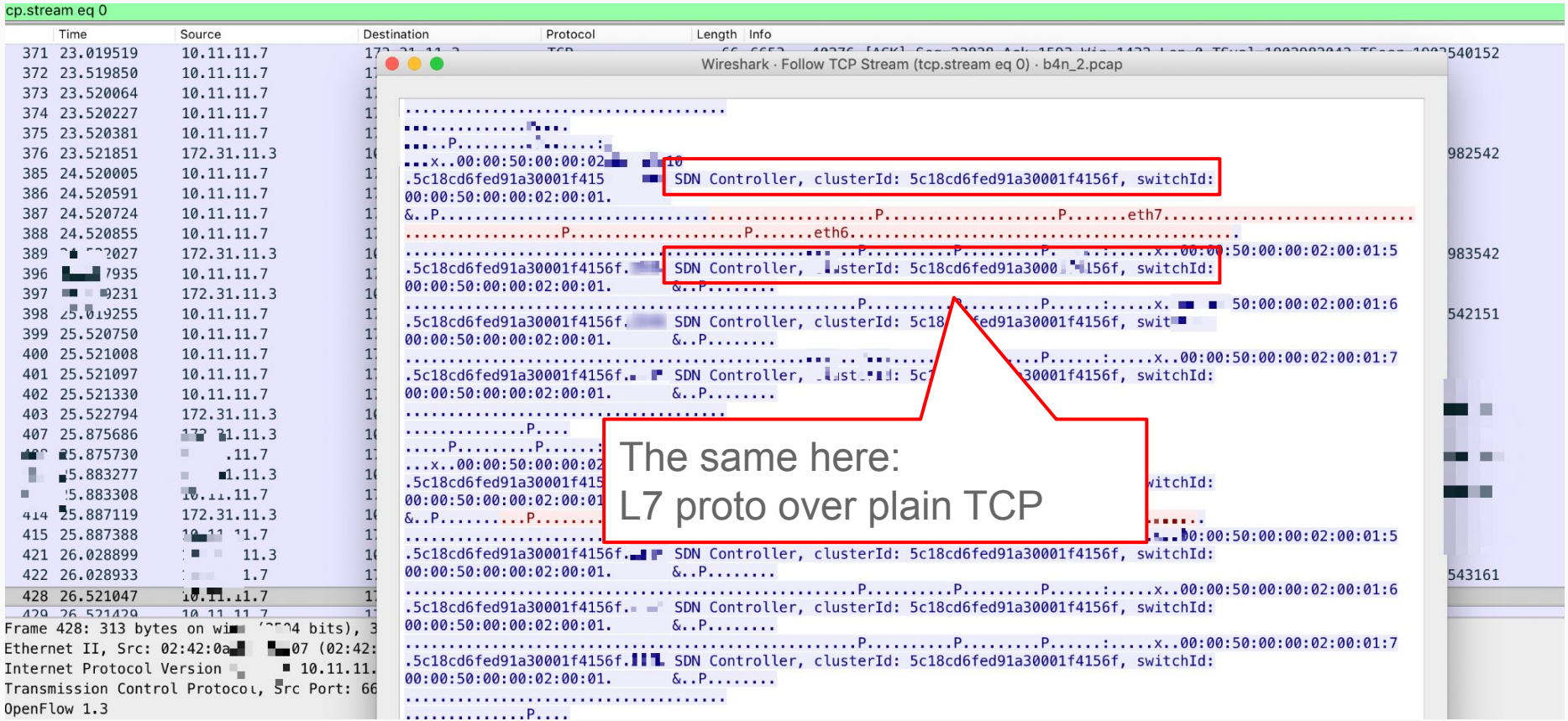
- No crypto approach
- Unprotected
  - TCP 830 (GRPC)
  - TCP 5000 (API)
  - TCP 6653 (OpenFlow)
  - TCP 27017 (Mongo)
- No mutually authenticated
- There is no ready to use decisions for some protocols (e.g., OpenFlow)
- Brain4Net says we have tested a deployment without secure communications

B4N GRPC

```
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

B4N OpenFlow



TLS Vulnerability Measurements

Overview

- The research began with “Scalable Scanning and Automatic Classification of TLS Padding Oracle” paper
- Investigated scope
  - Alexa top million websites
  - The CBC padding oracle attack
- What about SD-WAN deployments on the Internet?
