A techpaper on

TeX – Traffic eXchange Cryptography and Connectivity

Author: Shabin Shanmugalingam, Patrick Dombrowski
Date: 10.10.2021
Version 1.0

Inhaltsverzeichnis

Overview

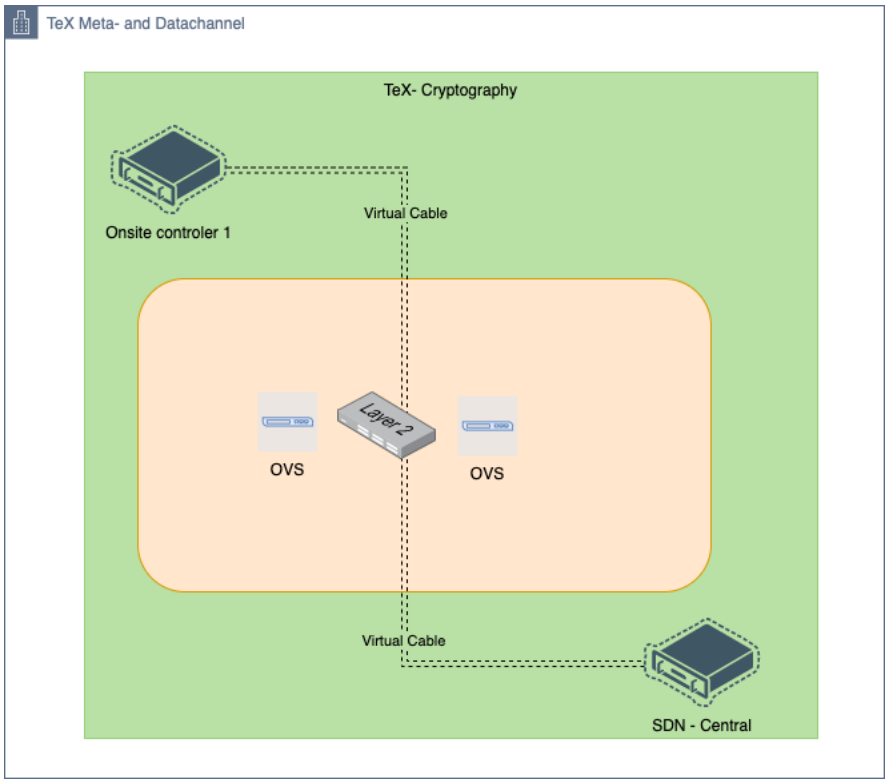
In Software-Defined Networking (SDN) depending on what kind of software and tooling you are using, you have different approaches on how to secure and keep the SDN secure to a certain, provable degree.

Mostly forgotten but still existent, are the additional mechanisms for monitoring, alerting, CI/CD all those fancy DevOps things.

TeX-Cryptography uses different ways and technics to provide –

• **Meta and data channel**: this layer is between the sender and the recipient. The SDN sends openflow Information through the meta channel and separates the actual data channel.
• **Virtual Cable**: each end-to-end encrypted channel is a secure channel.

These layers and other important aspects of the cryptography are described in detail in the following chapters.



We connect each on-prem or cloud controller with the so-called Virtual Cable, which was invented due to the PoC. It basically bridges, fragments the SDN into its areas of deployment.
The underlying layer doesn't really matter as far as it connects SDN controller and on-prem/cloud controller. Typically, you would use the Internet for that purpose, which might be a standard DSL, Fiber, Sat, Docsis or "what-ever" connection.

You are not bound by typically used MPLS or dark fiber approaches (more on that in the PoC Techpaper)

Virtual Cable

The Virtual Cable is a new-layer approach which is used to connect the SDN meta and data channel.

Note: Read about what wireguard can do for you!

