Superstes WIKI

Superstes

Nov 12, 2023

CONTENTS

Tip: If you find some discrepancy or missing information - open a GitHub issue

Was this information useful? Then star the repository on GitHub

Knowledge for everyone.

CHAPTER

ONE

IT-INFRASTRUCTURE

Tip: If you find some discrepancy or missing information - open a GitHub issue

Was this information useful? Then star the repository on GitHub

1.1 IT-Infrastructure

Tip: If you find some discrepancy or missing information - open a GitHub issue

Was this information useful? Then star the repository on GitHub

Warning: It has been some time since I've set-up a system this way. (*as of writing this*)

The information be missing some details.

1.1.1 GRUB

Redundant boot disks

EFI boot does not yet support software-raid (MD) => see: Debian documentation - UEFI Grub

After installing the system with a single boot-partition, we will have to reinstall it on the second one!

Disk

You will have to use a second disk to get redundancy.

Create a boot partition at its beginning and mark it as bootable! 512MB should be enough.

Reinstall

See also Debian documentation - Grub reinstall

First we will have to boot from a live-system! Example Debian Live-system

After that we can install grub on the second disk:

```
# sda is the new disk
mount /dev/sda3 /mnt
mount /dev/sda2 /mnt/boot
mount /dev/sda1 /mnt/boot/efi
mount --rbind /dev  /mnt/dev
mount --rbind /proc /mnt/proc
mount --rbind /sys  /mnt/sys
chroot /mnt
grub-install /dev/sda --efi-directory=/boot/efi --target=x86_64-efi
```

Make sure to enable both of the disks in the UEFI/BIOS boot sequence.

Sync

After installing redundant boot-partitions - we still have a problem when doing a system update.

It will only update the kernel version on the currently active partition!

To fix this we can:

- Mount both boot partitions in the host system

```
# find disk ID's
ls -l /dev/disk/by-id/

cat /etc/fstab
> ...
> /dev/disk/by-id/wwn-0x5001b444a60ff504-part1                       /boot/efi ␣
↪vfat defaults,noatime,nofail 0 2
> /dev/disk/by-id/wwn-0x5001b444a60ff504-part2                       /boot     ␣
↪ext4 defaults,nofail 0 1
> /dev/disk/by-id/ata-SanDisk_X400_M.2_2280_256GB_170839425792-part1 /boot2/efi␣
↪vfat defaults,noatime,nofail 0 2
> /dev/disk/by-id/ata-SanDisk_X400_M.2_2280_256GB_170839425792-part2 /boot2    ␣
↪ext4 defaults,nofail 0 1
```

- Add a sync script: (*/usr/local/sbin/grub_sync.sh*)

```
#!/bin/bash
set -euo pipefail

PATH_BAK='/var/backups/boot'
RETENTION_DAYS=30

if mount | grep "on /boot type" -q && mount | grep "on /boot2 type" -q
then
  mkdir -p "$PATH_BAK"
```

(continues on next page)

(continued from previous page)

```
  echo '### REMOVING OLD BACKUPS of /boot2'
  find "${PATH_BAK}/" -mtime +${RETENTION_DAYS} -name "*.tar.gz" -type f  # to show␣
↪the files to be deleted
  find "${PATH_BAK}/" -mtime +${RETENTION_DAYS} -name "*.tar.gz" -type f -delete

  echo '### BACKING-UP current /boot2'
  tar -czf "${PATH_BAK}/$(date '+%Y-%m-%d_%H-%M-%S').tar.gz" /boot2/ 2>/dev/null

  echo '### SYNCING /boot to /boot2'
  rsync -av --delete /boot/ /boot2 --exclude "lost+found"
else
  echo 'Missing at least one boot-partition in mounts!'
  exit 1
fi
```

- Add sync job

```
crontab -e

# sync boot-partitions daily
0 0 * * * /bin/bash /usr/local/sbin/grub_sync.sh
```

Tip: If you find some discrepancy or missing information - open a GitHub issue

Was this information useful? Then star the repository on GitHub

1.1.2 LVM

Intro

LVM is a disk-management system that adds some nice functionality in comparison to using bare partitions.

It creates a layer between the physical disks and partitions.

This allows you to:

- Work around the common problem of resizing partitions that are not in the last position on the physical disk
- Creation of volume snapshots
- Thin-Provisioned volumes
- Options to configure RAID
- Implementation of write caches

More information can be found in the RedHat documentation!

Usage

Navigation

How to get the status and information of your current setup.

Overview

The ‘lsblk’ command shows you an overview of your current storage configuration.

```
root@superstes:~# lsblk
> NAME          MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
> sda             8:0    0   30G  0 disk
> |-sda1          8:1    0  487M  0 part /boot
> |-sda2          8:2    0    1K  0 part
> `-sda5          8:5    0 29.5G  0 part
>   |-vg0-root  254:1    0  9.7G  0 lvm  /
>   |-vg0-var   254:2    0  9.7G  0 lvm  /var
>   `-vg0-swap  254:3    0  1.9G  0 lvm  [SWAP]
> sdb             8:16   0  100G  0 disk
> `-vg1-data    254:0    0   50G  0 lvm  /mnt/data
```

Physical volumes

These commands show you how LVM sees your physical disks.

```
root@superstes:~# pvs
> PV         VG  Fmt  Attr PSize    PFree
> /dev/sda5  vg0 lvm2 a--   <29.52g  <8.35g
> /dev/sdb   vg1 lvm2 a--  <100.00g <50.00g

root@superstes:~# pvdisplay
# prettified
> --- Physical volume ---
> PV Name               /dev/sdb
> VG Name               vg1
> PV Size               100.00 GiB / not usable 4.00 MiB
> Total PE              25599
> Free PE               12799
> Allocated PE          12800
>
> --- Physical volume ---
> PV Name               /dev/sda5
> VG Name               vg0
> PV Size               29.52 GiB / not usable 2.00 MiB
> Total PE              7557
> Free PE               2137
> Allocated PE          5420
```

That can be useful if you resize the (*virtual*) physical disk and are not sure if LVM realized the changes.

Volume groups

These commands show you the existing Volume Groups.

```
root@superstes:~# vgs
> VG  #PV #LV #SN Attr   VSize    VFree
> vg0   1   3   0 wz--n-  <29.52g  <8.35g
> vg1   1   1   0 wz--n- <100.00g <50.00g

root@superstes:~# vgdisplay
# prettified
> --- Volume group ---
> VG Name               vg1
> VG Access             read/write
> VG Status             resizable
> MAX LV                0
> Cur LV                1
> Open LV               1
> Max PV                0
> Cur PV                1
> Act PV                1
> VG Size               <100.00 GiB
> Alloc PE / Size       12800 / 50.00 GiB
> Free  PE / Size       12799 / <50.00 GiB
>
> --- Volume group ---
> VG Name               vg0
> VG Access             read/write
> VG Status             resizable
> VG Size               <29.52 GiB
> Alloc PE / Size       5420 / 21.17 GiB
> Free  PE / Size       2137 / <8.35 GiB
```

Logical Volumes

These commands show you the existing Logical Volumes.

```
root@superstes:~# lvs
> LV   VG  Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
> root vg0 -wi-ao---- <9.66g
> swap vg0 -wi-ao---- <1.86g
> var  vg0 -wi-ao---- <9.66g
> data vg1 -wi-ao---- 50.00g

root@superstes:~# lvdisplay
# prettified
> --- Logical volume ---
> LV Path                /dev/vg1/data
> LV Name                data
> VG Name                vg1
> LV Write Access        read/write
> LV Status              available
```

(continues on next page)

(continued from previous page)

```
> LV Size                50.00 GiB
>
> --- Logical volume ---
> LV Path                /dev/vg0/root
> LV Name                root
> VG Name                vg0
> LV Size                <9.66 GiB
>
> --- Logical volume ---
> LV Path                /dev/vg0/var
> LV Name                var
> VG Name                vg0
> LV Size                <9.66 GiB
>
> --- Logical volume ---
> LV Path                /dev/vg0/swap
> LV Name                swap
> VG Name                vg0
> LV Size                <1.86 GiB
```

Create

OS Setup

If you set-up Ubuntu or Debian you need to create your LVM-config using the graphical setup!

Once your main partitions are placed on non-LVM volumes - it is hard to move them.

It might be easier to just reinstall the machine.

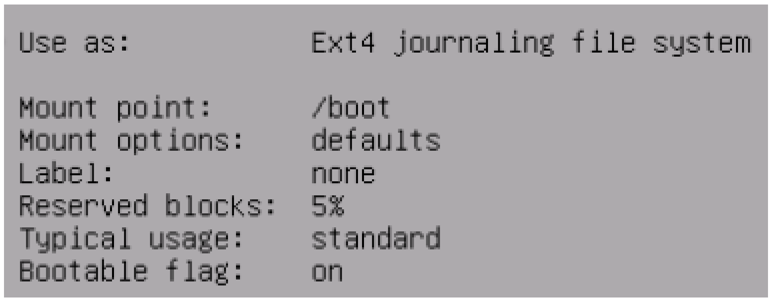
Bootable

You might need to create your **boot-partition 'outside' the LVM**!

I found that EFI boot did not really work when put inside a LVM volume.

Therefor you might want to create a 512MB primary partition at the begin of your disk-layout that acts as 'EFI bootable' or '/boot'.

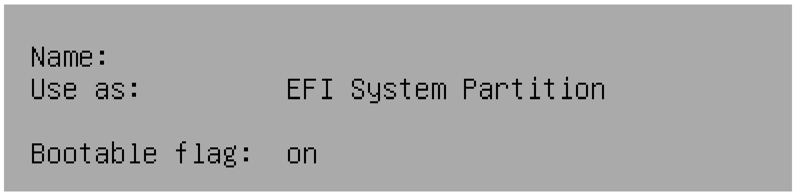
Example for BIOS:



```
Use as:             Ext4 journaling file system

Mount point:        /boot
Mount options:      defaults
Label:              none
Reserved blocks:    5%
Typical usage:      standard
Bootable flag:      on
```

Example for UEFI:



```
Name:
Use as:         EFI System Partition

Bootable flag:  on
```

Add physical volume

1. Add physical volume
2. Make sure LVM recognized the new disk:

   ```
   pvscan
   ```

   Remember the name of your new disk.
3. Create a new volume group or add the physical volume to an existing one:

   ```
   # create new one
   vgcreate <NAME-OF-NEW-VOLUME-GROUP> /dev/<NAME-OF-NEW-DISK>
   #  example: vgcreate vg1 /dev/sdb

   # add to existing one (ADVANCED USAGE)
   vgextend <NAME-OF-EXISTING-VOLUME-GROUP> /dev/<NAME-OF-NEW-DISK>
   #  example: vgextend vg0 /dev/sdb
   ```
4. Create a new logical volume - if needed:

   ```
   lvcreate -n <NAME-OF-NEW-VOLUME> -L <SIZE-OF-NEW-VOLUME> <NAME-OF-VOLUME-GROUP>
   #  example: lvcreate -n data -L 20G vg1
   ```
5. Create file-system: (*'ext4' in this case*)

   ```
   mkfs.ext4 /dev/mapper/<NAME-OF-VOLUME-GROUP>-<NAME-OF-NEW-VOLUME>
   #  example: mkfs.ext4 /dev/mapper/vg1-data
   ```
6. Mount volume:

   ```
   # 1. create mount directory
   mkdir -p /<PATH-TO-MOUNTPOINT>
   #  example: mkdir -p /mnt/data

   # 2. mount permanently
   # 2.1. edit fstab-config
   sudo nano /etc/fstab

   # 2.2. append line
   /dev/mapper/<NAME-OF-VOLUME-GROUP>-<NAME-OF-NEW-VOLUME> /<PATH-TO-MOUNTPOINT>
   ↪<NAME-OF-FILESYSTEM> defaults 0 2
   #  example: /dev/mapper/vg1-data /mnt/data ext4 defaults 0 2

   # 2.3. save and exit
   ```

(continues on next page)

(continued from previous page)

```
# 2.4. mount
sudo mount -a

# check if volume was mounted correctly (yes - if shown in output)
mount | grep "/<PATH-TO-MOUNTPOINT>"
#  example: mount | grep "/mnt/data"
```

Resize

Warning: If you don't know what you are doing => you should not make changes like these on an important system!

You might **BREAK YOU SYSTEM**!!

Try it on an useless VM and play around with it or leave it to the pros.

We need to go through these steps:

1. Extend the physical drive or partition

   1.1. Drive

   You might need to resize the disk in your virtual environment/hypervisor.

   1.2. Partition - entering fdisk

   ```
   # start fdisk targeting the modified disk
   fdisk /dev/<NAME-OF-DISK>
   #  example: fdisk /dev/sdb

   # show current partition layout
   p
   ```

   1.3. Partition - direct vs nested

   Sometimes the target partition might be encapsulated inside an 'extended' partition.

   It might look like this:

   ```
   fdisk -l /dev/sda
   > Device     Boot   Start      End  Sectors  Size Id Type
   > /dev/sda1  *       2048   999423   997376  487M 83 Linux
   > /dev/sda2       1001470 25163775 24162306 11.5G  5 Extended
   > /dev/sda5       1001472 25163775 24162304 11.5G 8e Linux LVM
   ```

   In that case you will have to delete and re-add both of these partitions to extend them.

   ```
   # delete partition you want to extend
   d
   => number of partition

   # delete the extended partition
   ```

(continues on next page)

(continued from previous page)

```
d
=> number of partition

# re-create the extended partition
n
=> choose 'extended'
e
=> enter or choose custom partition number
=> enter
=> enter

# re-create the target partition
n
=> enter
=> enter
=> remove the Signature?
n  # else your LVM config will be gone

# modify partition type
t
=> enter partition number
8e  # for LVM disk

# verify the layout is the same as before (except being bigger)
p

# save and write
w

# example:
fdisk /dev/sda
d ENTER 5 ENTER
d ENTER 2 ENTER
n ENTER e ENTER 2 ENTER ENTER ENTER
n ENTER ENTER n ENTER
t ENTER 5 ENTER 8e ENTER
w
```

If that is not the case it is a little easier:

```
# delete partition you want to extend
d
=> number of partition to increase

# re-create the partition
n
=> enter or choose custom partition number
=> enter
=> enter
=> remove the Signature?
n  # else your LVM config will be gone

```

(continues on next page)

(continued from previous page)

```
# modify partition type
t
=> enter partition number
8e  # for LVM disk

# verify the layout is the same as before (except being bigger)
p

# save and write
w

# example:
fdisk /dev/sda
d ENTER 2 ENTER
n ENTER p ENTER 2 ENTER ENTER ENTER n ENTER
t ENTER 2 ENTER 8e ENTER
w
```

2. Resize the LVM physical volume

```
pvresize /dev/sdX
```

3. Extend the LVM volume group

```
vgextend vg0 /dev/sdX
```

4. Extend the LVM logical volume

```
lvextend /dev/vg0/lv1 -L 20GB
```

5. Update the partition size

```
resize2fs /dev/mapper/vg0-lv1
```

```
#!/bin/bash
PD='sda'
LVM_PV="${PD}2"
LVM_VG='vg0'
LVM_LV='lv1'
EXT='20GB'

fdisk "/dev/${PD}"
pvresize "/dev/${LVM_PV}"
vgextend "${LVM_VG}" "/dev/${LVM_PV}"
lvextend "/dev/${LVM_VG}/${LVM_LV}" -L "${EXT}"
resize2fs "/dev/mapper/${LVM_VG}-${LVM_LV}"
```

Tip: If you find some discrepancy or missing information - open a GitHub issue

Was this information useful? Then star the repository on GitHub

1.1.3 Web Application Firewalling

First we'll go through some basic knowledge and after that we'll look into **practical** log analysis.

