Practical Security

System and Network Administration

Revision 2 (2020/21)

Pierre-Philipp Braun <pbraun@nethence.com>

Table of contents

Security Basics

assuming a server or network device – not a desktop

What are the most important things to do in terms of security?…

==>

1. check what is listening – *seen in sna/daemons*
2. system updates – *idem*
3. no passwords (or at least no weak passwords)

Systems security in a nutshell

- weak passwords = no passwords
- KEEP YOUR SERVERS AND NETWORK DEVICES UP-TO-DATE !
- `LISTEN` vs firewall

How to do that again?…

==> That should be clear by now

```
netstat -lntup
```

Also check remotely for clarity and in case a rootkit is hiding some listening port

```
nmap -sTUV -p0-65535 YOUR-SERVER
```

Auto-updates policy

Need to evaluate the risk of downtime

- Debian/Ubuntu restarts the services after upgrade
- RHEL/CentOS does not
- care about listening daemons mostly
- kernel patch - what part is truly in use? (downtime)

Auto-updates on GNU/Linux

update systems –> licensing lecture

Ubuntu

```
apt install unattended-upgrades
dpkg-reconfigure -plow unattended-upgrades
```

Slackware choice 1:

LAB // PoC autoslack for 14.2 vs current[1] – is it too dangerous to auto-update current and why?

Slackware choice 2: DIY

```
vi /etc/cron.daily/DAILY


# don't do that on current
/usr/sbin/slackpkg update
/usr/sbin/slackpkg -batch=on -default_answer=y upgrade-all
```

[1] http://www.slackware.com/~david/zuul/autoslack/

Auto-updates on BSD systems

Possibly scriptable

- FreeBSD: ?
- DragonFlyBSD: ?
- NetBSD: grab from nightly builds and erase everything but `/etc/`

Possibly automated

- OpenBSD: syspatch utility
- MirBSD: *idem*?

Network security in a nutshell

Split activities into VLANs

- e.g. DMZ, VoIP, user, mgmt/backup, cluster, storage

Know your location - what network segment?

- got public IP?
- –or– NAT outbound traffic allowed?
- how many enemies you have?

Isolate insecure industrial tools and software e.g. SCADA

Userland rootkit checkers

some good practice I wouldn't recommend

Regularly check against rootkits

- makes me think of grand-ma who absolutely needs an anti-virus
- it most probably won't detect anything targeted…

LAB // evaluate those detectors against DIY modifications…

mainly GNU/Linux

- Lynis
- Chkrootkit
- Rkhunter
- ClamAV
- LMD

–or– simply overwrite the binaries (BSD & Slackware)