Layer 1: 🔑 Security (Cryptography)

- Wireguard is used to connect each controller to an Open vSwitch (OVS)

Layer 2: 📞 Connectivity

- OVS is used to provide the openflow switch

Layer 3: 🚀 Integration

- A pseudo virtual interface is used to hold the management IP of the meta channel

In order to understand what the respective layer does, attached some code examples:

Note: The full configs and examples can be found @https://te-xchange.ch/exa

Wireguard server config example:

(If you made it that far into the document, you most likely have the knowledge to fill in the %VAR%)

```
[Interface]
Address = %SERVERIP%/24
ListenPort = %PORT%
PrivateKey = %KEY%
MTU = 1600
PostUp = iptables -A FORWARD -i %i -j ACCEPT; iptables -A FORWARD -o %i -j ACCEPT;
iptables -t nat -A POSTROUTING -o enp7s0 -j MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables -D FORWARD -o %i -j ACCEPT;
iptables -t nat -D POSTROUTING -o enp7s0 -j MASQUERADE
PostUp = sysctl -q -w net.ipv4.ip_forward=1

# 10: 10 > wgclient_%.conf
[Peer]
…
```

Wireguard client config example:

```
[Interface]
Address = %IP%
PrivateKey = %KEY%
MTU = 1600

[Peer]
PublicKey = %PUBKEY%
PresharedKey = %PSK%
AllowedIPs = %METACHANNELNETWORK%
Endpoint = %ENDPOINTWGSERVER:PORT%
PersistentKeepalive = 10
```

OVS config example:

```
##%SDNCONTROLLER1%
ovs-vsctl add-br %BRIDGE%

ovs-vsctl add-port %BRIDGE% vxlan1 -- set interface vxlan1 type=vxlan mtu_request=1550
options:key=%KEY% options:remote_ip=%SDNCONTROLLER2%

##%SDNCONTROLLER2%
ovs-vsctl add-br %BRIDGE%
ovs-vsctl add-port %BRIDGE% %INTERFACE%

ovs-vsctl add-port %BRIDGE% vxlan1 -- set interface vxlan1 type=vxlan mtu_request=1550
options:key=%KEY% options:remote_ip=%SDNCONTROLLER1%
```

We are connecting the meta channel trough wireguard server and client to form an VXLAN tunnel between our locations.

Pseudo virtual interface example:

Note: In this case for RedHat based OS

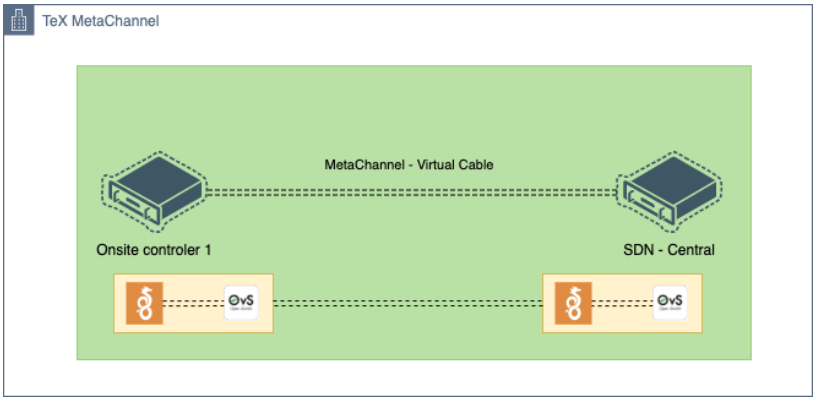
```
##%SDNCONTROLLER1%
ovs-vsctl add-port %BRIDGE% %PSEUDOINTERFACE%
nmcli con mod %PSEUDOINTERFACE%ipv4.addresses 10.10.100.200/24
nmcli con mod %PSEUDOINTERFACE%ipv4.method manual
nmcli con mod %PSEUDOINTERFACE%connection.autoconnect yes
nmcli con up %PSEUDOINTERFACE%

##%SDNCONTROLLER2%
ovs-vsctl add-port %BRIDGE% %PSEUDOINTERFACE%
nmcli con mod %PSEUDOINTERFACE%ipv4.addresses 10.10.100.100/24
nmcli con mod %PSEUDOINTERFACE%ipv4.method manual
nmcli con mod %PSEUDOINTERFACE%connection.autoconnect yes
nmcli con up %PSEUDOINTERFACE%
```

The meta channel zone

The meta channel is by default its own zone which has the highest security standards.
It is used to exchange the openflow, metrics and the meta traffic in the SDN.