  - Probably, they are not in Alexa top websites

Method

1. Run TLS-Attacker against the appropriate interfaces from the SD-WAN Knowledge Database.
2. If vulnerabilities were found, rescan the node two times to minimize false positives.
3. If the vulnerabilities are still present, check them using PoC scripts in Python.
4. Save the confirmed results to the database.

We scanned 7200 SD-WAN nodes

Attack	Number of vulnerable nodes
Sweet32	1873
CBC Padding Oracle	121
CRIME	30
Logjam	29
DROWN	14
ROBOT	6
Heartbleed	1

Some Results

Product	Attacks	Version
Talari SD-WAN	Sweet32	r6_1_ga_p6_11032017
Nuage SD-WAN VNS	Bleichenbacher, Breach	
SilverPeak Unity Edge Connect	Breach	
Cisco SD-WAN	Breach	
Citrix NetScaler SD-WAN	Bleichenbacher, Sweet32	
Citrix SD-WAN Center	SSL Poodle	
Versa Flex VNF	Bleichenbacher	20161214-191033-494bf5c-16.1r2

Some Results

Product	Attacks	Version
Sonus SBC Management Application	Bleichenbacher, Breach	r6_1_ga_p6_11032017
Sonus SBC Management Application	Sweet32	5.0
FortiGate SD-WAN	SSL Poodle, Sweet32, EarlyCcs	
RiverBed Steel Head	Padding Oracle, CVE-20162107, Sweet32	0.15.8

Secure Design

Scope

- Orchestration plane
- Zero-touch provisioning
- Bringup protocols
- Control plane
- Data plane protection
  - Encrypted overlays
  - VPN virtual functions

Peculiarities

- Huge number of interfaces, services, protocols and data flows
- Different platforms