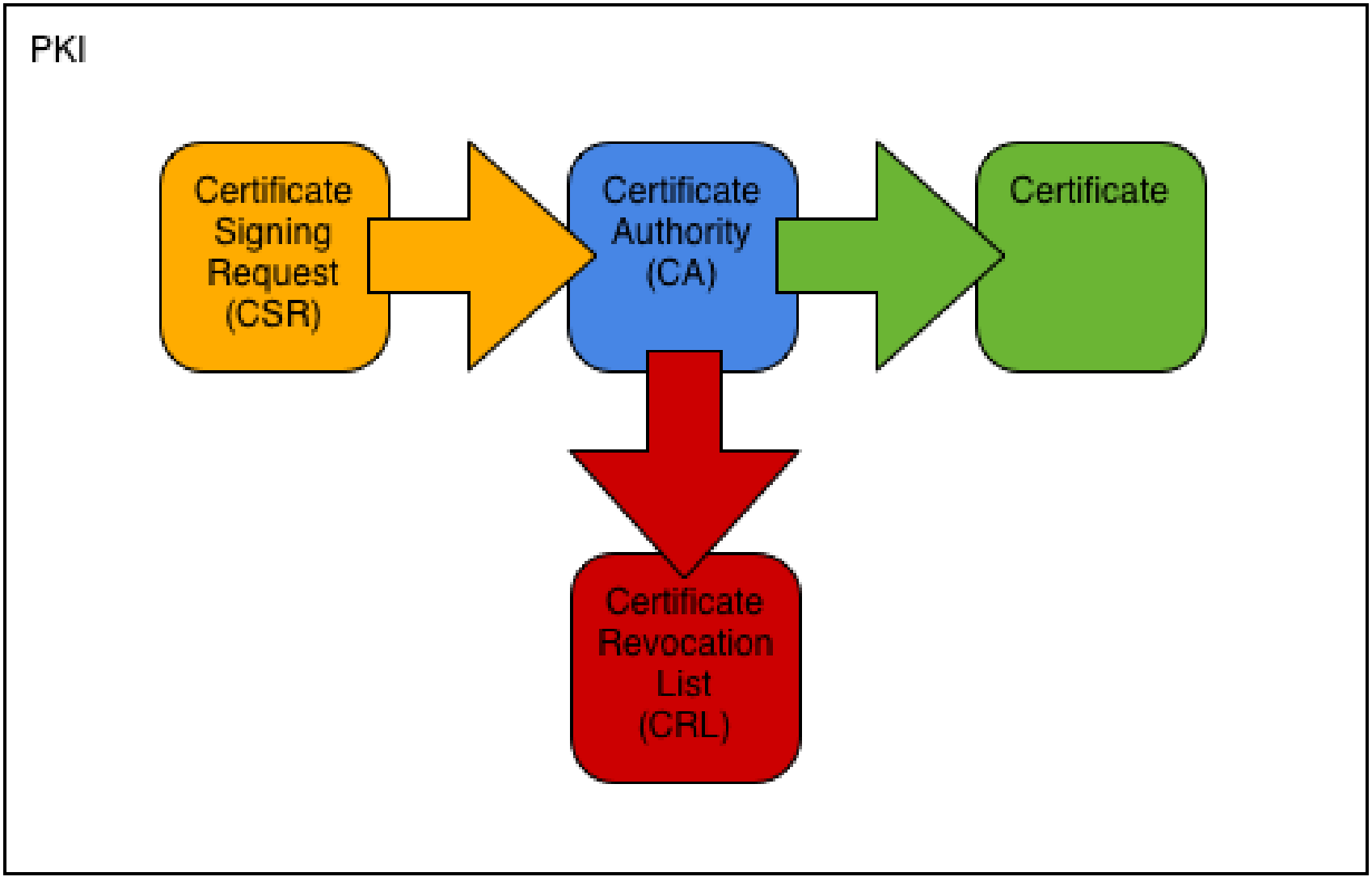
// Questions on security basics?

PKIX

What is the purpose of an SSL certificate?…

==> bind pubkey & domain name **and signed by** (intermediate) **authority**

Looks like there's only one root…



Works many times… **and at the same time**

SSL/TLS certificate types

- Signed by official CAs (embedded Mozilla & Chrome)
- Self-signed (just like a root cert)
- Signed by a private CA (pushed to workstations or added once)

Ubuntu ships self-signed for convenience (what about Debian?)

```
/etc/nginx/snippets/snakeoil.conf
/etc/ssl/certs/ssl-cert-snakeoil.pem
/etc/ssl/private/ssl-cert-snakeoil.key
```

// Questions on PKIX?

Applied Cryptography

Categories for crypto

- Symmetric ciphers
- Public Key ciphers
- Hash algorithms
- Key negotiation algorithms

*Name a few **symmetric** ciphers (block vs stream)…*

64-bit symmetric block ciphers

DES, 3DES

IDEA

Blowfish (Bruce Schneier)

128-bit symmetric block ciphers

AES (Rijndael, NIST)