Level of attacks

There are some different levels of attacks you will encounter:

1. **Bad crawlers**

   Gathering information from your site for some unknown use-case using GET/HEAD.

   You might be able to block them by implementing *bot-detection* and *crawler verification*.

2. **Probes**

   Checking your site for known exploits that are far-spread or common data-/information-leaks.

   They only use GET/HEAD.

   You might be able to block them by creating a simple ruleset from the information you could gather by analyzing the request logs.

3. **Dumb scripts**

   This one might try to exploit single functions behind an URL or submit unprotected forms.

   Attackers will not change the user-agent of their http-client or set it to none.

   They might already use a pool of IP-addresses.

   You can easily block them by creating a list of bad user-agents.

4. **Better script**

   Also might try to exploit single functions behind an URL or submit unprotected forms.

   Attackers change their user-agent and use a pool of IP-addresses.

   Sometimes they still use a static user-agent or show the same major browser version but with different minor versions and operating system versions. (*p.e. Chrome 103.x.x from Windows NT 6.x-10.x*)

   The TLS-fingerprint will always be the same as the actual http-client does not change.

   They might also forget to set a referer or some *accept* headers that are set by all mainstream browsers nowadays.

   Client-side fingerprinting will catch them.

5. **Advanced bot tools**

   There are some bot-tools out there that can mimic a default browser in nearly every aspect.

   It's a continuous battle between them and client-side fingerprinting to fight for/against detection.

   These kind of attacks might be detectable by abnormal traffic patterns *related to GeoIP country/ASN*.

   Most unofficial botnets use hijacked servers to proxy their requests. If you are able to find out that user-actions are performed by clients that originate from hosting-/datacenter-IPs you might be onto something.

   You can also just make it harder to perform important actions by using captcha or some other logic. Of course - that is a fight against AI tools.

6. **Automation with real Browsers**

   To get a clean browser fingerprint some attacker may use actual mainstream browsers (with GUI) and use auto-clickers or similar tools to automate website interactions.

These can get stopped by using captcha or monitoring the mouse movement & keyboard strokes using javascript.

7. **Human bots**

   As a last resort - some might even leverage humans as 'bots'.

   Some countries have low price-points for human work-hours. An example is India.

   There are whole companies that provide such a service. You tell them what to do and they have thousands of smartphones with proxies/VPNs that can target a website.

   As the devices are actual 'users' and can handle some anti-bot logic they might be hard to fight.

   It helps if you create some alerts for web-actions (*p.e. POST on some form*) that will be triggered if a threshold is exceeded. That way you can manually check if there is some unusual traffic going on.

Fingerprinting

Fingerprinting is used to:

- identify clients across IP addresses
- give us more information to identify bad traffic patterns that might indicate an attack

Usually systems use both client-side and server-side fingerprinting.

See also: information provided by niespodd

Server-Side Fingerprint

This is most of the times implemented on your proxy/WAF or the application itself.

It is pretty straight-forward to implement, but has some major limitations as we are limited by the information we get from the client.

Therefor it alone cannot be used to get an unique fingerprint per client.

GeoIP information like country and ASN can be very useful to limit the matching-scope of such a fingerprint - if you want to lower the risk of blocking/flagging a single one. (more uniqueness, less global matching)

How can it be assembled?

- boolean values (1/0)
  - existence of headers (*referer, accept, accept-encoding, accept-language*)
  - existence of header values
  - settings inside headers
  - Is the domain inside the request or 'just' set as host-header?
  - sorting of URL-parameters or values inside headers or cookies
  - upper-/lower-/mixed-case of values
  - unusual special characters inside values
  - LF or CRLF line breaks used
  - usage of whitespace
- hashes
  - hash of some header-/cookie-value

    - of the TLS-Fingerprint

- information

    - limiting the match-scope by using the first 8-16 bits of the IP-address, GeoIP country and/or GeoIP ASN

TLS Fingerprint

The SSL/TLS fingerprint can be useful as it differs between http-clients.

Nearly each http-client software and version of it uses a specific set of TLS settings. But there are overlaps between http-clients or be forged by attackers.

The common settings used to build such a fingerprint are:

```
SSLVersion,Cipher,SSLExtension,EllipticCurve,EllipticCurvePointFormat

Example: 769,47-53-5-10-49161-49162-49171-49172-50-56-19-4,0-10-11,23-24-25,0
Example Hashes:
  * aeaf2f865911f886e3f721156a5f552e (wget)
  * ac507a278bdeca60a1c96c29fa244b81 (curl)
  * 63f63ca1aa38d95aae0be017b760408b (firefox 118-120)
```

Such a fingerprint enables us to compare it against the user-agent a client supplied. If there is an abnormality we can investigate it.

You could also create a mapping of known-good user-agents to TLS fingerprints and block/flag requests that don't match.

But for now there seems to be no public JA3 fingerprint database to compare your findings against. But I have a project like that in my backlog.. (;

See also: JA3 SSL-fingerprint

Client-Side Fingerprint

Creating a browser-fingerprint with information that is available on the client can be very useful.

We have access to much more information on the client that allows us to create fingerprints with high-precision uniqueness.

Information that might be used:

- is javascript enabled
- environment/settings
    - screen size
    - color depth/inversion
    - languages
    - timezone
    - storage preferences
    - canvas

  - fonts
  - are cookies enabled
  - audio
- operating system
- hardware
  - cpu & concurrency
  - screen resolution
  - memory
  - screen touch support
- browser
  - type and flavor
  - plugins
  - webGL

Some existing libraries: (*Note: I have not tested them*)

- (*pay-to-win*) fingerprintJS Repository, fingerprintJS example, fingerprintJS information
- creepJS Repository, creepJS example, creepJS information
- supercookie, supercookie example, supercookie Docs
- cross-browser, cross-browser example

Bot detection

You might want to flag requests that might be bots so your application can handle them differently.

You can use a boolean flag or a bot-score.

Either way you might want to block 'dumb script' bots first or flag them as such.

Script bots

How might one detect them?

- Search the user-agent for common http-clients used by scripting languages:
  - Headless browsers (*headless*)
  - Python3 libraries (*python*, scrapy, aiohttp, httpx, . . .)
  - Golang (go-http-client, . . .)
  - Javascript packages (axios, . . .)
  - Shell tools (curl, wget, . . .)
  - Powershell
  - C++ (cpp-httplib)

  - C#
  - Java (*java*)
  - Ruby (*ruby*)
  - Perl (*perl*)
  - PHP (guzzlehttp, phpcrawl, . . .)
  - Darknet crawlers (test-bot, *tiny-bot, fidget-spinner-bot, . . .)
  - No user-agent at all

This list is only scratching the surface of tools/libraries that are used in the wild. You will have to check your logs and extend the list if needed.

Actual bots

Note: This bot-check will not differentiate between 'good crawler bots' and others.

Note: If you have a client-side fingerprint implemented - you might also use that detailed information to further filter-out bots.

You might want to **flag a request as possible bot** if:

- They use known good-crawler user-agents
- They have 'bot', 'spider' or 'monitor' in their user-agent
- They have headers like 'accept', 'accept-language', 'accept-encoding' or 'referer' unset

  Note: You might encounter false-positives on the users 'entry page' if it is the first site the user opens (*no referer set*). But this is uncommon.

  Note: Old browsers (<2015) might not set the *accept* headers. But this is uncommon nowadays.

- They originate from
  - a datacenter (*GeoIP database needed*)
  - a country that is not expected to request your site (*GeoIP database needed*)
  - an ASN that is known to be used by bots: ASN spamlist, Spamhaus ASN-DROP (*GeoIP database needed*)
  - IPs that are known to be used by bots: Tor exit node IPList, Spamhaus DROP

Going further - you might want to **flag them as 'bad'** if:

- They use a common good-crawler user-agent but failed the *crawler verification*
- You can use the ASN-/IP-Lists mentioned above

Crawler verification

The large organizations that use crawlers to supply their services with information will provide you with a way to verify if a crawler, that uses their user-agent, is a legitimate one.

Most will either:

- configure their bot source-IPs with a specific reverse-DNS (PTR) record
- supply you with a way to check their source-IP with a list of IP-ranges

Examples:

- Google Bots via reverse-DNS
- Bing Bots via reverse-DNS
- Yandex Bots via reverse-DNS
- Facebook Bots via IP-List

GeoIP information

GeoIP databases can help you to identify attacks.

You might either want to implement such lookups into your proxy or log-server.

The most **interesting data** in my opinion is:

- Country
- ASN (*internet-/hosting-provider*)
- Hosting detection
- VPN/Proxy/Tor/Relay detection

You can check-out some databases:

- Maxmind Free GeoLite2-databases: GeoLite2 database download, GeoLite2 database information, Maxmind Docs
- Maxmind Paid databases: Maxmind API, Maxmind databases
- IP-Info: ipinfo.io API, ipinfo.io Docs
- IP-API: ipapi.is Docs, ipapi.is FREE databases

Analyzing Request Logs

If you want to protect a web-application from threats you will have to analyze the requests targeting it.

When analyzing request/access logs the right way, you might be able to detect 'hidden' attacks targeting the application.

I would highly recommend log-server solutions like GrayLog or Grafana Loki to have a Web-UI that enables you to deeply analyze your log-data.

Server-Side information

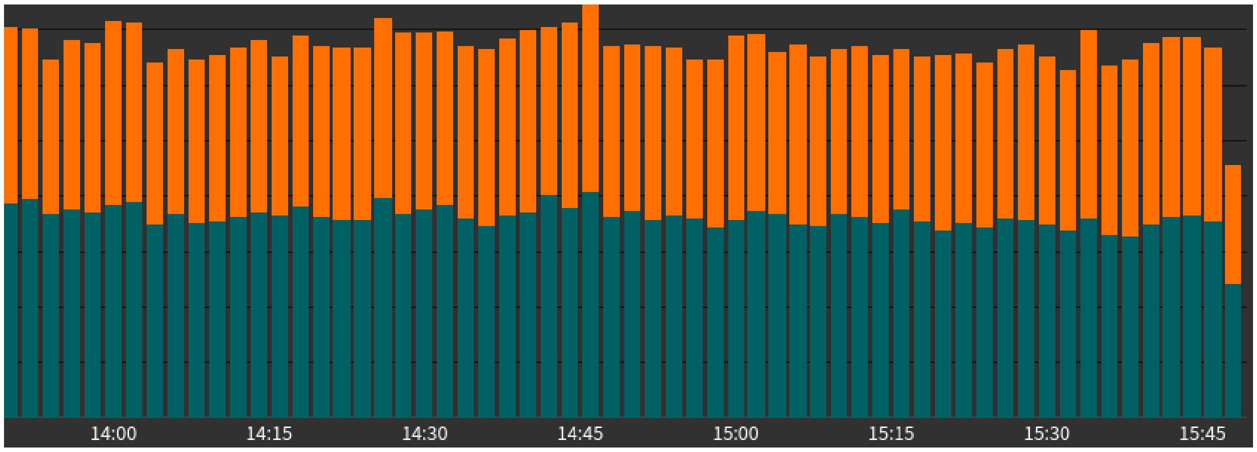
As mentioned above in the *server-side fingerprint section* - we do not have that much information available when only doing server-side inspection.

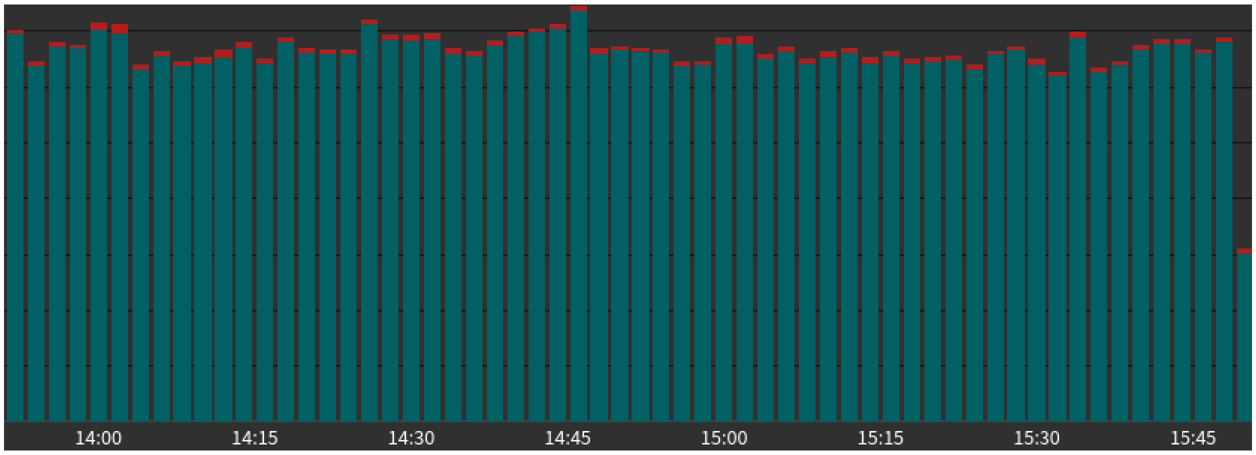
Therefore it can be very useful to implement some *GeoIP database lookups* to gain more options for analyzing the data we have.

Most times we will want to group our data by two to four attributes to visualise correlations.