- SD-WAN edge devices (uCPE) often do not have HSM modules (TPM, secure microcontrollers)
- CPE (uCPE) devices without hardware-backed crypto are like cloud instances

SD-WAN Bringup with SPIRE

SD-WAN Bringup

- SD-WAN Bringup
  - All entities authenticate each other
  - Edge routers must securely join the SD-WAN
  - All entities establish secure communication channels between each other
  - All entities have identities in cryptographic sence
- Cisco defines and describes own bringup security protocol very thoroughly
- Let’s see how we can do the same using existing projects

Authentication

- The following methods are used
  - TLS client authentication
  - Challenge-response protocols
  - Token-based - We check that a router possess a token
- HSM-backed routers should use the first two methods
- Cloud routers should use a token-based method due to the fact that the private key can be stolen easily

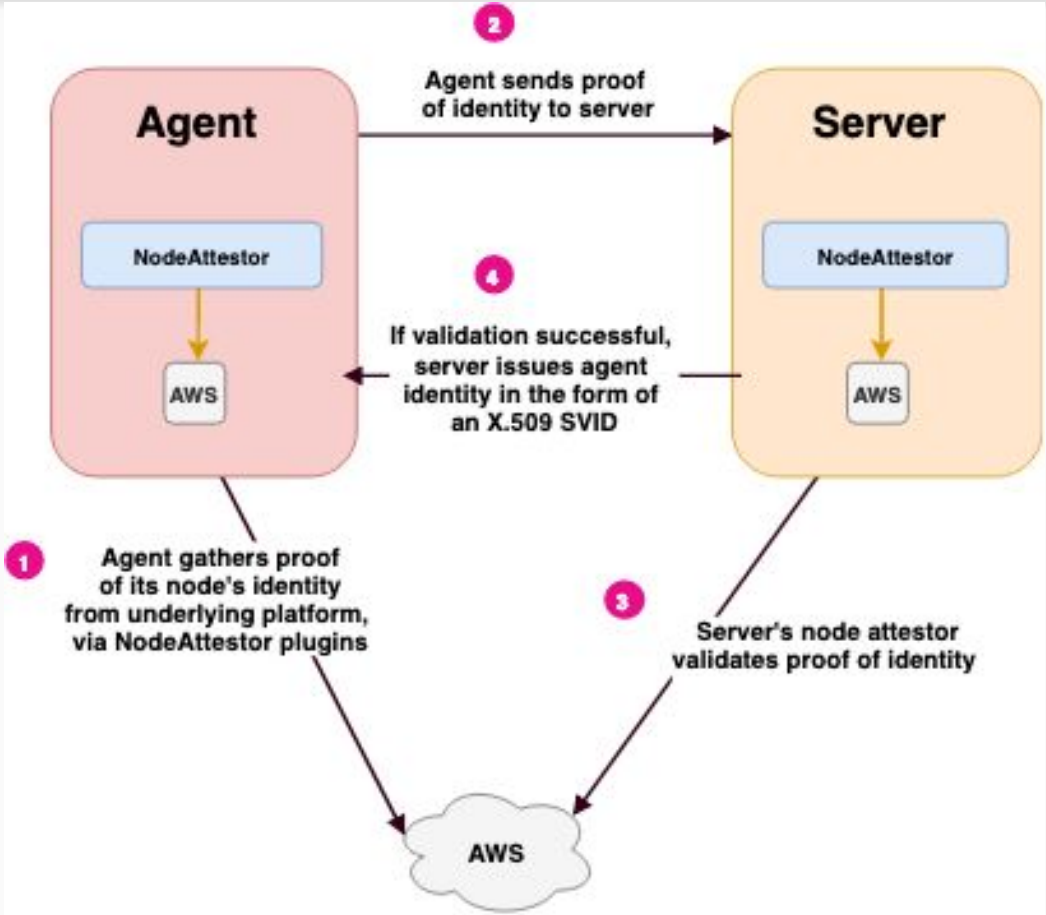
Token-based Authentication

- If a CPE doesn’t have a HSM/TPM or another hardware-backed secure storage an identity key can be easily obtained or copied
- In this case CPE should be considered as a virtual node
- The main authentication method here is based on join token conception

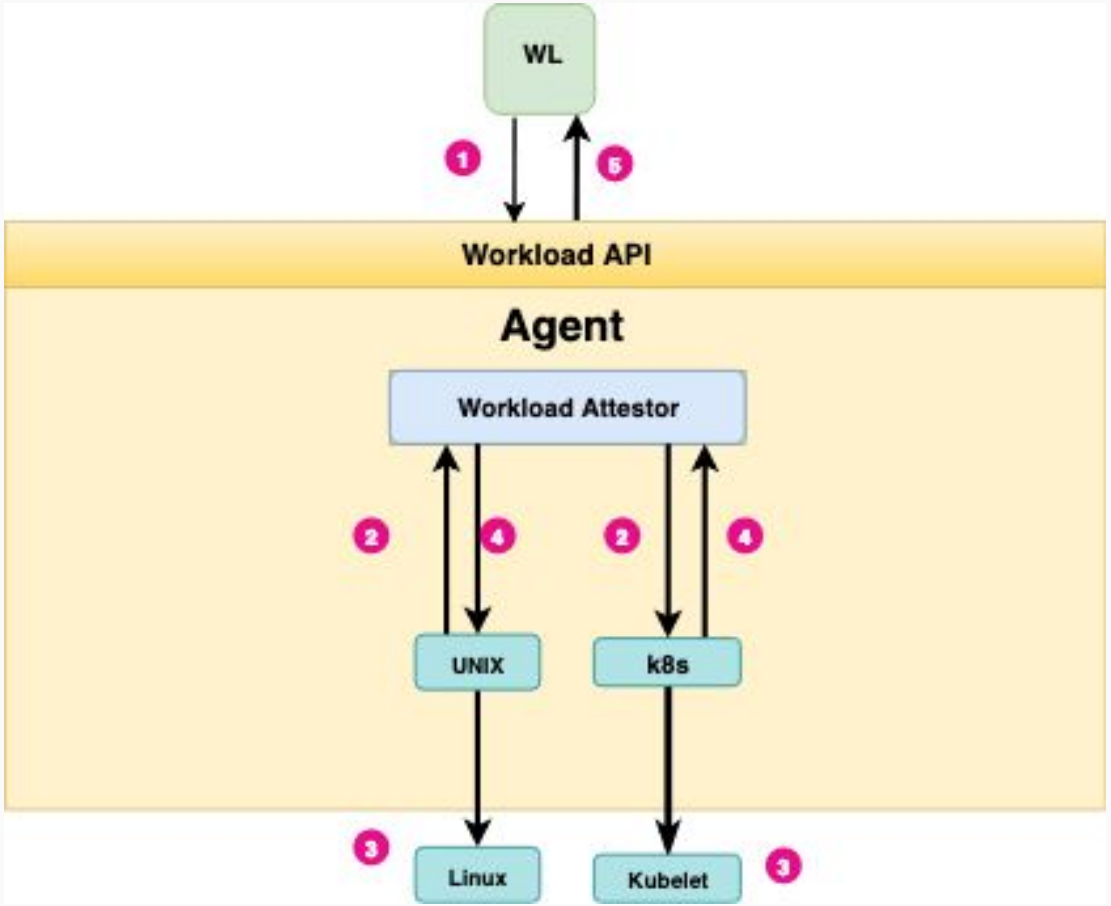
SPIFFE and SPIRE

- SPIFFE - The Secure Production Identity Framework For Everyone
- SPIFFE ID
  - X509
  - JWT
- SPIRE - SPIFFE Runtime Environment