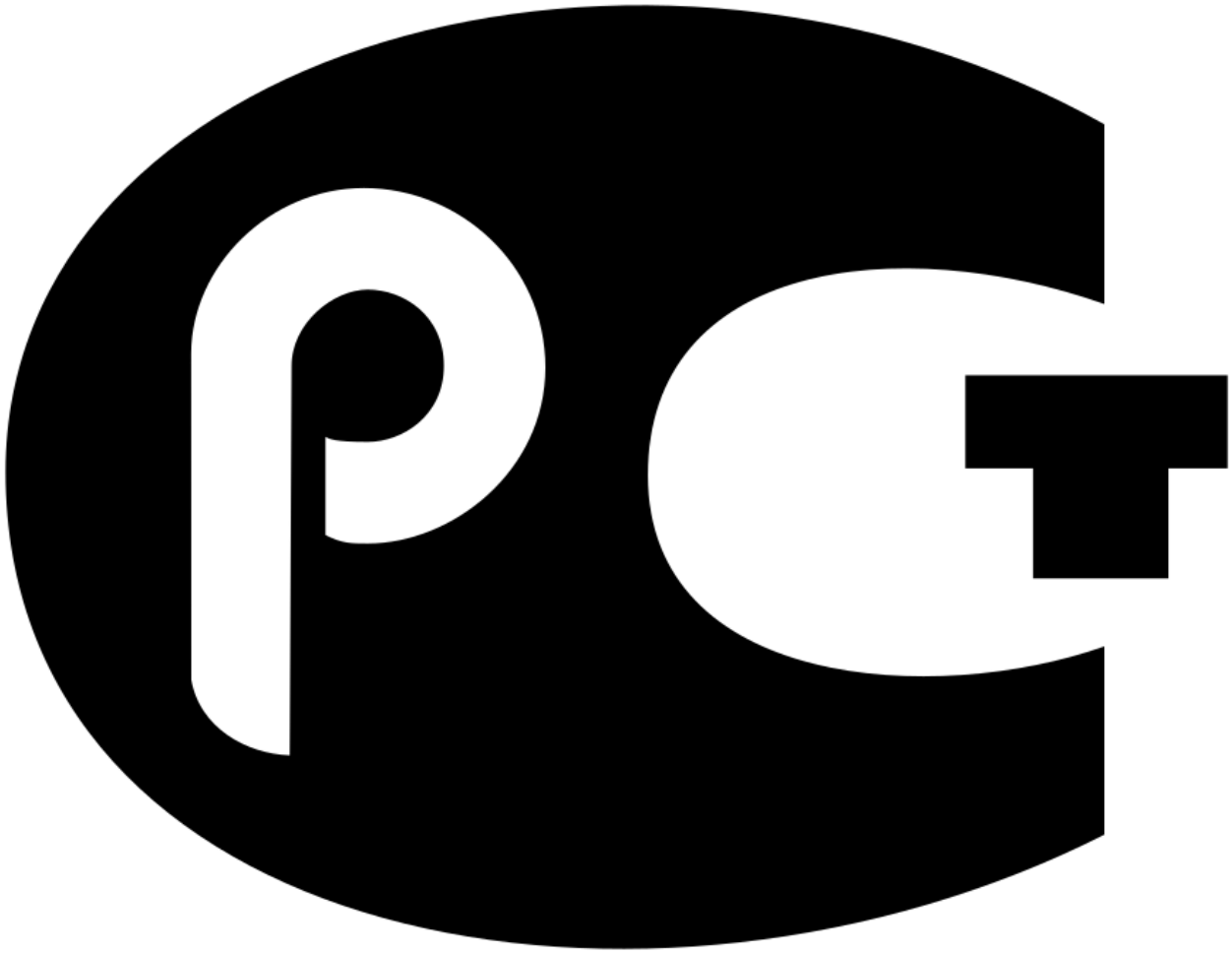
Camellia (Japan)

Twofish (Bruce Schneier)

SEED (South Korea)

ARIA (based on AES, South Korea)

GOST (w/o the H, Russia)



LAB // ГОСТ only with LibreSSL and GNUTLS?

Symmetric stream ciphers

FISH (not seen)

RC4 (deprecated)

CHACHA20

A5/1

Modes of operation (for block cipher)

```
ECB (there's a catch)
CBC (to-be deprecated)
CTR (make it a stream)
GCM (idem)
```

But what do you do for the last block, if there is not enough data to fit-in?

Padding

For the last block and with non-streaming modes

```
zero-padding (the catch is not that obvious)
PKCS#1 v1.5
OAEP
PSS
```

What about hash functions?…

Hash functions

MD5 (not really deprecated)

SHA-1 (only deprecated for SSL certs)

SHA-2 (SHA-256, ...)