Wireguard
Isolated OVS bridge



Meta traffic can be used to understand the flow of the SDN or provide monitoring and alarming metrics.

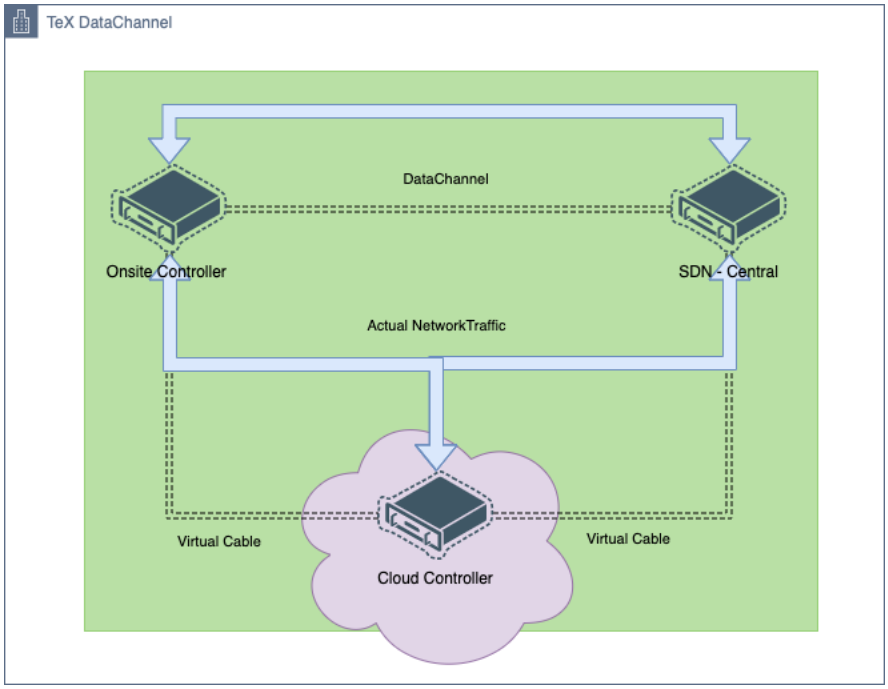
In this masked tcpdump we can see the VXLAN tags, and the traffic exchanged within the Virtual Cable.

```
MASKED.20.38292 > MASKED.14.6654: Flags [P.], cksum 0x5fca (correct), seq 1:465, ack 24,
win 502, options [nop,nop,TS val 2449929567 ecr 3737203347], length 464
03:26:40.803492 IP (tos 0x0, ttl 64, id 62947, offset 0, flags [DF], proto UDP (17), length 102)
    MASKED.14.39889 > MASKED.vxlan: [no cksum] VXLAN, flags [I] (0x08), vni 103
IP (tos 0x0, ttl 64, id 16936, offset 0, flags [DF], proto TCP (6), length 52)
    MASKED.14.6654 > MASKED.21.40690: Flags [.], cksum 0x350a (correct), seq 24, ack 465,
win 498, options [nop,nop,TS val 54885677 ecr 174125455], length 0
03:26:40.803518 IP (tos 0x0, ttl 64, id 63529, offset 0, flags [DF], proto UDP (17), length 102)
    MASKED.41118 > MASKED.12.vxlan: [no cksum] VXLAN, flags [I] (0x08), vni 102
```

The data channel zone

The actual data, like VLAN`s, ARP information or L3 routing traffic is included in the data channel.
Which has by default not a real way of being secure (encryption) other than the Virtual Cable.

For obvious reasons you should make sure that each server, application or traffic participant is in charge of encrypting its own traffic.



The other traffic

“The other network traffic” could be called “your usual network traffic”.
I will give insight into TeX-Cryptography Monitoring.