- SPIRE 101
- Examples
  - SPIFFE
  - SPIRE

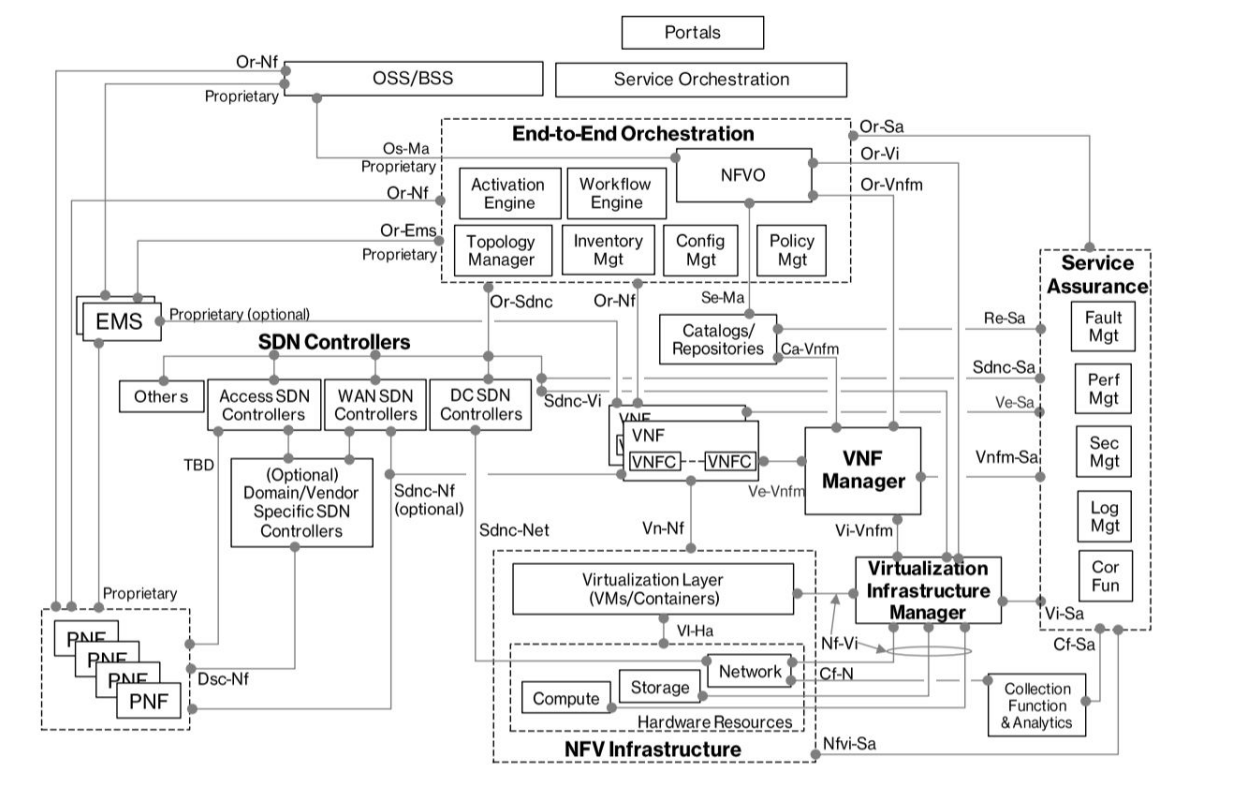
SPIRE Node Attestation Example



SPIRE Workflow Attestation Example



SD-WAN Architecture



Securing SD-WAN with SPIRE

- Design document (commented by Evan Gilman)
- The goal is to implement scalable identification of SD-WAN entities
- Mappings
  - SPIRE Server is deployed on the SD-WAN Controller
  - SPIRE Agent is deployed on each SD-WAN edge device, controller, orchestrator, analytic systems, etc.
  - SPIRE workloads are SD-WAN processes (points) which need an identity

Node Attestors

SPIRE Attestor	Applicability within SD-WAN
Join token	Cloud only
x509pop, sshpop, tpm	On-prem, cloud
aws_iid, azure_msi, gcp_iit	Cloud-based SD-WAN: Azure, GCP, AWS

Keys

- Machine identity
  - PKC key pair, long-term, X509
  - The identity may refer to a customer or a purpose
  - The certificate is issued by customer's CA
  - Stored in TPM or in persistent memory
- Agent identity (SPIRE native)
  - PKC key pair, short-term, in SVID format
  - The identity refers to a SPIRE Agent on a machine
  - The certificate is issued by SPIRE CA or Upstream CA
- Workload identity (SPIRE native)
  - PCK key pair, short-term, in SVID format
  - The identity refer to a service on a concrete machine with a SPIRE Agent
  - The certificate is issued by SPIRE CA or Upstream CA
  - Stored in memory or on disk

Assumptions

- X509pop attestor is used