Bot flagging

If you have bot-flags configured you can get a brief overview of how many bots and script-bots access your site:

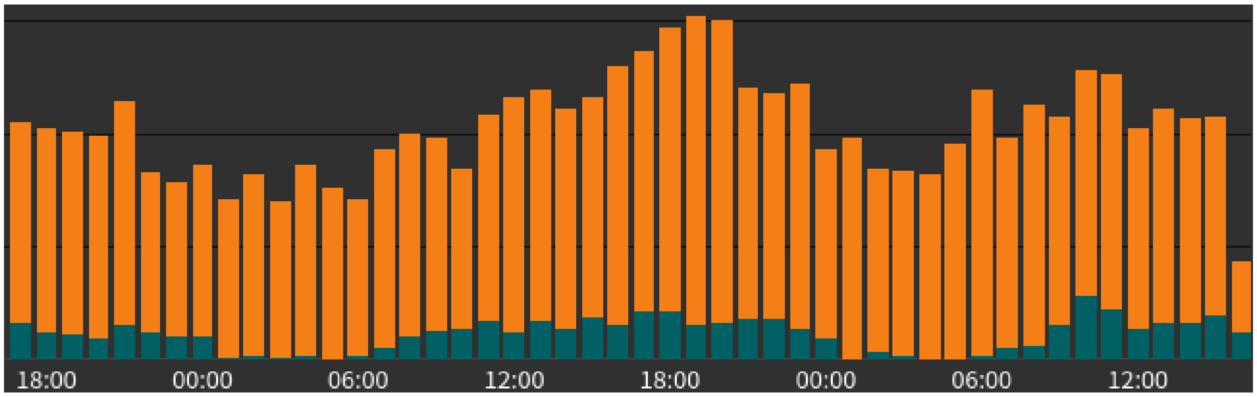




Per-path analysis

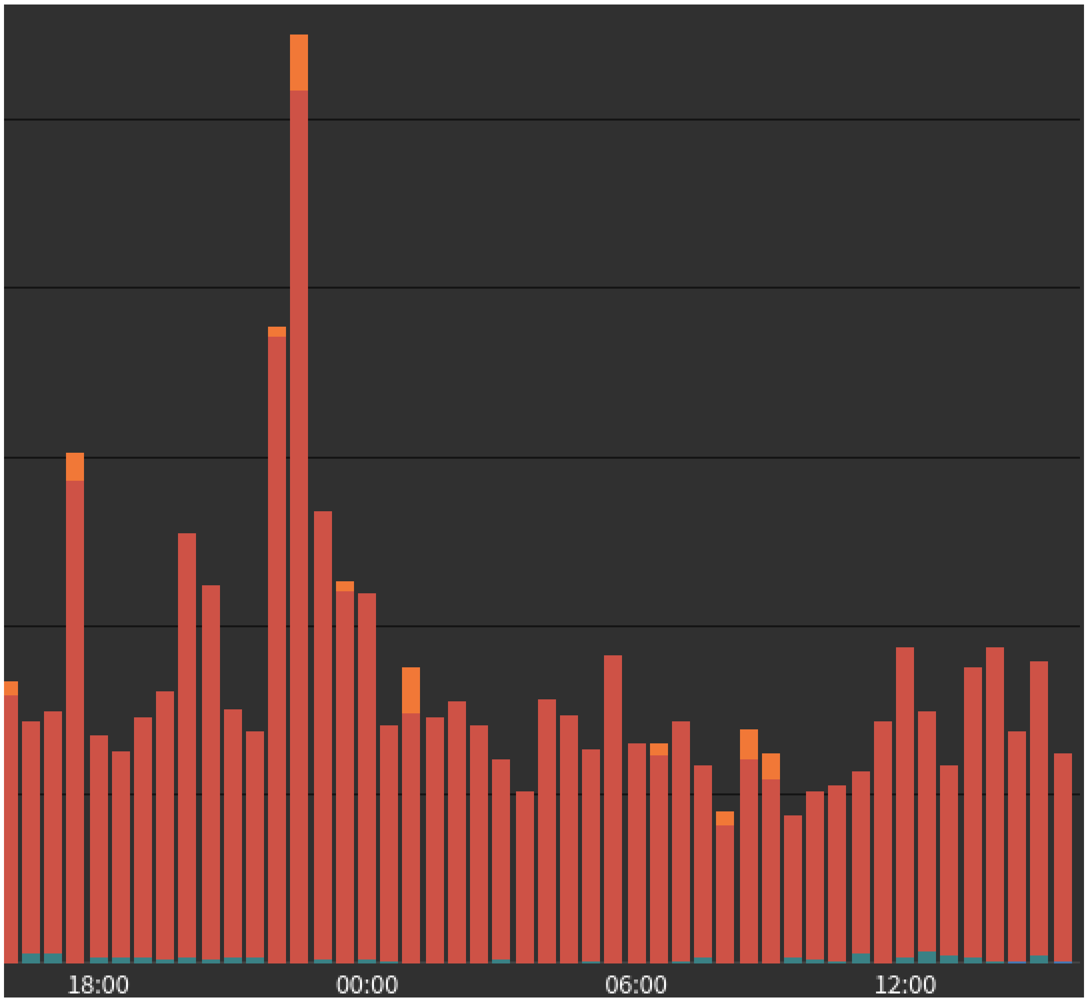
If you have specific endpoints/URLs/paths that are targeted by attack - you should filter the logs to reduce the scope of your statistics and get better results:

1. Bots that target the path using POST:

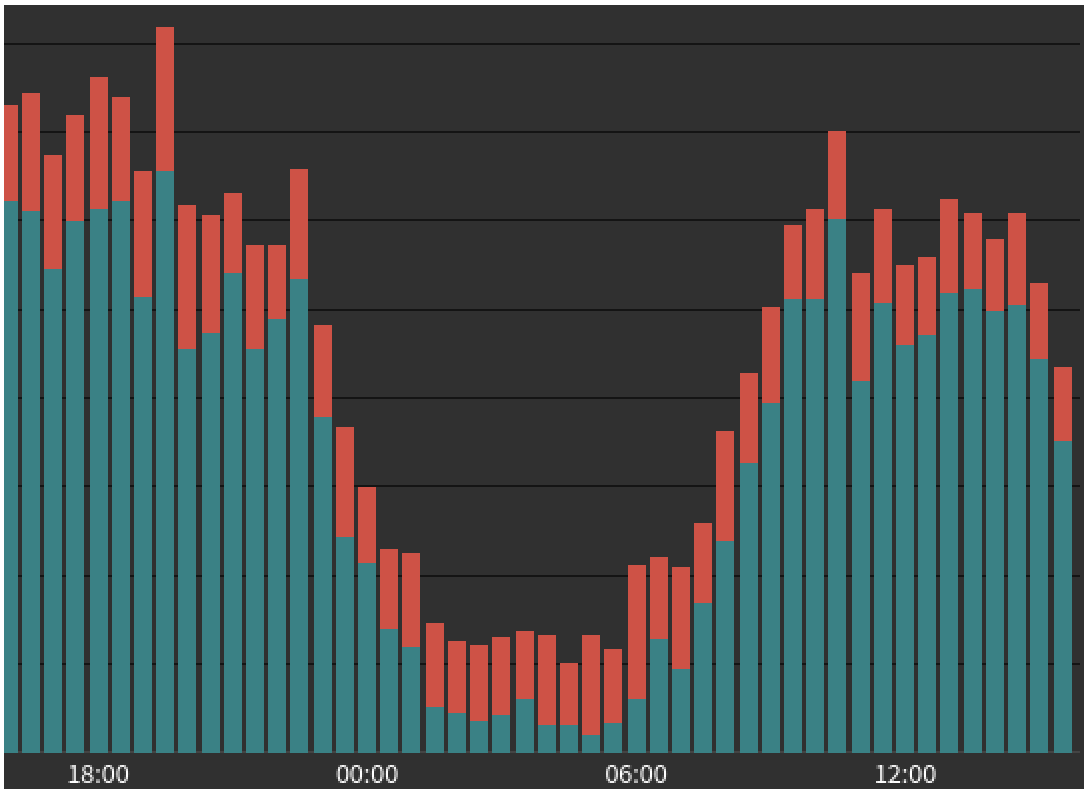


2. Status-Codes/Blocks of POST requests

To one path:



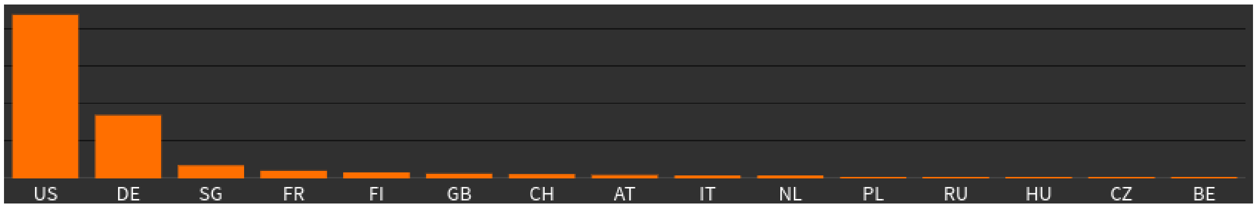
To another path:



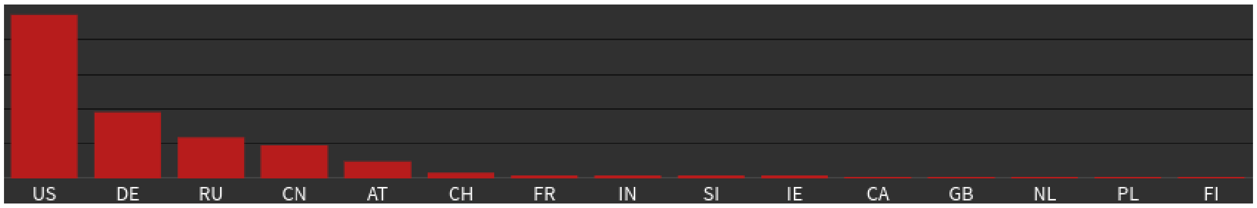
GeoIP information

You might recognize how useful such information can be.

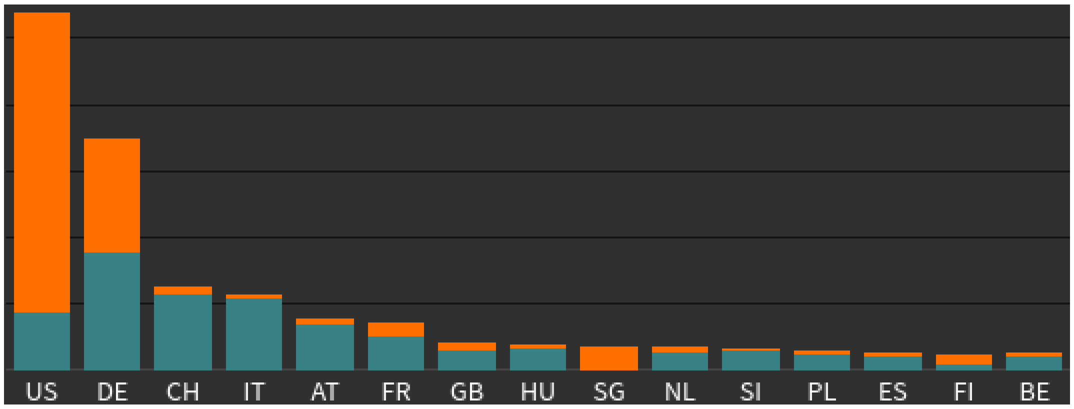
Bots by country:



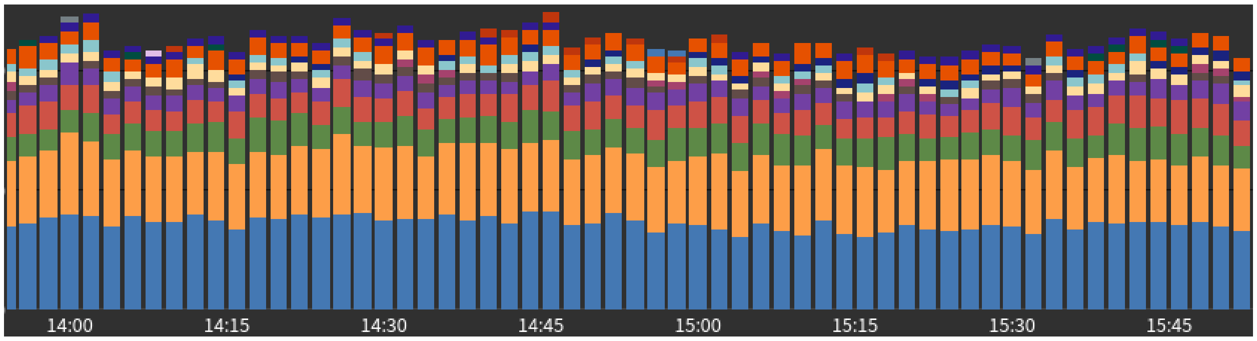
Script-Bots by country:



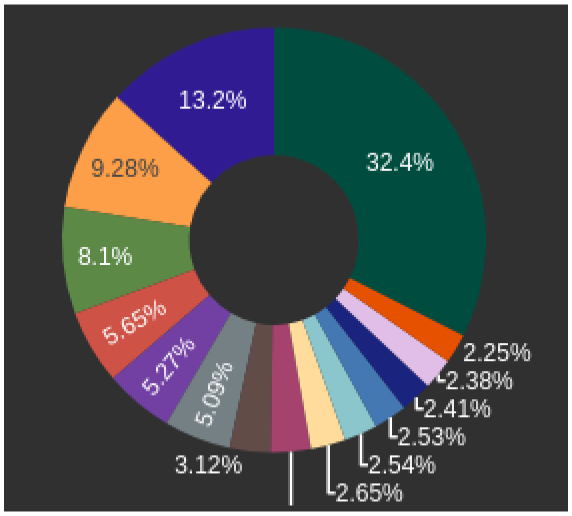
Normal requests vs bot-requests by country:



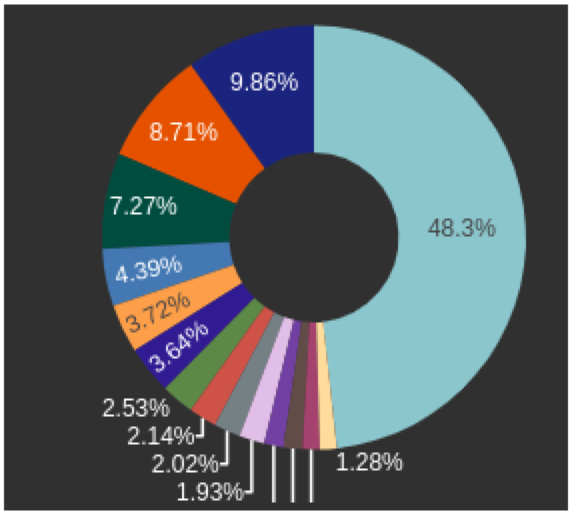
Requests by country over time:



Requests by ASN:



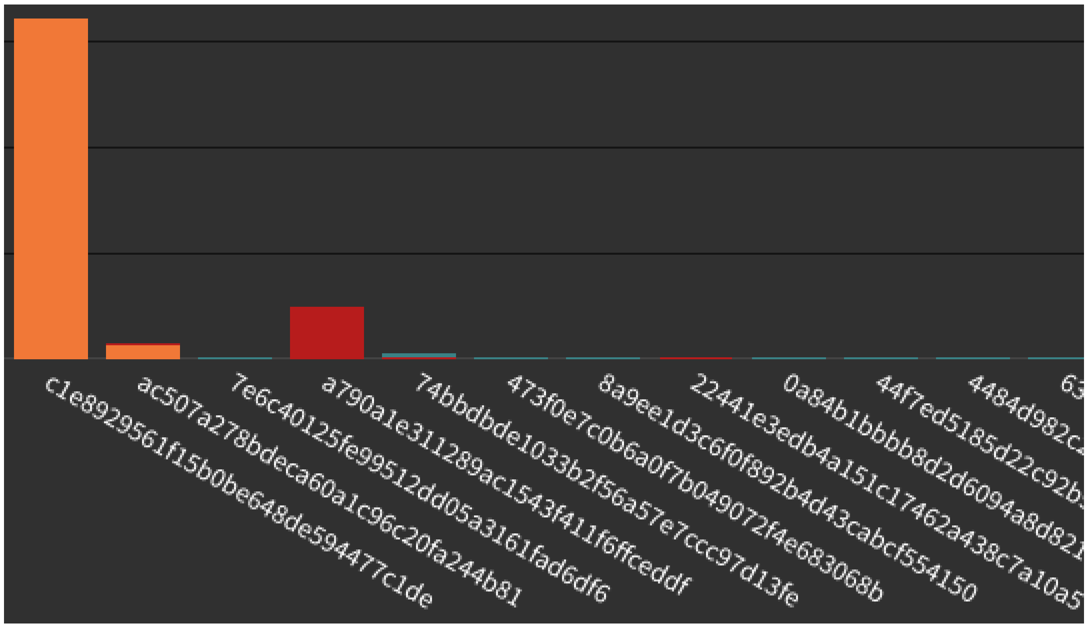
Blocks by ASN:



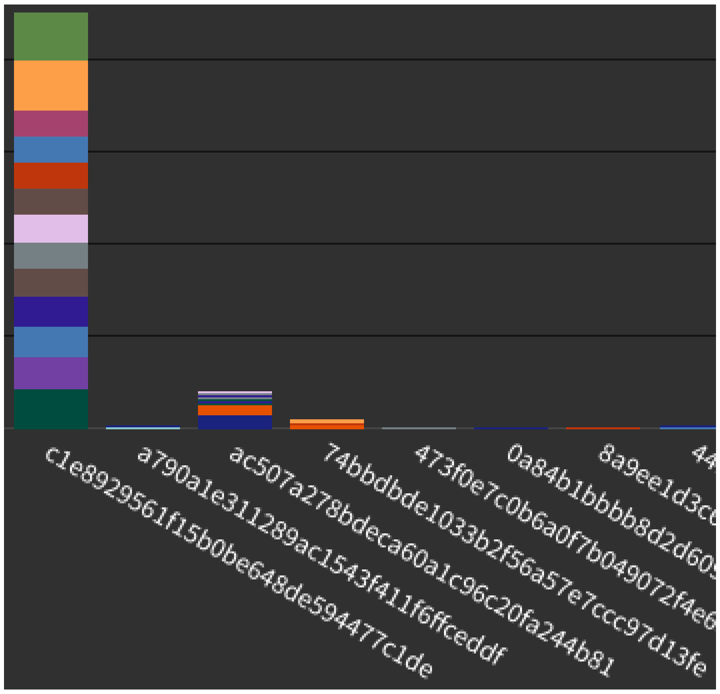
Fingerprints

Fingerprints like JA3-TLS can also be useful for analyzing traffic.

POST requests to specific path by status-code:



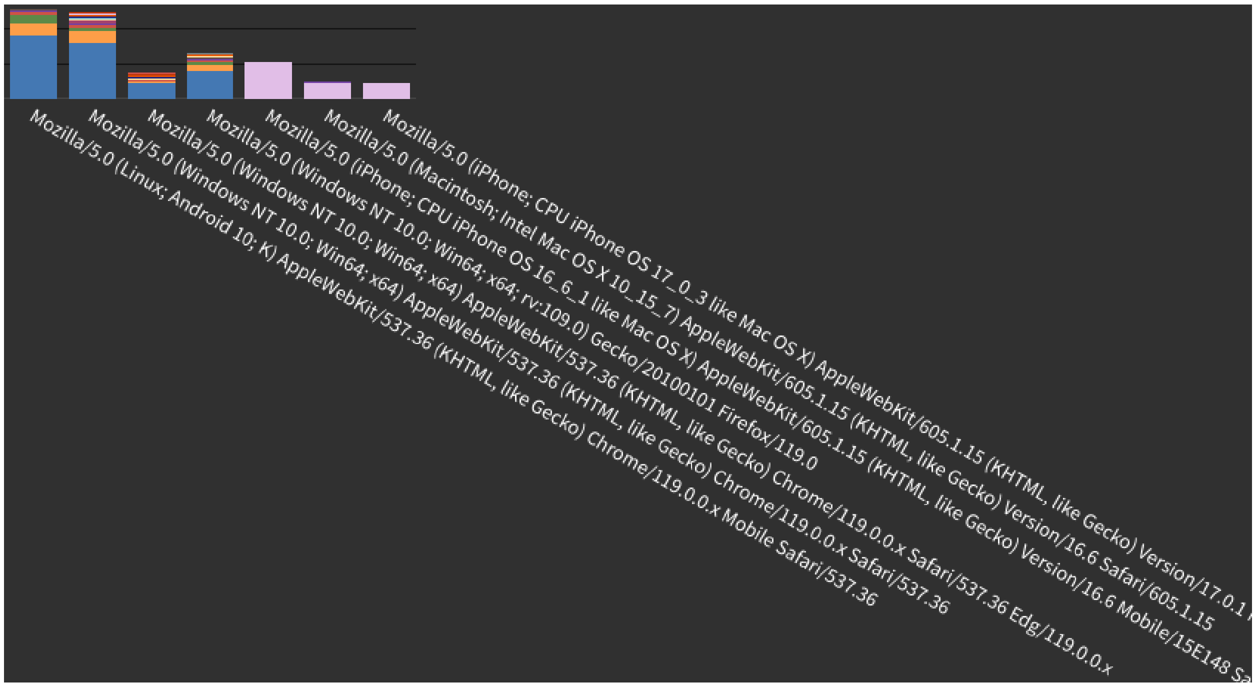
POST requests to specific path by source-network:



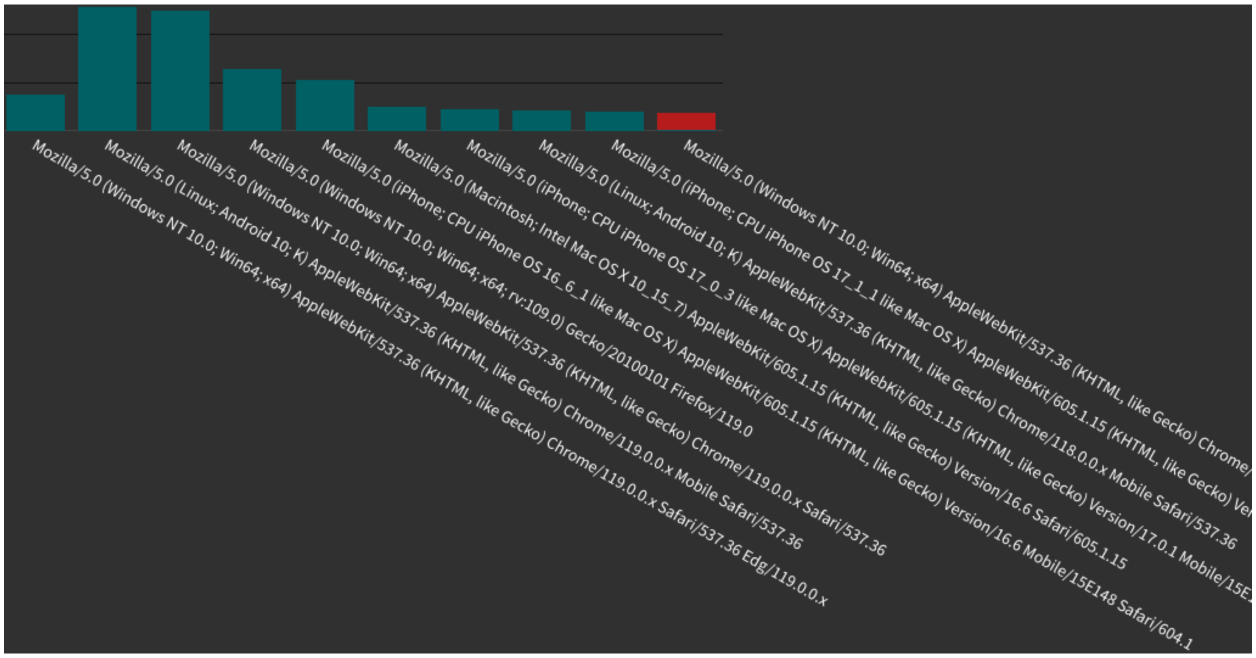
User Agents

You may also be able to find useful links by checking the user agents.

POST requests to specific path by user-agent and TLS-fingerprint.

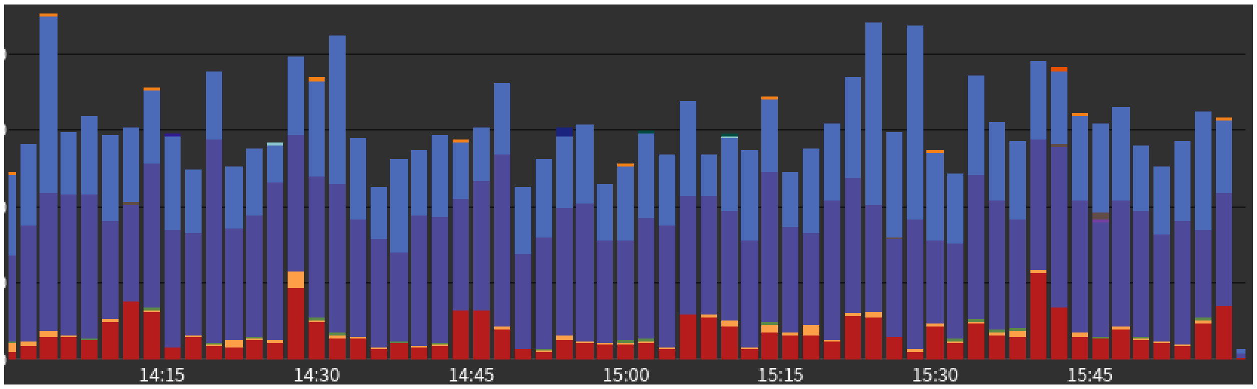


POST requests to specific path by user-agent, TLS-fingerprint and block.

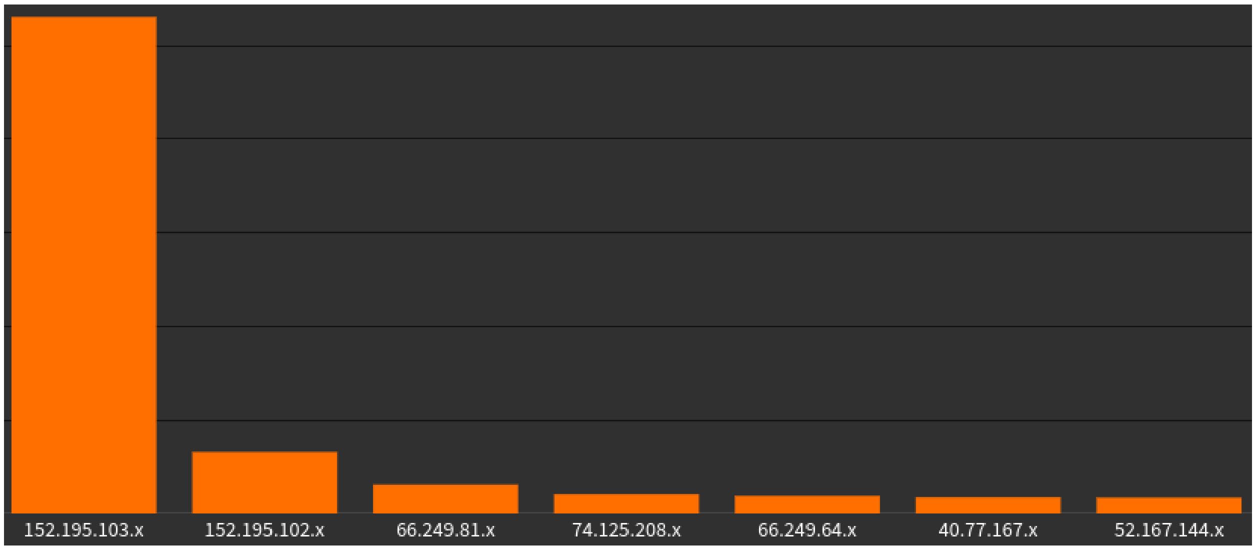


Errors & Blocks

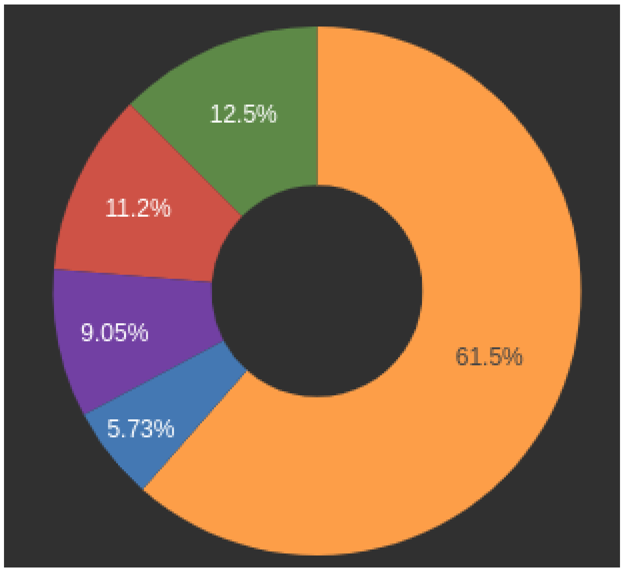
HTTP 4xx responses over time: (*red* + *orange* = *blocks*)