- TeX-Cryptography Monitoring is based on prometheus, alertmanager and grafana
- We use alertmanager for alarming on signal messenger
- We use grafana for visualization and alarming in metrics
- We secure the prometheus pull scrubbing with a self-written GO-proxy (TeXprox)

TeXprox

We highly utilize reverse proxy technology to secure everything in our data channel.
In the case of prometheus we secure the node_exporter by listening on 127.0.0.1 and reverse proxying the traffic with the TeXprox from every prometheus instance running.

TeXprox runs with the following settings enabled by default:

- HTTPS BasicAuth enforced
- Native GO-Lang
- TLS v1.3
- ecparam secp384r1 – sha256

Breaking the security

Everything we mentioned until now, could be broken by gaining root rights to any of the SDN controllers.
What in this case would be normal to result in access into flow tables and related encryption key infrastructure.

Chances

To minimize such risks, search at duckduckgo how to secure root access.
The following suggestions will help you to keep a certain standard of security.

- Use ed25591 ssh keys, they are good! You find discussion about that topic online.
- Use age (password or sshkey ed25591 crypto), not gpg, if it really matters, because the “standard” gpg is not “allowing” to use ECDH, ECDSA, EDDSA, thus it is broken in our opinion.
    - Or at least use a gpg version which was “repaired” so you can use ECDH, ECDSA, EDDSA.
    - gpg (GnuPG) 2.3.2
    - libgcrypt 1.9.4
    - Use with –expert flag

- Use a password manager, like gopass in age backend mode.
- Be sure to deny ssh root access.
- Be sure to remove the root password or generate a random one and delete /root/.ssh/authorized_keys
- Remove root SSH keys, there is no need for it.
- Use automation, like cloud-init to generate new SSH server keys and params for each new server (ansible, terraform, you name it).
- In case you use KVM for VM, always use luks qcow2 image formats. This format encrypts the content of the VM “hdd”.

- For hardware server connections, consider using macsec. We know, it is complicated but in our opinion it is worth it.

- Use onion layer security, whenever possible (Core layer can reach into the outer layers, but outer layers cannot reach back).

- Secure all your internal network communications, all of them! Like we do with TeXprox.
    - There are other good reverse proxies available like nginx and envoy.

- Use cryptographic proof for everything, E-Mail (S/MIME, gpg [ed25591]), Keybase. It is trustful for you and others.
    - https://keyoxide.org/4B34D927286A87810D39E14EEF725D594766969D
    - https://keybase.io/patrickdch

- For cloud environment use their respective encryption-key-infrastructure (AWS: KMS).
- Always use MFA, even if it is not proven to be super effective, use it! It raises the level of complexity.
- Zero trust is great, but don’t use it with SDN, at least not Faucet controller. It will eventually break the consistence of the network (Imagine everyone could enter your meta channel zone). 🙃

- And finally, take your time to understand what security in large scale really means, because it starts with you and your habits.

- For example, WhatsApp is easy and convenient, but better use Signal, Matrix or Threema. These are invented with certain security in mind. 🧐

Thoughts and Considerations

We think that you should provide a high degree of security trough encryption wherever and whenever it is possible.

The prior years have shown that there is indeed a need to secure communications, data transfers, your personal details and much more.

Which should be a personal right in our opinion.

If you are a service provider of any sorts – your data should be secure.

Securing data or connections is no black magic and brings none or at least a little overhead in your daily DevOps/SRE routine.

In fact, most of the security products available on the market will not be needed, if we provide a maximum secure service.

Will there be a need to over secure your endpoints or webservices?

In case the SDN is maximum secure, you don’t need to think about the internal network.

For example, the TeXprox was invented out of necessity, to secure internal communications between prometheus and the node_exporter.

Prometheus is not easy to configure once it is used inside containers.

Honestly, a reverse proxy is not a solution for everyone, in our case it’s easy to scale, does what it is supposed to do and is easy to use.

Notes

1. This publication is meant to give insight into the complex technology of SDN.

Acknowledgements

Many thanks to the contributors to this techpaper listed below who provided significant input for this paper.

Shabin Shanmugalingam @linkedIn
Patrick Dombrowski @linkedIn
Anzhela Dombrowski @linkedIn, for fixing our genius English language.

Legal statement Techpaper