- Each SD-WAN node gets the following credentials on a provisioning phase
  - A machine key and the corresponding certificate issued by a vendor or customer CA
  - A trust bundle CA certificate
- SPIRE Server has the machine key CA certificate
- Any interaction with a controller begins with establishing trust through SPIRE
  - SPIRE
  - ZTP

SPIRE commands

Server-side

```
#spire-server entry create -node -spiffeID
spiffe://sdwan.com/router1 -selector
x509pop:subject:cn:example.com

#spire-server entry create -ttl 96 -spiffeID
spiffe://sdwan.com/router1/ztp -parentID
spiffe://example.com/router1 -selector unix:uid:1000
```

Agent-side

```
# spire-agent run -conf agent.conf &
# su -c "./cmd/spire-agent/spire-agent api fetch x509 "
ztp -write ./svid/
```

Securing SD-WAN with SPIRE

- Pros
  - Unified and common mechanism for entire SD-WAN infrastructure
  - It can be reused in or integrated with cloud native (Kubernetes) or service mech (Istio, Envoy) systems
  - SPIRE is a root of trust
  - SPIRE already has strong authentication methods leveraging TPM, SSH keys or X509 certificates
  - You can implement a new crypto protocol and add it to SPIRE
- Cons
  - Depends on 3rd party SPIFFE/SPIRE framework
  - Developed SD-WAN will inherit SPIFFE/SPIRE features

Key Management

Crypto in SD-WAN

- Crypto for SD-WAN is still in its infancy
- There are no known specific standards (RFC, ISO, etc.)