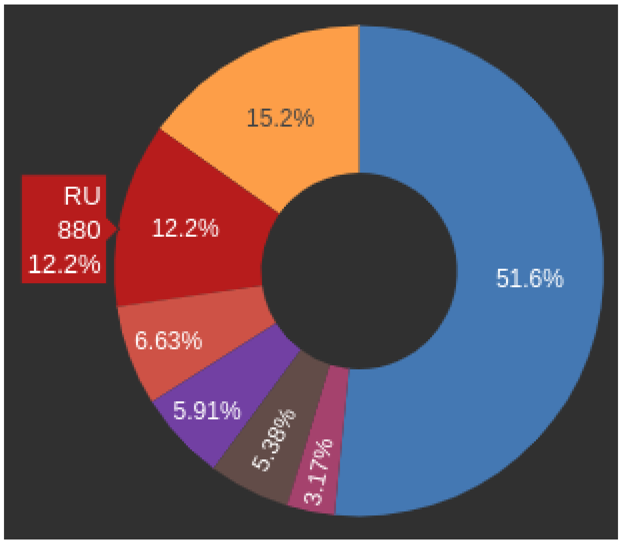
HTTP 4xx responses by source-networks:



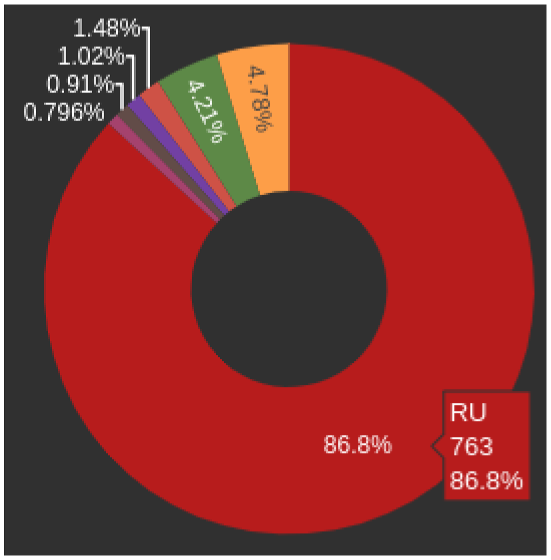
HTTP 4xx responses by ASN:



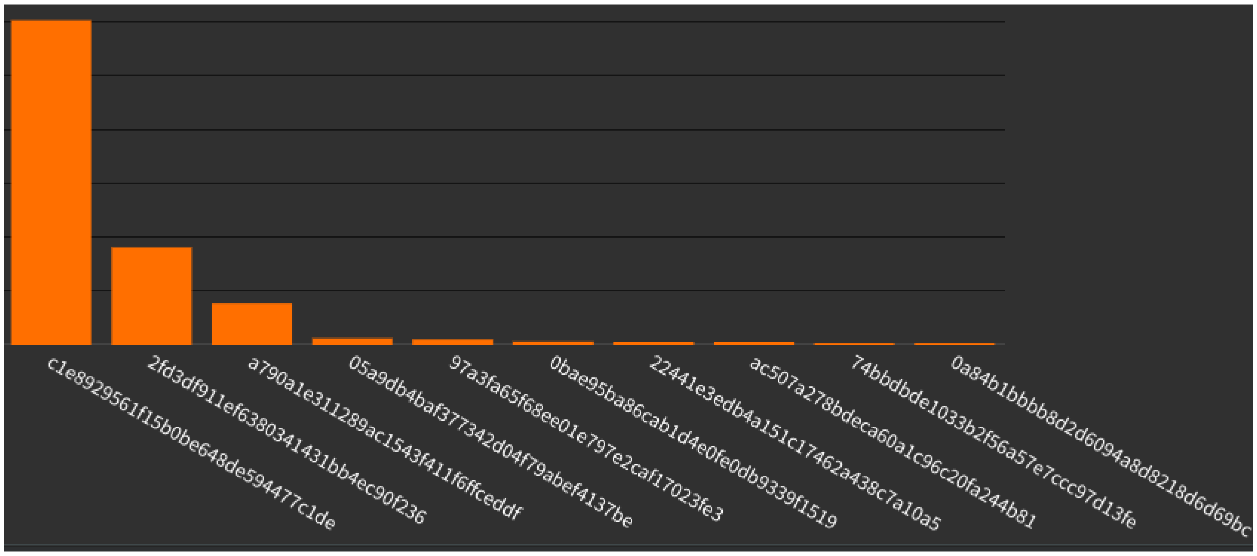
HTTP 4xx responses by country:



Blocks by country:



Blocks by TLS fingerprint:



Client-Side information

You will have to send/pass the information, gathered by Javascript on the client, to your server.

tbc..

CHAPTER

TWO

NETWORK

Tip: If you find some discrepancy or missing information - open a GitHub issue

Was this information useful? Then star the repository on GitHub

2.1 Network

Tip: If you find some discrepancy or missing information - open a GitHub issue

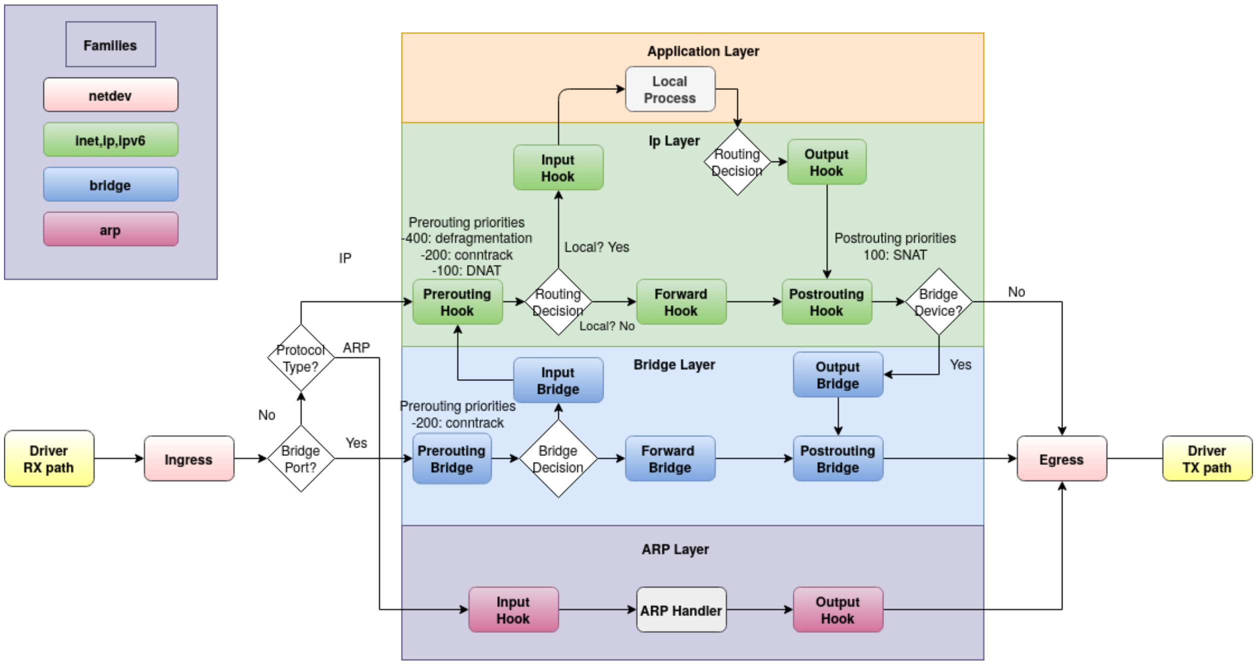
Was this information useful? Then star the repository on GitHub

Warning: Writing of this documentation is still in progress! Read it with a grain of salt!

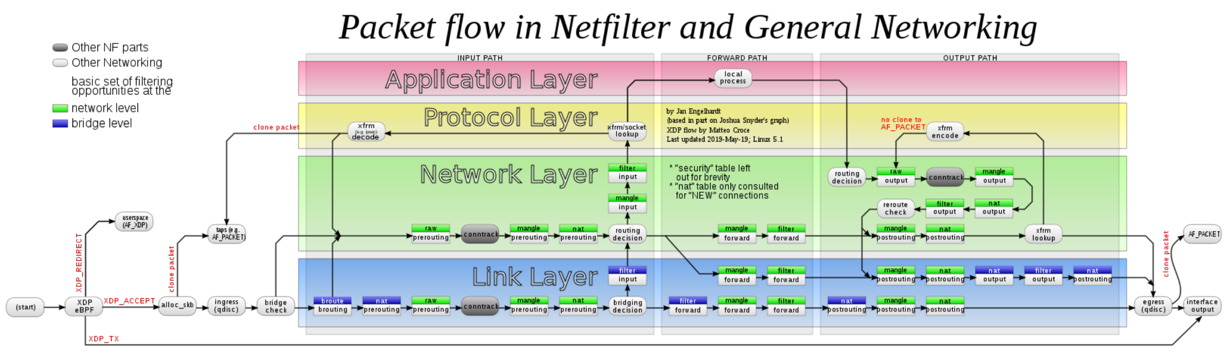
2.1.1 Firewall - NFTables

Introduction

Chain hooks/Table families



Packet flow



References

- Quick reference
- Change history
- Differences with IPTables
- Configuration
  - tables, table families
  - chain know-how, chains, chain hooks
  - rule know-how, rules

  - sets
  - dynamic sets, elements
  - counters
  - limits
  - meters
  - maps, vmaps
  - set meta information, match meta information
- NAT
  - source/destination NAT
  - multi NAT
  - load balancing
- Examples
  - Ruleset for server
  - Ruleset for workstation
- math operations
- mangle package headers
- expressions
- routing information
- bridge filtering
- connection tracker helpers
- debugging

Installation

Kernel Modules

Some functionality of NFTables might not be enabled by default.

To check which was enabled at compile-time - check the config file:

```
cat "/boot/config-$(uname -r)" | grep -E "CONFIG_NFT|CONFIG_NF_TABLES"
```

To find all existing modules:

```
find /lib/modules/$(uname -r) -type f -name '*.ko' | grep -E 'nf_|nft_'
```

To enable a module:

```
modprobe nft_nat
modprobe nft_tproxy
```

Usage

Config File

NFTables can be completely configured from one or more config files.

Most times you might want to use:

- a main config file: `/etc/nftables.conf`
- a configuration directory to include further files: `/etc/nft.conf.d/`

The systemd service will load the main config file by default:

```
# /lib/systemd/system/nftables.service
[Unit]
...

[Service]
...
ExecStart=/usr/sbin/nft -f /etc/nftables.conf
ExecReload=/usr/sbin/nft -f /etc/nftables.conf
ExecStop=/usr/sbin/nft flush ruleset
...
```

Main config file example:

```
#!/usr/sbin/nft -f
flush ruleset
include "/etc/nft.conf.d/*.conf"
```

Then you can add your actual configuration in the configuration directory!

To **test your configuration**:

```
nft -cf /etc/nftables.conf
```

CLI

- CLI overview
- Scripting

Programmatically

THere are some libraries/modules that enable you to manage NFTables from code directly:

- Backend for the libraries: libnftables
- GoLang: github.com/google/nftables, source code
- Python3: documentation, source code, examples

Ansible

See: NFTables Ansible-Role

Troubleshooting

Trace

You can trace traffic that flows through you chains.

See also: NFTables documentation - trace

You need to:

- Tag traffic you want to trace by adding the `meta nftrace set 1` option to a rule.
- Listen to this traces by running `nft monitor trace` in a separate terminal.

You may want to start the trace at the point where the traffic enters.

Example for **input traffic**:

```
chain input {
    type filter hook input priority 0; policy drop;

    # enable tracing for: tcp-traffic to port 1337 originating from a specific network
    tcp dport 1337 ip saddr 192.168.10.0/24 meta nftrace set 1

    ...

}
```

Example for **output traffic**:

```
chain output {
    type filter hook output priority 0; policy drop;

    # enable tracing for: http+s to a specific target
    tcp dport { 80, 443 } ip daddr 1.1.1.1 meta nftrace set 1

    ...

}
```

Example **monitor information**:

```
nft monitor trace
> trace id a95ea7ef ip filter trace_chain packet: iif "enp0s25" ether saddr␣
↪00:0d:b9:4a:49:3d ether daddr 3c:97:0e:39:aa:20 ip saddr 8.8.8.8 ip daddr 192.168.2.
↪118 ip dscp cs0 ip ecn not-ect ip ttl 115 ip id 0 ip length 84 icmp type echo-reply␣
↪icmp code net-unreachable icmp id 9253 icmp sequence 1 @th,64,96␣
↪24106705117628271805883024640
> trace id a95ea7ef ip filter trace_chain rule meta nftrace set 1 (verdict continue)
> trace id a95ea7ef ip filter trace_chain verdict continue
```

(continues on next page)

(continued from previous page)

```
> trace id a95ea7ef ip filter trace_chain policy accept
> trace id a95ea7ef ip filter input packet: iif "enp0s25" ether saddr 00:0d:b9:4a:49:3d␣
↪ether daddr 3c:97:0e:39:aa:20 ip saddr 8.8.8.8 ip daddr 192.168.2.118 ip dscp cs0 ip␣
↪ecn not-ect ip ttl 115 ip id 0 ip length 84 icmp type echo-reply icmp code net-
↪unreachable icmp id 9253 icmp sequence 1 @th,64,96 24106705117628271805883024640
> trace id a95ea7ef ip filter input rule ct state established,related counter packets␣
↪168 bytes 53513 accept (verdict accept)
```

Translate IPTables

Most times the behaviour of IPTables and NFTables is pretty much the same.

In some Distributions the default IPTables backend is already migrated to NFTables.

Why translate from IPTables?

There are 1000x more resources related to IPTables out there that might help you get things working.

I would recommend:

- having a blank VM to test IPTables ruleset
- save the working minimal-ruleset `iptables-save > /etc/iptables/rules.ipt`
- translate the ruleset to nftables `iptables-restore-translate -f /etc/iptables/rules.ipt > /etc/iptables/rules.nft`
- test the NFTables ruleset and remove the default chains you don't need (*IPTables is a little more messy with its defaults*)

BTW: one can also restore IPTables rules by using `iptables-restore < /etc/iptables/rules.ipt`

Config

NFTables base-config example

TPROXY

Quote from the tproxy kernel docs:

```
Transparent proxying often involves "intercepting" traffic on a router.
This is usually done with the iptables REDIRECT target; however, there are serious␣
↪limitations of that method.
One of the major issues is that it actually modifies the packets to change the␣
↪destination address -- which might not be acceptable in certain situations. (Think of␣
↪proxying UDP for example: you won't be able to find out the original destination␣
↪address. Even in case of TCP getting the original destination address is racy.)
The 'TPROXY' target provides similar functionality without relying on NAT.
```

This functionality allows us to send traffic to an userspace process and read/modify it.

This can **enable powerful solutions**! Per example see: blog.cloudflare.com - Abusing Linux's firewall

Warning: TPROXY seems to only support local targets.