SHA-3 (NIST 2015) -- sponge construction

Loads of acronyms…

- MAC Medium Access Control address (OSI layer 2)
- MAC Message Authentication Code (more than just a hash)
- MAC Mandatory Access Control (vs DAC/RBAC)

MACs / HMACs

IDEA

HMAC-MD5

SHA-1

Poly1305 (Daniel J. Bernstein)

Name a few public key ciphers (or signature schemes)…

Public key (asymmetric) ciphers

`RSA`

`ElGamal encryption // LAB`

Signature schemes

`DSA`

`ECDSA`

`ElGamal signature`

`Schnorr signature`

Typical use-case (SSL)

- RSA/ECDSA to authenticate
- then goes for symmetric cipher
- and eventually takes advantage of AES offloading

Name a few key agreement algorithms…

Key exchange algorithms

```
DH
DHE (PFS / ephemeral)
ECDH
ECDHE (PFS / ephemeral)
```

Note it's also possible to simply encrypt the secret and send it to Alice

CIA triad / quadrad / polyad

- Confidentiality
- Integrity
- Availablility
- (Non-repudiation)
- Authenticity (Authentication)
- (Accountability)

Apply secure channels & crypto to those concepts

…which one leverages a secret (symmetric cipher)?

…which one leverages public key cryptography?

…and which one protects against MITM?

==>

- Confidentiality –> symmetric encryption & key agreement (DH)
- Integrity –> hash function (possibly using privkey)
- Availablility
- (Non-repudiation)
- Authentication –> public key crypto
  - –> auth with private key / sign
  - –> and also used for key agreement (RSA)
- (Accountability)

How to authenticate / is any public key or certificate fine?…

==> you need a Trust Anchor

Is it a BI-DIRECTIONAL process?…

==> you need to authenticate both sides if server is not public

Client authenticates server

- PKI chain of trust e.g. HTTPS
- Certificate pin on pubkey or fingerprint
- TOFU on pubkey or fingerprint
- DNSSEC chain of trust

// Questions on practical cryptography?

Sniffing Methods

- Assuming a LAN, NOT A WLAN

What is promiscuous mode?…

==> catch and show everything passing by, even if it’s not for you

Turn the network interface into promiscuous mode manually

```
ifconfig eth0 promisc
ip link set eth0 promisc on
```

or let some sniffing tool do it for you

```
tcpdump -i eth0
wireshark -i eth0
tshark -i eth0 -Y 'FILTER-HERE'
tshark -i eth0 -2 -R 'FILTER-HERE'
```

or some IDS/IPS system

```
snort
suricata
zeek
```

or some other detections tools

```
arpwatch
```

Assuming you are just a node on the network, not a router

Assuming you're connected to a SWITCH (not a HUB)

What can you see there?…

==> Not much

However you get to obtain far more information compared to a network scan/discovery

That's normal

L2 broadcast

- ARP request
- IPX RIP request (Kyocera)

L2 pure-network stuff

- LOOP
- STP
- and more…

IP6 link layer bcast

- ICMPv6 neighbor solicitation
- ICMPv6 neighbor advertisement
- Apple Zone Information Protocol request

That’s also normal

L3 multicast

- VVRP announcement
- OSPF multicast Hello
- MDNS Multicast/zeroconf DNS
- LLMNR Win+Ubu vs DNS

IP6 internet layer mcast

- LLMNR Win+Ubu vs DNS
- MDNS Multicast/zeroconf DNS
- DHCPv6 (dst ff02::1:2)

L3 broadcast

- DB-LSP-DISC (Dropbox discovery udp/17500)
- NBNS (NetBIOS udp/137)
- BROWSER (NetBIOS udp/138)

That's NOT normal

and some stuff you should not even see (switch already flooded or misbehaving)

- L2 mac to mac MS NLB cluster
- L3 ip/lan to ip e.g. TCP, ISAKMP, SIP
- L3 ip/pub to ip e.g. TLSv1.2
- L3 OSPF
- L3 Cisco NetFlow

(in short, ARP, broadcast, multicast)

Because we're not on the path (not even passively)

From a network administrator perspective

In what other situation would we get to sniff more traffic?…

==> ways to *be there already*

You own the routers & gateways

- ONLY THE TRAFFIC GOING THROUGH

You own the switches and can port-mirror

- PASSIVE ONLY

From an intruder point of view who needs get there on the path

How to otherwise get to be there on the path, and sniff more?…

==> ways to *get there on the path*

Active-capable (MITM-ready)

- Rogue DHCP
- ARP spoofing
- DNS spoofing
- Rogue Wifi AP and 2G BTS

Passive-only (Eve-friendly)

- MAC Flooding

// Questions on sniffing methods?