- Vendors have to invent key distribution protocols
- SD-WAN vendors do not reuse mechanisms from cloud native projects

Key Management Overview

- Control plane
  - TLS/DTLS, SSH
  - ZTP
- Data plane and cryptographic overlays
  - IPsec
  - WireGuard / nQUIC/ Noise
  - PQC protocols
  - IKE-less IPsec
  - SSH
  - Custom cryptographic protocols (like Cisco OMP)
- How to manage cryptographic keys?

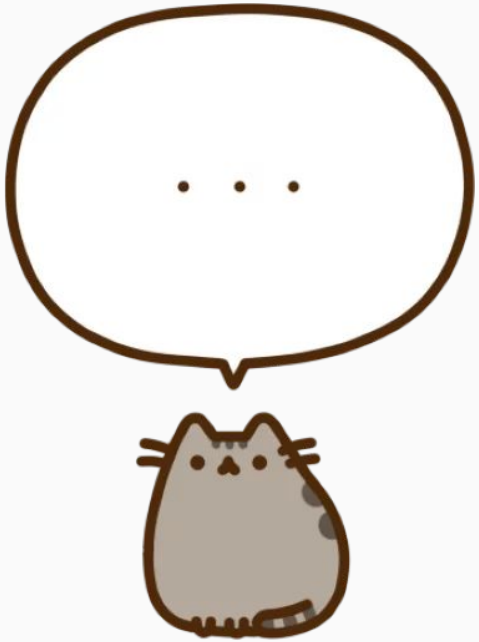
Why peer-to-peer key exchange is not the case within SDN/SD-WAN?

- SDN mainly use peer-to-controller trust model
- Latency
- Entropy generation on a CPE may be not a good idea
- Complexity (key rotation)
- Network shape is not persistent

Source: https://www.ietf.org/archive/id/draft-carrel-ipsecme-controller-ike-01.txt

SD-WAN Key Management Drafts

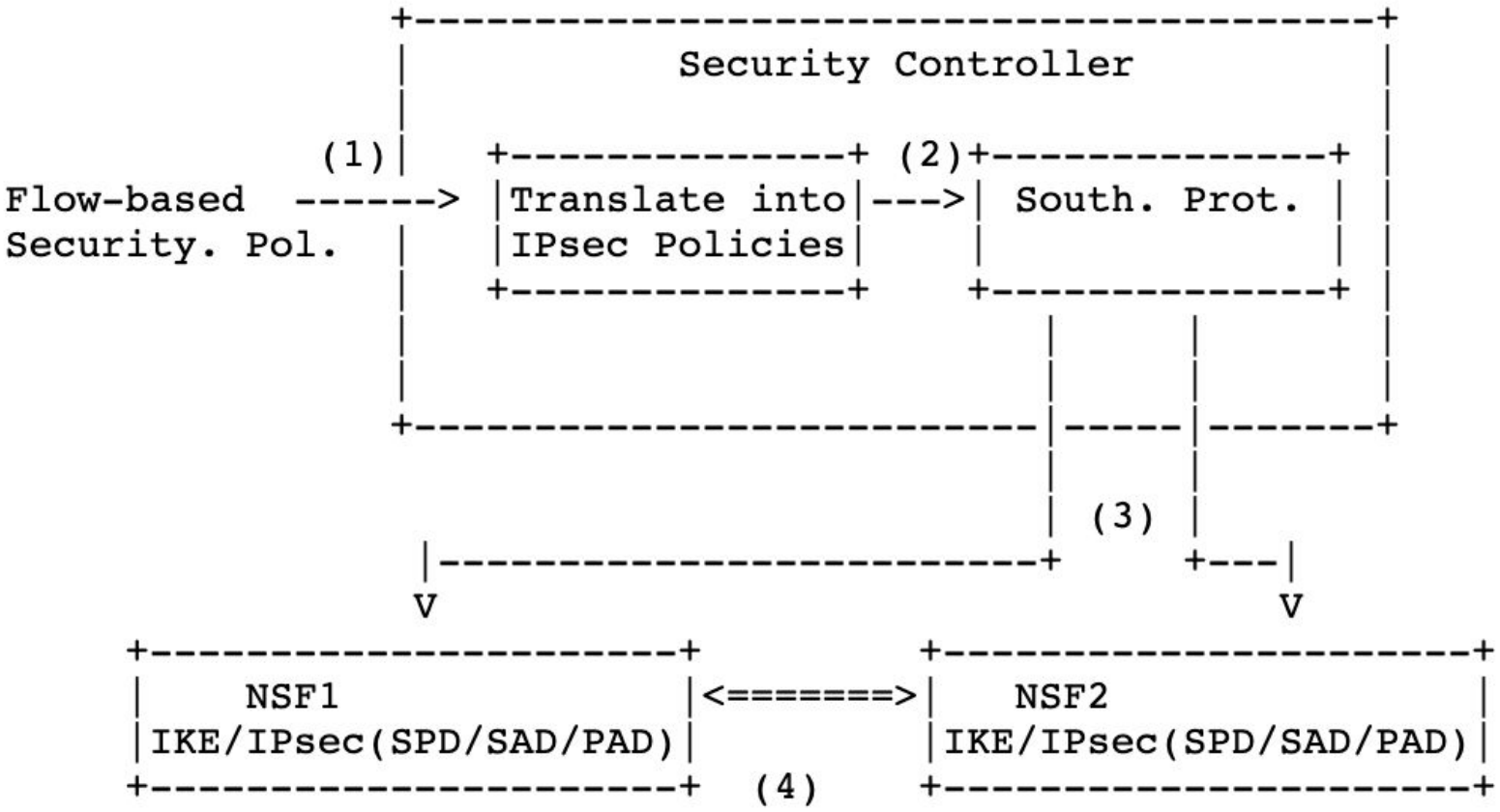
- Software-Defined Networking (SDN)-based IPsec Flow Protection
- IPsec Key Exchange using a Controller
- A YANG Data Model for SD-WAN VPN Service Delivery



SDN-based IPsec Management

- One controller, IKE/IPsec in the NSF
  - Controllers deliver credentials (PSK, private keys, certificates) to edge devices over secure channels
  - Edge devices perform IKE (or another key exchange protocol) and then IPsec
- One controller, IPsec in the NSF
  - Controllers deliver credentials (PSK, private keys, certificates) to edge devices over secure channels
  - Edge devices perform IKE (or another key exchange protocol)
- Two controllers, IKE/IPsec in the NSF
  - Controllers negotiate credentials and deliver them to edge devices over secure channels
  - Edge devices run IPsec
- Two controllers, IPsec in the NSF
  - Controllers perform key exchange and deliver session (transport) keys to edge devices over secure channels
  - Edge devices run IPsec

SDN-based Flow Protection



SDN-based Flow Protection Problems

- The main problem is that one peer (controller) dictates the key entirely - an edge router does not contribute to the key