As one can see in the kernel sources - there is a check if the target port is in use: nft_tproxy.c

References

- Kernel - TPROXY
- PowerDNS - TPROXY
- Squid - TPROXY
- Policy Routing - TPROXY
- NFTables source - TPROXY
- Kernel source - TPROXY

Usage

One thing you'll need to know: The TPROXY operation can only be used in the **prerouting - filter (mangle)** chain!

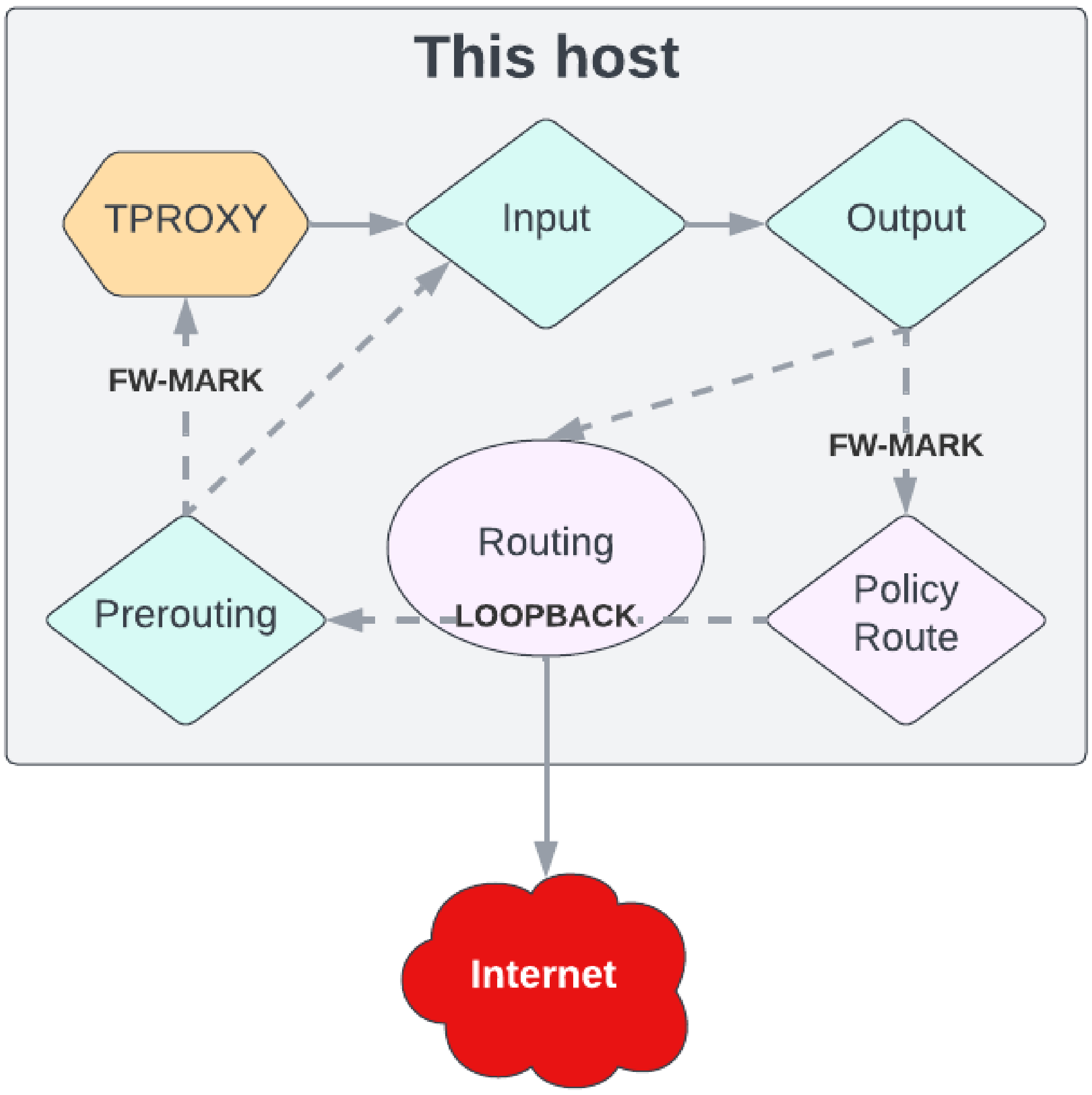
Traffic that passes this chain/hook by default can easily be proxied.

OUTPUT CHALLENGE:

Because of this - traffic that enters at the 'output' (*originating from the same host*) chain/hook can not be redirected directly.

We need to route it to 'loopback' so it passes through 'prerouting'.

NOTE: This image shows the problem we are facing in a very abstract way. It might not display the traffic-flow in a correct manner!



REMOTE PROXY CHALLENGE:

You might want to target a remote proxy server. This does not work with this operation on its own.

One would need to use a proxy-forwarder tool that can handle this for you.

I've patched an existing tool for exactly this purpose: proxy-forwarder

With a tool like that you can wrap the plain traffic received from TPROXY and forward or tunnel it.

```
# NFTables =TCP=> TPROXY (forwarder @ 127.0.0.1) =HTTP[TCP]=> PROXY

> curl https://superstes.eu
# proxy-forwarder
2023-08-29 20:49:10 | INFO | handler | 192.168.11.104:36386 <=> superstes.eu:443/tcp |␣
```

(continues on next page)

(continued from previous page)

```
↪connection established
# proxy (squid)
NONE_NONE/200 0 CONNECT superstes.eu:443 - HIER_NONE/- -
TCP_TUNNEL/200 6178 CONNECT superstes.eu:443 - HIER_DIRECT/superstes.eu -

> curl http://superstes.eu
# proxy-forwarder
2023-08-29 20:49:07 | INFO | handler | 192.168.11.104:50808 <=> superstes.eu:80/tcp |␣
↪connection established
# proxy (squid)
TCP_REFRESH_MODIFIED/301 477 GET http://superstes.eu/ - HIER_DIRECT/superstes.eu text/
↪html
```

Examples

- NFTables TPROXY example, local NFTables TPROXY example
- IPTables TPROXY example, local IPTables TPROXY example

Service

To keep invalid configuration from stopping/failing your `nftables.service` - you can add a config-validation in it:

```
# /etc/systemd/system/nftables.service.d/override.conf

[Service]
ExecStartPre=/usr/sbin/nft -cf /etc/nftables.conf

ExecReload=
ExecReload=/usr/sbin/nft -cf /etc/nftables.conf
ExecReload=/usr/sbin/nft -f /etc/nftables.conf

Restart=on-failure
RestartSec=5s
```

This will catch and log config-errors before doing a reload/restart.

When doing a system-reboot it will still fail if your config is bad.

Addons

NFTables lacks some functionality, that is commonly used in firewalling.

You can add a scheduled scripts that add these functionalities to NFTables!

See: Ansible-managed addons

DNS

It is nice to have variables that hold the IPs of some DNS-record.

NFTables CAN resolve DNS-records - but will throw an error if the record resolves to more than one IP.. (Error: Hostname resolves to multiple addresses)

See: NFTables Addon DNS

IPLists

This addon was inspired by the same functionality provided on OPNSense

It will download existing IPLists and add them as NFTables variables.

IPList examples:

- Spamhaus DROP
- Spamhaus EDROP
- Tor exit nodes

See: NFTables Addon IPList

Failover

See: NFTables Addon Failover

Examples

Ansible

See: Ansible-based examples

IPv4 Baseline

IPv6 Baseline

Security Baseline

Docker host

Proxmox host (PVE)

Forwarder (Router, Network firewall, VPN Server)

Integrations

Fail2Ban

Squid

Tip: If you find some discrepancy or missing information - open a GitHub issue

Was this information useful? Then star the the repository on GitHub

---

Warning: Writing of this documentation is still in progress! Read it with a grain of salt!

2.1.2 Proxy - Squid

Warning: This application let's you intercept and modify network traffic.

That can be illegal => you are warned.

Introduction

Whenever we are referring to a 'client' - it will be a server, workstation or network device in most cases.

Setup

Manual

Docker

You can build a docker image as seen in this repository!

References

- Config examples (*WARNING: some examples are deprecated and will not work on current versions*)

Installation

SSL

If you are only 'peaking' at SSL connections - this should be enough:

```
sudo apt install squid-openssl  # the package needs to have ssl-support enabled at␣
↪compile-time

openssl dhparam -outform PEM -out /etc/squid/ssl_bump.dh.pem 2048

# openssl create self-signed cert
openssl req -x509 -newkey rsa:4096 -keyout /etc/squid/ssl_bump.key -out /etc/squid/ssl_
↪bump.crt -sha256 -days 3650 -nodes -subj "/C=XX/ST=StateName/L=CityName/O=CompanyName/
↪OU=CompanySectionName/CN=Forward Proxy"

# create ssl cache DB
sudo mkdir -p /var/lib/squid
sudo rm -rf /var/lib/squid/ssl_db
sudo /usr/lib/squid/security_file_certgen -c -s /var/lib/squid/ssl_db -M 20MB
sudo chown -R proxy:proxy /var/lib/squid
```

If you want to intercept SSL connections (*Man-in-the-middle like*) - you will have to go through some more steps: squid docs - ssl interception

---

Modes

HTTP_PORT

The `http_port` mode can be used as target proxy in applications like browsers.

Usual port 3128 is used for this mode.

The application creates a HTTP-CONNECT tunnel to the proxy and wraps its requests in it.

DNS resolution is done by the proxy.

HTTPS_PORT

Like mode `http_mode` but the HTTP-CONNECT tunnel is wrapped in TLS.

Usual port 3129 is used for this mode.

For the proxy to be able to handle the DNS resolution - **ssl-bump** must be configured. Else the proxy will not be able to read the Server-Name-Identifier used in the TLS handshake.

INTERCEPT

In this mode the proxy will expect the plain traffic to arrive.

You will have to create a dedicated listener with **ssl-bump** enabled if you want to handle TLS traffic.

See also:

- Squid documentation - interception
- Squid documentation - policy routing

SSL-BUMP

SSL-BUMP allows us to:

- read TLS handshake information
- intercept (*read/modify*) TLS traffic

PEAK

By *peaking* at TLS handshake information in ssl-bump step-1 we are able to gain some important information:

- target DNS/hostname from SNI

Benefits:

- less performance needed than full ssl-interception
- faster than full ssl-interception
- less problems with applications that check certificates on their end (*p.e. banking*)
- no need to create/manage an internal Sub-CA to dynamically create and sign certificates for ssl-intercepted targets

Drawbacks:

- less options to filter the traffic on
- connections to *trustable* targets could carry dangerous payloads

In some cases a basic DNS ‘allow-list’ will be enough to ensure good security. Many automated attacks can be blocked using this approach.

INTERCEPT

This one will be used in **zero-trust** environments.

See also: *Security Stances*

Note:

...

> Even incorrectly used TLS usually makes it possible for at least one end of the communication channel to detect the proxies existence. Squid SSL-Bump is intentionally implemented in a way that allows that detection without breaking the TLS. Your clients **will be capable of identifying the proxy exists**. If you are looking for a way to do it in complete secrecy, dont use Squid.

Benefits:

- ssl-interception gives us much information that can be used to run IPS/IDS checks on
- possible dangerous payloads like downloads can be checked by anti-virus
- more restrictions make even interactive attacks harder to go through

Drawbacks:

- complex ruleset if you go with an *implicit-deny* approach
- much more performance needed
- increasing latency
- with a bad ruleset you will still have security-leaks but also have worse performance (*lose-lose*)

TPROXY

TProxy is a functionality built into current kernels.

It allows us to redirect traffic without modifying it. This solves the issue with overwritten destination-IPs by using Destination-NAT.

The major two integrations of TPROXY we will focus on are the ones in IPTables and NFTables.

In both implementations this is how we will need to handle the three main traffic types:

- **INPUT** traffic: can be redirected to TPROXY directly
- **FORWARD** traffic: can be redirected to TPROXY directly
- **OUTPUT** traffic: needs to be **routed to loopback** to be redirected to TPROXY

Why do we need to send 'output' traffic to loopback? Because TPROXY is only available in the 'prerouting-filter' chain and 'output' traffic does not hit that one by default.

NFTables

See: *NFTables TProxy*

IPTables

See: IPTables TPROXY

Config

Config options

- Matching all subdomains of a Domain can be done by prepending a dot (*'wildcard' matching*)

  Example: '.example.com'

  You may not all 'example.com' and '.example.com' as it will result in a syntax error

- You may want to exclude Port-Probes from your logs:

  ```
  acl hasRequest has request
  access_log syslog:local2 squid hasRequest
  ```

You **need to define listeners**:

See also: Squid documentation - http_port

```
# clients =HTTP[TCP]=> SQUID =TCP=> TARGET
http_port 3128 ssl-bump tcpkeepalive=60,30,3 cert=/etc/squid/ssl_bump.crt key=/etc/squid/
↪ssl_bump.key cipher=HIGH:MEDIUM:!RC4:!aNULL:!eNULL:!LOW:!3DES:!MD5:!EXP:!PSK:!SRP:!DSS␣
↪tls-dh=prime256v1:/etc/squid/ssl_bump.dh.pem options=NO_SSLv3,NO_TLSv1,SINGLE_DH_USE,
↪SINGLE_ECDH_USE

# clients =HTTPS[TCP]=> SQUID =TCP=> TARGET
https_port 3128 ssl-bump tcpkeepalive=60,30,3 cert=/etc/squid/ssl_bump.crt key=/etc/
↪squid/ssl_bump.key cipher=HIGH:MEDIUM:!RC4:!aNULL:!eNULL:!LOW:!3DES:!MD5:!EXP:!PSK:!
↪SRP:!DSS tls-dh=prime256v1:/etc/squid/ssl_bump.dh.pem options=NO_SSLv3,NO_TLSv1,SINGLE_
↪DH_USE,SINGLE_ECDH_USE

# clients =ROUTED TCP=> SQUID =TCP=> TARGET
http_port 3129 intercept
https_port 3130 intercept ssl-bump tcpkeepalive=60,30,3 cert=/etc/squid/ssl_bump.crt␣
↪key=/etc/squid/ssl_bump.key cipher=HIGH:MEDIUM:!RC4:!aNULL:!eNULL:!LOW:!3DES:!MD5:!
↪EXP:!PSK:!SRP:!DSS tls-dh=prime256v1:/etc/squid/ssl_bump.dh.pem options=NO_SSLv3,NO_
↪TLSv1,SINGLE_DH_USE,SINGLE_ECDH_USE

# clients =TPROXY TCP=> SQUID (@127.0.0.1) =TCP=> TARGET
http_port 3129 tproxy
https_port 3130 tproxy ssl-bump tcpkeepalive=60,30,3 cert=/etc/squid/ssl_bump.crt key=/
↪etc/squid/ssl_bump.key cipher=HIGH:MEDIUM:!RC4:!aNULL:!eNULL:!LOW:!3DES:!MD5:!EXP:!
↪PSK:!SRP:!DSS tls-dh=prime256v1:/etc/squid/ssl_bump.dh.pem options=NO_SSLv3,NO_TLSv1,
↪SINGLE_DH_USE,SINGLE_ECDH_USE
```