Spoofing Methods

How to ARP spoof?…

Tools for ARP spoofing

`dsniff/arpspoof`

`ettercap/bettercap`

3 main use-case examples

==> Brutal all/all

```
arpspoof -i $netif
```

==> Semi-Brutal target/all

```
arpspoof -i $netif -t $ip
```

==> Classic target/peer-or-gw and *both-ways*

```
arpspoof -i $netif -t $ip -r $gw
```

Advanced use-cases & fine-tuning

==> Multi-target/gateways

- Want to target two hosts/services at once
- -or- got two reverse-proxies to spoof against one same end-point

==> In any case…

- Exclude monitoring & IDS/IPS hosts
- (Unless victim has `arpwatch` on a port-mirror – would catch it anyway?)

Let's become active

What is MITM again?…

==> MITM is…

- relay packets from one end-point to another, without those noticing
- i.e. IP forwarding == passive MITM
- messing up the packets == active MITM
- …(messing with communication protocol, key exchange, …)
- …(essential point: encrypted does **NOT** mean authenticated)

Remain unnoticed (do it before ARP spoof…)

```
sysctl -w net.ipv4.ip_forward=1
echo 0 > /proc/sys/net/ipv4/conf/eth0/send_redirects
```

- We need to forward
- …otherwise we could not even call it in the “middle” of anything
- (that would be just impersonating)

Active MITM

Once ARP spoofed, you will are victim's gateway — and here's where we become active at layer 3

```
iptables -t nat -A PREROUTING -p tcp --dport 443 \
    -j REDIRECT --to-port 33231
```

- Can be even more fine-tuned with src and dst
- Hence you need PREROUTING to taint the traffic on some *exotic* port

Eve on clear-text

All you can read is the clear-text so far

- Clear-text passwords w/ `dsniff/dsniff`
- HTTP full URLs w/ `dsniff/urlsnarf`
- Specific clear-text you want w/ `ngrep`
- Ettercap/Bettercap (what plugin?)

What about HTTPS, SSH, …?

==> Encrypted communications (WE ARE BLIND)

However we got even more information than before now, since we got to sniff more traffic.

It's time to find, identify and define victims! (services or client communications)

- What kind of services are there and for what purpose?
- To what peers or gateway are those involved with? (what stream we need to get into)

Note – gateway can be found by looking at L2

Examples

- Target a precise workstation in a user network
- Target a non-secured reverse proxy ***back-end*** in the DMZ

// Questions on spoofing methods?

Now what packets to mess-up with and become active above L3?…

==> Depends on network segment

- VLAN with front-facing IPs
- DMZ
  - NAT port forwarding
  - Reverse-proxies
- User network
- VoIP
- Backup or mgmt network

What protocols are MITM-happy, meaning we can mess with it?…

==> IMPERSONATE **AND** TAMPER MITM-HAPPY PROTOCOLS

- HTTP
- DNS
- DNSSEC?
- SMTP
- ESMTP
- ESMTPS

Examples

- HTTP tamper by stripping the `S` out of `HTTPS` links (`sslstrip` and other tools)
- DNS spoofing
- ESMTP tamper by stripping `STARTTLS` out during a session (`striptls` or DIY)
- ESMTPS impersonate with your own self-signed cert (it will pass through)

What happens with truly established HTTPS sessions?…

==> SOMEHOW SECURE

SSL/TLS DOES HANDLE AUTHENTICITY / AUTHENTICATION

YOU CANNOT EASILY GET INTO AN HTTPS STREAM

However we've got far-more information than with IP accounting already

- we are below l7 but got more meta-data…

// Questions on spoofing methods?

Practical Security Tips & Tricks

- remote sniffing
- read an X.509 certificate

Super-duper remote sniffing

```
ssh -l root GOT_MIRROR \
"/usr/sbin/tcpdump -n -e -i eth3 -s0 -w - " \
| wireshark -k -i -
```

```
-e also show MAC addreses
-s0 backward compatible w/ now default packet snapshot
    length of 262144 bytes
```

- The interface is plugged to a dedicated port-mirror here
- …otherwise need to filter out ssh itself)

Read an X.509 certificate

```
openssl x509 -in domain.crt -noout -text
```

This is the end