  - If a controller's PRNG is compromised, subverted or insecure there is no chance to get a key with strong cryptographic properties
  - We know such incidents (Juniper, Fortinet)
- The security of the protocol must be analysed
- It is bad crypto hygiene to use data channel for keys
- Designing a secure mechanism that uses this approach is not necessarily straightforward

SDN-based Flow Protection Insecure Protocol

- The controller has a weak PRNG
- Two protocols are used between controller and edge routers: TLS 1.3 and a protocol within the Noise protocol framework
- The controller generates “random” Curve25519 private key for Noise and send it over TLS-channel
- An attacker can predict the Noise private key due to weak PRNG
- An edge router receives the private key, generates the public key and establishes a new channel using a Noise protocol
- If a chosen Noise protocol pattern or its implementation is vulnerable to KCI attack then an attacker can impersonate the controller

SDN-based Flow Protection Insecure Protocol

- KCI - Key Compromise Impersonation
- KCI is a weakness of an authenticated key exchange protocol that allows an attacker who has compromised the secret credentials of a client to impersonate any peer to the client
- For example, in WireGuard
    - The handshake responder cannot assume the connection is authentic until they have received at least one valid data packet; otherwise, they are vulnerable to key-compromise impersonation (KCI)

Key distribution and rotation tools for WireGuard

> *Ahh, my apologies, I read "pre-shared" and assumed you were talking*
> *about PSK mode. But I think you're really interested in more general*
> *key distribution.*
>
> *Some people are just doing this over TLS with basic rest APIs*
> *beforehand.*

If you have established a TLS session from A to B then you can just derive your PSK from the master secret of that TLS session.