You can define the **IPs Squid should use for outbound traffic**. This can be useful to define specific firewall rules for those addresses:

```
tcp_outgoing_address 192.168.10.2
tcp_outgoing_address 2001:db8::1:2
```

You may want to cover at least those basic filters:

- **only allow**
  - specific destination ports

    ```
    acl dest_ports port 80
    acl dest_ports port 443
    acl dest_ports port 587
    http_access deny !dest_ports
    ```

  - only allow proxy-access **from specific source** networks

    ```
    acl src_internal src 127.0.0.0/8
    acl src_internal src 192.168.0.0/16
    acl src_internal src 172.16.0.0/12
    acl src_internal src 10.0.0.0/8
    http_access deny !src_internal
    ```

  - only allow access **to specific destinations**
    - filter on an IP-basis

      ```
      acl dst_internal src 192.168.0.0/16
      acl dst_internal src 172.16.0.0/12
      acl dst_internal src 10.0.0.0/8
      http_access allow dst_internal
      http_access deny all
      ```

    - filter on a DNS-basis (*SSL-Bump 'Peak' needed*)

      ```
      acl domains_allowed dstdomain example.com
      acl domains_allowed dstdomain superstes.eu
      http_access allow domains_allowed
      http_access deny all
      ```

- **check server certificates** for issues (*expired, untrusted, weak ciphers*)

  ```
  tls_outgoing_options options=NO_SSLv3,NO_TLSv1,SINGLE_DH_USE,SINGLE_ECDH_USE␣
  ↪cipher=HIGH:MEDIUM:!RC4:!aNULL:!eNULL:!LOW:!3DES:!MD5:!EXP:!PSK:!SRP:!DSS
  acl ssl_exclude_verify dstdomain .example.com
  sslproxy_cert_error allow ssl_exclude_verify
  sslproxy_cert_error deny all
  ```

- enable **ssl-bump 'peaking'**

  ```
  sslcrtd_program /usr/lib/squid/security_file_certgen -s /var/lib/squid/ssl_db -M␣
  ↪20MB

  acl CONNECT method CONNECT
  ```

(continues on next page)

(continued from previous page)

```
acl ssl_ports port 443
acl step1 at_step SslBump1

http_access deny CONNECT !ssl_ports
http_access allow CONNECT step1  # without 'step1' here one would be able to 'tunnel
↪' unwanted traffic through the proxy
ssl_bump peek step1 ssl_ports
ssl_bump splice all
```

Service

To keep invalid configuration from stopping/failing your `squid.service` - you can add a config-validation in it:

```
# /etc/systemd/system/squid.service.d/override.conf

[Service]
ExecStartPre=
ExecStartPre=/usr/sbin/squid -k parse
ExecStartPre=/usr/sbin/squid --foreground -z

ExecReload=
ExecReload=/usr/sbin/squid -k parse
ExecReload=/bin/kill -HUP $MAINPID

Restart=on-failure
RestartSec=5s
```

This will catch and log config-errors before doing a reload/restart.

When doing a system-reboot it will still fail if your config is bad.

Examples

Transparent Proxy

Sometimes setting the environment-variables ‘HTTP_PROXY’, ‘HTTPS_PROXY’, ‘http_proxy’ and ‘https_proxy’ for all applications and HTTP-clients may be problematic/too inconsistent.

An attacker might also be able to modify the environmental variables once a vulnerability has been exploited.

Destination NAT

In some older tutorials and write-ups you will see that people DNAT traffic from a 'client' system to a remote proxy server.

This **IS NOT SUPPORTED** by squid.

It will lead to an error like this: 'Forwarding loop detected'

Why is that?

Squid's transparent operation modes DO NOT handle DNS resolution! They instead use the actual destination IP from the IP-headers and send the outgoing traffic to it. That is because of some vulnerability

When using DNAT the destination IP is set to the proxy's IP. Therefore => loop.

Routed Traffic

You can use this option if the proxy server shares a Layer 2 network with the system that sends or routes the traffic.

Practical examples of this:

- Network gateway (*router*) sends traffic to proxy for interception
- 'Client' devices use the proxy as gateway instead of the actual router

In this case we will need to set-up Squid listeners in **intercept** mode to process the traffic.

You could also use the **tproxy** mode - but that might be more complicated to set-up when you want to check the traffic that enters at the 'forwarding' chain.

Forwarded Traffic

In some situations you will not be able to use the option to route the traffic to the proxy.

This might be because:

- you are not controlling the gateway/router
- the 'client' device is isolated (*only connected to WAN*)
- client and/or network restrictions don't allow for re-routing the traffic

Practical examples of this:

- A Cloud VPS or Root Server that is only connected to WAN
- Distributed Systems using a central proxy (*p.e. on-site at customers*)

In this case we might need other tools like proxy-forwarder to act as forwarder:

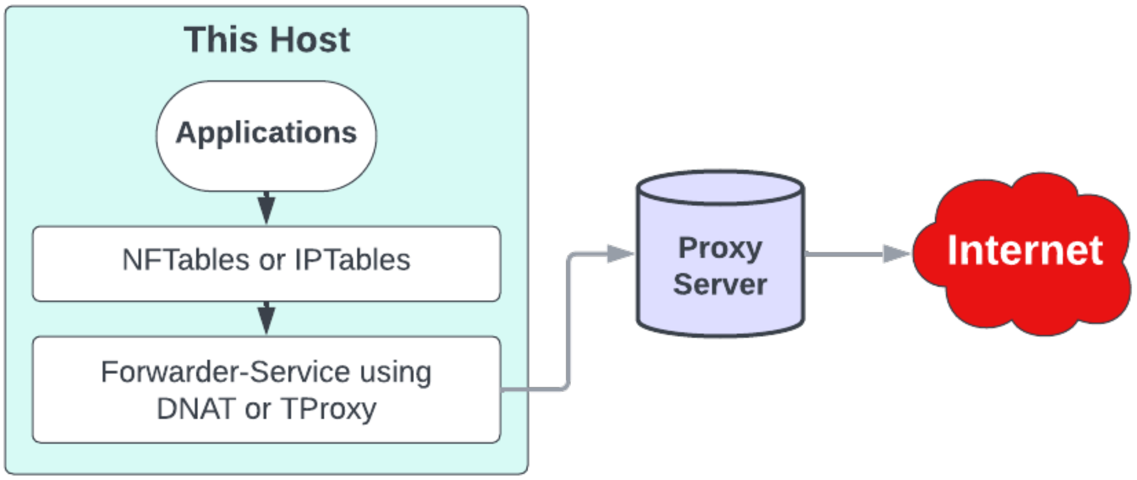
```
> curl https://superstes.eu
# proxy-forwarder
2023-08-29 20:49:10 | INFO | handler | 192.168.11.104:36386 <=> superstes.eu:443/tcp |␣
↪connection established
# squid
NONE_NONE/200 0 CONNECT superstes.eu:443 - HIER_NONE/- -
TCP_TUNNEL/200 6178 CONNECT superstes.eu:443 - HIER_DIRECT/superstes.eu -

> curl http://superstes.eu
```

(continues on next page)

(continued from previous page)

```
# proxy-forwarder
2023-08-29 20:49:07 | INFO | handler | 192.168.11.104:50808 <=> superstes.eu:80/tcp |␣
↪connection established
# squid
TCP_REFRESH_MODIFIED/301 477 GET http://superstes.eu/ - HIER_DIRECT/superstes.eu text/
↪html
```



Troubleshooting

What does not work?

One might want to try some other ways of sending/redirecting traffic to a squid proxy.

Here are some examples that **DO NOT WORK**

- Destination NAT to remote Squid server in transparent mode

  ```
  # journalctl -u squid.service -n 50
  ...
  WARNING: Forwarding loop detected for
  ...
  TCP_MISS/403 ORIGINAL_DST/<proxy-ip>
  ...
  ```

- DNAT 80/443 to squid in non-transparent mode

  ```
  # journalctl -u squid.service -n 50
  ...
  Missing or incorrect access protocol
  ...
  NONE/400
  ...
  ```

- IPTables/NFTables TPROXY to socat forwarder

  SOCat is actually correctly receiving and forwarding the traffic - BUT practically it acts like a DNAT operation

```
# 'client'
socat tcp-listen:3129,reuseaddr,fork,bind=127.0.0.1,ip-transparent tcp:<proxy-ip>
↪:3129

# journalctl -u squid.service -n 50
...
WARNING: Forwarding loop detected for
...
TCP_MISS/403 ORIGINAL_DST/<proxy-ip>
...
```

- Intercept/TPROXY mode with Squid inside docker container

  Essentially docker seems to be NATing the traffic.

```
ERROR: NF getsockopt(ORIGINAL_DST) failed on conn18 local=192.168.0.2:3130␣
↪remote=192.168.0.1:48910 FD 12 flags=33: (2) No such file or directory
ERROR: NAT/TPROXY lookup failed to locate original IPs on conn18 local=192.168.0.
↪2:3130 remote=192.168.0.1:48910 FD 12 flags=33
```

Known problems

- **Clients have many timeouts**

  It may be that your cache size is too small.

  This can happen when many requests hit the proxy in a short time period.

  Possible Solution:

  – Increase your main cache:

    `cache_mem 1024 MB` (see docs - cache_mem)

  – Increase your session cache:

    `sslproxy_session_cache_size 512 MB`

  – Increase your ssl cache (*only if you intercept ssl*)

    ssl_db `=> sslcrtd_program /usr/lib/squid/security_file_certgen -s /var/lib/squid/ssl_db -M 256M`

  – Increase your ssl session timeout

    `sslproxy_session_ttl 600`

- **Bus error**

  It seems this happens when the value of `sslproxy_session_cache_size` is larger than the one of `ssl_db`

- **NONE_NONE/409 & SECURITY ALERT: Host header forgery detected**

  This error can occur whenever the squid proxy runs in `intercept` mode and resolves the target hostname to another IP than the client.

  That check can help against attacks that can trick the proxy into allowing bad traffic.

As today's DNS servers use very low TTLs it might happen that some traffic triggers this check as false-positive.

You can disable this check **for HTTP (plaintext) traffic** by setting `host_verify_strict off` (*default*)

HTTPS traffic will still be forced fail for some unclear reason.. :(

See also: Squid wiki - host_verify_strict & Squid wiki - host header forgery

You could - of course use the proxy-forwarder to translate the intercepted TCP traffic into HTTP & HTTPS requests that you are able to send to the 'forward-proxy' port of squid. (*that one will ignore that check…*)

Tip: If you find some discrepancy or missing information - open a GitHub issue

Was this information useful? Then star the repository on GitHub

2.1.3 Proxy Tool - GOST

Warning: This tool can be used to hide/forward malicious network traffic.

That can be illegal => you are warned.

Introduction

GOST is a tool for proxying pretty much anything and anyhow you want/need to.

Check out the nice documentation!

It can act as proxy server and client/forwarder.

If you need to be able to route some traffic through some kind of proxy - this is the tool for you!

It can proxy:

- TLS Tunnel
- HTTP/HTTPS Tunnel
- PROXY Protocol
- DNAT/REDIRECT Traffic (*originated from the same host*))
- TRPOXY integration
- ICMP tunnel
- And has many more hacky features

Forwarding to HTTP Proxy

```
# NFTables =TCP=> TPROXY (forwarder @ 127.0.0.1) =HTTP[TCP]=> PROXY (squid http_port)
```

The current implementation of HTTP-forwarding in gost does not work correctly.

Problems:

- HTTP not working (*always wants to tunnel over HTTP-CONNECT*)
- HTTPS over IPv6 not working

Solution:

I've created a patched version of gost for exactly this purpose: proxy-forwarder

DNAT

You can use it to catch DNAT traffic and forward it to a remote proxy-server like squid:

```
proxy-forwarder -P 4128 -F http://192.168.10.20:3128
# creates tcp & udp listeners for IPv4 & IPv6 on localhost:4128

# NAT non-internal targets to the proxy
## nftables
nft add rule nat output ip daddr != { 192.168.0.0/16, 10.0.0.0/8, 172.16.0.0/12 } tcp␣
↪dport { 80, 443 } dnat to 127.0.0.1:3128

## iptables
iptables -t nat -A OUTPUT -p tcp ! -d 172.22.0.0/12 --dport 443 -j DNAT --to-destination␣
↪127.0.0.1:3128
iptables -t nat -A OUTPUT -p tcp ! -d 172.22.0.0/12 --dport 80 -j DNAT --to-destination␣
↪127.0.0.1:3128
```

TPROXY

If you want to also proxy UDP traffic - you might want to use the TPROXY integration:

```
proxy-forwarder -P 4128 -F http://192.168.10.20:3128 -T

# to also set a fw-mark on processed traffic
proxy-forwarder -P 4128 -F http://192.168.10.20:3128 -T -M 100
```

Config Examples:

- IPTables TPROXY
- NFTables TPROXY

Privileges

You can run GOST TPROXY-mode with non-root users if you add a capability to the binary:

```
sudo setcap cap_net_raw+ep /usr/local/bin/gost
sudo chown root:gost /usr/local/bin/gost
chmod 750 /usr/local/bin/gost
```

Tip: If you find some discrepancy or missing information - open a GitHub issue

Was this information useful? Then star the repository on GitHub

2.1.4 Proxy Tool - ProxyShark

Warning: This application let's you intercept and modify network traffic.

That can be illegal => you are warned.

Introduction

I came across a challenge that needed me to modify tcp packages as man-in-the-middle.

I found a nice tool to do that; but without much information on how to set it up or use it.

Therefore I will provide those info's to you:

The tool I'm talking about is called 'proxyshark'. It's a python2 program/script written by the company '*CONIX Cybersecurity*' from france.

Just so I acknowledged it:

1. I know you could code this from scratch by only using the netfilterqueue and scapy modules
2. There are cleaner, faster and more up-to-date projects out there.. (p.e. pypacker)

Source

Original

- GitHub

My fork containing the fixes as described below:

- ProxyShark Fork

Dependencies

At first I had some problems getting started with this script since the dependencies are ‘a little’ outdated.

Disclaimer: You might want to use this on a vm - since the dependencies will foul your system a little.

```
# install pip2 dependencies
apt install build-essential python-dev libnetfilter-queue-dev tshark git python2

# install pip2
wget https://bootstrap.pypa.io/pip/2.7/get-pip.py
sudo python2 get-pip.py

# install pip packages
python2 -m pip install dnspython pyparsing

# scapy workaround
python3 -m pip install scapy
sudo mkdir /usr/lib/python2.7/dist-packages/scapy
cd /usr/lib/python3/dist-packages/
sudo cp -avr scapy/* /usr/lib/python2.7/dist-packages/scapy
```

Bugs & Fixes

Code

Note: The fixes are included in my fork of this project => link can be found above.