That's exactly what the Cisco AnyConnect protocol does, for "upgrading" its HTTPS connection to DTLS. It first connects via HTTPS and does all the authentication and client configuration that way, and then establishes a UDP connection *if* it isn't prevented by stupid firewalls.

Source: https://lists.zx2c4.com/pipermail/wireguard/2018-May/002904.html

Key Export

- A and B have already established a TLS channel
- A and B need a new secret key
- k = PRF(master_secret)
- Is it secure?

RFC 5705

- RFC 5705 Keying Material Exporters for Transport Layer Security (TLS)
- Requirements
  - Both client and server need to be able to export the same EKM value
  - EKM values should be indistinguishable from random data to attackers who don't know the master_secret
  - It should be possible to export multiple EKM values from the same TLS/DTLS
  - Knowing one EKM value should not reveal any useful information about the master_secret or about other EKM values
- Designing a secure mechanism that uses exporters is not necessarily straightforward

RFC 5705

```
K = PRF(SecurityParameters.master_secret, label,
            SecurityParameters.client_random +
             SecurityParameters.server_random +
             context_value_length + context_value
             )[length]
```

Security of Key Exporters

- Safely Exporting Keys from Secure Channels: On the Security of EAP-TLS and TLS Key Exporters
- TLS-like protocols is a protocol as follows:
  - Authenticated and confidential channel establishment (ACCE)
  - The handshake includes a random nonce from each party
  - Each party maintains a value called the master secret during the handshake.
  - The session (exported) key is derived from the master secret, the nonces, and possibly some other public information
- The session key is indistinguishable from random from any party other than the two protocol participants

Security of Key Exporters

- An ACCE protocol is a protocol executed between two parties. The protocol consists of two phases, called the ‘pre-accept’ phase and the ’post-accept’ phase
- Pre-accept phase. In this phase a ‘handshake protocol’ is executed. Both communication parties are mutually authenticated, and a session key k is established. However, it need not necessarily meet the security definition for AKE protocols.
- Post-accept phase. In this phase data can be transmitted, encrypted and authenticated with key k.
- It was shown that
  - TLS_RSA is ACCE secure
  - TLS_DH is ACCE secure

Key Distribution Design

- Good
  - Key export within peer-to-peer model
- Not known
  - A custom protocol over secure channel (TLS-like protocol)
  - SDN-based IPsec management
- Bad
  - Use some constants (e.g., certificates) as a PSK

Conclusions

Design Philosophy

- When a vendor is developing a new product it should consider and take into account modern requirements, state-of-art technologies, attacks, etc.
- There are no guaranteed ways to succeed, but there are easy ways to fail: “insecure by design” approach is one of them

Thanks!

dnkolegov@gmail.com

https://twitter.com/dnkolegov

https://github.com/sdnewhop