The capture bugged out on me.

Therefore I fixed some ‘bugs’ that broke it:

```
# Line 1159 => comment out the existing regex and replace it with this one:

# regex = r'^.*(\d+\.\d+) +([^ ]+) +-> +([^ ]+) +([^ ]+) +[^ ]+ +(.*)$'
regex = r'.*?(\d+\.\d+)\s*((?:[0-9]{1,3}\.){3}[0-9]{1,3}).*?((?:[0-9]{1,3}\.){3}[0-91,3}
↪)\s*([A-Z]{3,20})\s*(.*)'  # might be problematic in edge-cases

# Line 1674 => add those lines before the 'if item_showname' matching:
if type(item_show) == 'unicode':
    item_show = item_show.encode('utf8')
if type(item_showname) == 'unicode':
    item_showname = item_showname.encode('utf8')
```

Execution

Tshark

Run it with the '-t /usr/bin' argument to use the newer apt-version of **tshark** => the one from their repo bugged out on me..

Packet modification

Somehow the 'direct' packet modification did not work.

I found the list object '_items_to_commit' and modified the value in it => that worked

```
bpkt._items_to_commit[22]['value'] = 'aa02'
```

Usage

Some basic commands can be found in the ReadMe of the repository!

```
python2 ps1/proxyshark.py -v --capture-filter 'dst host 8.8.8.8' --packet-filter 'udp' -
↪t /usr/bin/
```

- Start the capture from the **interactive mode** by typing 'run'
- You should now see packages matching the filter by **typing 'packet'** (*last one*) or **'queue'** (*all*)
- Those packages can be 'caught' by defining a **breakpoint**.
- Captured packets can be **interacted** with in the default **python2 syntax**.
- **All possible attributes** etc. can be displayed by entering **'bpkt.__dict__'** (*after capturing some packet with a breakpoint*)
- Breakpoints can either pause the capture, so the packet can be edited manually, or modify it automatically with defined action

At first you might want to play around with the attributes of captures packets.

After that you can write automated actions to become a fully-grown m.i.t.m.

Examples

- show tcp data attributes

  ```
  bpkt['tcp.data']
  ```

- show tcp flags

  ```
  bpkt['tcp.flags']
  ```

- show ip destination

```
bpkt['ip.dst']
```

- rewrite tcp-reserved bits in auto-mode:

```
a add ta1 to tb1 "_flags = bpkt['tcp.flags'][0]['unmaskedvalue'][:1] + 'a'␣
↪+ bpkt['tcp.flags'][0]['unmaskedvalue'][2:]" "bpkt._items_to_commit[22][
↪'value'] = _flags" "print _flags" "bpkt.accept()"
```

Tip: If you find some discrepancy or missing information - open a GitHub issue

Was this information useful? Then star the repository on GitHub

Warning: Writing of this documentation is still in progress! Read it with a grain of salt!

2.1.5 Security Approaches

Introduction

When it comes to IT- and Network security there are some main approaches/stances you may want to know.

Tip: Feel free to start Discussions regarding this topic!

If you too have gained experiences in this area I would be interested if you are missing something.

What to secure

Network components

- Switches
- Routers
- Network firewalls
- VPN endpoints

Network clients

- User devices like laptops, desktops, thin-clients
  - Different operating systems like Windows, MacOS, Linux, ChromeOS, Android, …
- Servers
  - Different operating systems like Windows Server, Linux, …
- Smartphones, tables

- IOT, automation devices
- Printers

Cloud resources

Nearly all IT infrastructures have at least some components hosted on some cloud platform.

The main security of those services needs to ensured by the cloud providers - but consumers also need to ensure secure configuration on their side.

Virtualization infrastructure

Many companies use virtualization to share server-hardware between many virtual servers.

- Hypervisors
- Storage servers
- Control nodes

Some companies also use containerization like plain docker or even kubernetes to run their services.

Information

Information is essential - it also needs to be secured.

- Databases
- Files
- Backup systems
- Applications

Overview

I will try to keep this overview of existing stances/mindsets as objective as possible.

These are very theoretically - but I will go into how they can translate into practical usage in the *personal experience <net_security_approach_personal>* section!

Perimeter-based

Classic network designs were built around the concept of an enterprise LAN consisting of switches, routers and Wi-Fi connectivity.

The LAN contained one or more data centers, which housed applications and data.

This LAN formed the security network perimeter.

Accessing apps and services via the internet, VPNs and remote sites across WAN connections is considered external to the organization with perimeter-based security.

Everything connected to the LAN is considered "trusted," and devices coming from outside the perimeter are "untrusted".

This means external users must prove who they are through various security and identification tools.

Zero Trust

Reference: NIST SP 800-207 - Zero Trust Architecture

TLDR: 'Implicit trust is always a vulnerability, and therefore security must be designed with the strategy of "Never trust, always verify"'

Quotes

I'll quote some companies that provide solutions in the Zero-Trust area:

- **NIST SP 800-207**

  Source: NIST SP 800-207 PDF

  > A typical enterprise's infrastructure has grown increasingly complex. A single enterprise may operate several internal networks, remote offices with their own local infrastructure, remote and/or mobile individuals, and cloud services. This complexity has outstripped legacy methods of perimeter-based network security as there is no single, easily identified perimeter for the enterprise. Perimeter-based network security has also been shown to be insufficient since once attackers breach the perimeter, further lateral movement is unhindered.
  >
  > This complex enterprise has led to the development of a new model for cybersecurity known as "zero trust" (ZT). A ZT approach is primarily focused on data and service protection but can and should be expanded to include all enterprise assets (devices, infrastructure components, applications, virtual and cloud components) and subjects (end users, applications and other non-human entities that request information from resources). Throughout this document, "subject" will be used unless the section relates directly to a human end user in which "user" will be specifically used instead of the more generic "subject." Zero trust security models assume that an attacker is present in the environment and that an enterprise-owned environment is no different—or no more trustworthy—than any nonenterprise-owned environment. In this new paradigm, an enterprise must assume no implicit trust and continually analyze and evaluate the risks to its assets and business functions and then enact protections to mitigate these risks. In zero trust, these protections usually involve minimizing access to resources (such as data and compute resources and applications/services) to only those subjects and assets identified as needing access as well as continually authenticating and authorizing the identity and security posture of each access request.

- **VMWare**

  Source: VMWare - Zero Trust

  > Zero Trust Security is a concept created on the belief that implicit trust is always a vulnerability, and therefore security must be designed with the strategy of "Never trust, always verify". In its simplest form, Zero Trust restricts access to IT resources using strictly enforced identity and device verification processes.
  >
  > Zero Trust enforces the use of stringent security controls for users and devices before they can gain access to protected resources. Zero Trust identity authentication and authorization use the principle of least privilege (PoLP), which grants the absolute minimum rights required for a given function – before a single packet is transferred.

> This has become necessary because of the changes in how network resources are accessed. Gone are the days of a network perimeter or VPN-only access; today's increasingly mobile workforce and growth in the work-at-home movement demand new security methods be considered for users, while the increasingly distributed nature of computing with containers and micro-services means that device-to-device connections are increasing as well.
>
> Thus, Zero Trust requires mutual authentication to confirm the identity and integrity of devices regardless of location to grant access based on the confidence of device identity, device health, and user authentication combined.

- **Microsoft**

  Source: Transform to zero-trust-model

  > The traditional firewall (VPN security model) assumed you could establish a strong perimeter, and then trust that activities within that perimeter were "safe." The problem is today's digital estates typically consist of services and endpoints managed by public cloud providers, devices owned by employees, partners, and customers, and web-enabled smart devices that the traditional perimeter-based model was never built to protect. We've learned from both our own experience, and the customers we've supported in their own journeys, that this model is too cumbersome, too expensive, and too vulnerable to keep going.
  >
  > We can't assume there are "threat free" environments. As we digitally transform our companies, we need to transform our security model to one which assumes breach, and as a result, explicitly verifies activities and automatically enforces security controls using all available signal and employs the principle of least privilege access. This model is commonly referred to as "Zero Trust."

Basically

Benefits:

- tbc..

Drawbacks:

- Many different systems and lots of configuration is needed to ensure practical Zero-Trust - this brings some problems with it:
  - It may be very costly to operate those systems
  - Administrating those systems may consume more time
  - Keeping an overview of the configuration may get harder
  - How these systems need to be operated differs greatly from
- The users experience might be negatively impacted (*false-positive blocks, increased latency, more authentication*)
  - This essentially can bring down the company's productivity/efficiency by a little
- tbc..

To keep an overview over the configuration of all those systems one might also need to implement infrastructure-as-code to centralize it.

Personal experience

Here I will go into what these theoretical approaches can look like in practice.

Take this information with a grain of salt as it is spawned from the IT-environments I've interacted with and

Tip: If you find some discrepancy or missing information - open a GitHub issue

Was this information useful? Then star the repository on GitHub

2.1.6 Tor

Warning: Tor SHOULD NOT be used to masquerade the source of malicious network requests.

That can be illegal => you are warned.

Introduction

Tor is used to anonymize network requests and host/allow access to hidden (*.onion*) services.

Warning: Traffic you send over tor should **ALWAYS be encrypted**!

Else malicious tor nodes might be able to read and even modify your requests.

Just using the tor network won't be enough to stay anonymous!

You should also always follow the basic guidelines on how to stay anonymous.

If you are serious about your anonymity you might even go a set further and:

- Set-up a virtual machine to use for accessing tor.

  Per example: whonix

- Connect that vm to a network that has no access to internal resources. (*only internet access, maybe a guest-network or 'dmz'*)

Route traffic over tor

Tor uses the protocol SOCKS5.

A technical limitation of the tor implementation is that only TCP-traffic can be sent over those connections.

DNS

DNS-requests can be tunneled specially to prevent dns-leaking. (*your provider [and so on..] will know what you are accessing*)

Implementation examples:

- Cloudflare hidden resolver
- local tor dns-port

  Basically:

  ```
  # example on ubuntu/debian

  sudo -i
  #   installing
  apt update
  apt install tor -y
  #     if you want to start it on system boot
  systemctl enable tor

  #   configuration
  echo 'DNSPort 5353' > /etc/tor/torrc
  systemctl start tor
  echo 'nameserver 127.0.0.1' > /etc/resolv.conf  # will not be persistent
  ```

- You might want to enable DNS-over-HTTPS or DNS-over-TLS to keep your requests secure.

Web access

To access websites anonymously you can use the tor browser.

Basics:

- Use DuckDuckGo as **search engine**!

  Other engines like Google will compromise your anonymity!
- Don't log-in with your **personal accounts** => it will compromise your anonymity.
- Don't enable **JavaScript** (*disabled via NoScript by default*)

  Some websites won't work correctly => but it keeps you safe(r).
- You might see **websites blocking your requests** as the most providers are blocking or limiting requests from tor exit-nodes.

  Maybe pressing 'Ctrl+Shift+L', to use a new 'route' for accessing the current page, might help.
- Using **hidden services**

  The 'default' websites you use daily using your normal browser are in a domain of the internet called 'clear-net'.

  You can browse them without worrying too much - they might 'just' compromise your anonymity.

  Hidden services are in the 'deep-net' => those are hidden for the usual user and only reachable using tor.

  For clarification: tor is only ONE network that hosts hidden services - there are more out there

  Hidden services have their own search engines but they have not listed many of the existing services!

For the most** part you need to know the unique **.onion** address of a service to access it.

Warning: **Be aware**:

- You might see/find disturbing and/or illegal content on those hidden services.
- You need to have a basic technical understanding on how to interact with those services securely - else you might even get hacked.

Specific program

Whenever you want a program to use tor as 'gateway' for its connection you need to 'proxy' it.

That proxy needs a tor 'SOCKS' socket to connect to.

SOCKS socket:

- The easiest way of starting such a socket is by opening the tor browser

  It starts such a socket in the background!

  ```
  socks5 127.0.0.1 9150
  ```

- Another way is to install & start tor as service

  ```
  socks5 127.0.0.1 9050
  ```

Linux

On linux I would recommend using the application 'proxychains4' to achieve that.

You just need to set the SOCKS target to use.

```
# example: tor browser SOCKS
sudo -i
echo 'socks5  127.0.0.1 9150' > /etc/proxychains4.conf
```

After that you can just start the application that should connect over tor by prepending 'proxychains4' to its command:

```
# without tor
curl https://ipinfo.io/city
# using tor
proxychains4 curl https://ipinfo.io/city
```

SSH

You can also set a proxy for ssh-connections.

Another program called 'netcat' is needed to archive that.

You will need to install the variant 'netcat-openbsd' as the 'default' one does not implement the needed options.

```
# example on ubuntu/debian using tor browser SOCKS
#   install dependencies
sudo apt update
sudo apt install openssh netcat-openbsd -y

#   use
ssh -p PORT -o ProxyCommand="nc -X5 -x127.0.0.1:9150 %h %p" USER@SERVER
```

All Traffic

There are options to send all **TCP-Traffic and DNS-Requests** over the tor network.

Warning: This should only be used if you really know what you are doing - as there are many ways you might compromise your anonymity!

Linux

Here is the official guide to proxying.

I won't go into the details on how to set this up - as I have not got experience with it.

It is done something like this: (*copied from the guide*)

```
# example for 'middlebox' on ubuntu/debian
sudo -i
#   installing
apt update
apt install tor -y
#     if you want to start it on system boot
systemctl enable tor

#   writing config
echo 'VirtualAddrNetworkIPv4 10.192.0.0/10' > /etc/tor/torrc
echo 'AutomapHostsOnResolve 1' >> /etc/tor/torrc
echo 'TransPort 192.168.1.1:9040' >> /etc/tor/torrc
echo 'DNSPort 192.168.1.1:5353' >> /etc/tor/torrc
systemctl restart tor

#   adding traffic redirection
_trans_port="9040"
_inc_if="eth1"  # you need to update the interface
iptables -t nat -F  # WARNING: will remove all existing NAT-rules
```

(continues on next page)

(continued from previous page)

```
iptables -t nat -A PREROUTING -i $_inc_if -p udp --dport 53 -j REDIRECT --to-ports 5353
iptables -t nat -A PREROUTING -i $_inc_if -p udp --dport 5353 -j REDIRECT --to-ports 5353
iptables -t nat -A PREROUTING -i $_inc_if -p tcp --syn -j REDIRECT --to-ports $_trans_
↪port
```

Windows

You can use a tool like OnionFruit.

CHAPTER

THREE

CODING

Tip: If you find some discrepancy or missing information - open a GitHub issue

Was this information useful? Then star the repository on GitHub

3.1 Coding