Rootkits

How deep the rabbit hole is?

DCTF25, March 2025

Matej Kovačič, https://telefoncek.si

Rootkits

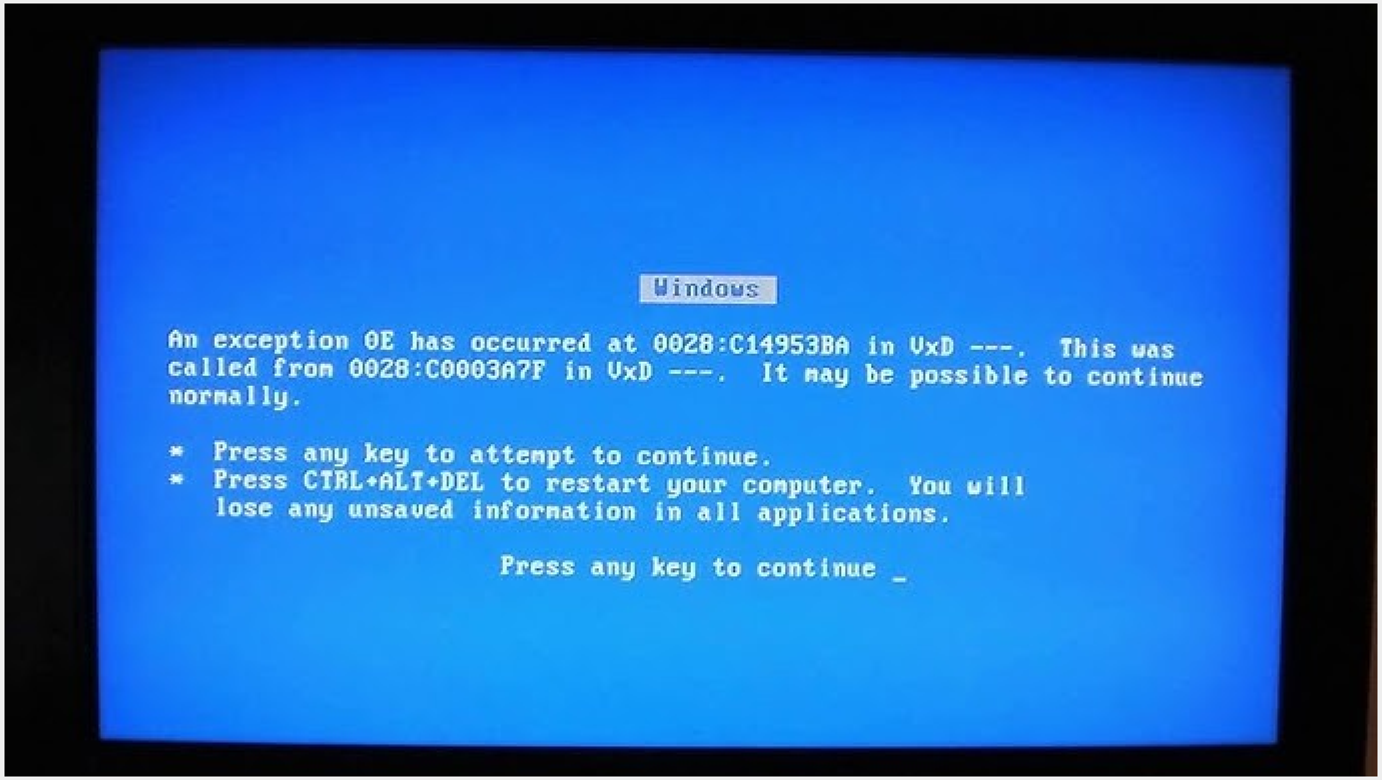
Rootkit: a malicious software designed to gain unauthorized access to a computer system and hide their presence.

Typically they enable remote file execution, system configuration changes, can log keystrokes or network activity and other forms of spying on user activities. They can hide files, processes, disable security policies, etc.

Bootkit:a specific type of rootkit, designed to infect a computer and to load their malicious code into memory before the operating system initializes.

By targeting the pre-boot environment, bootkits can bypass standard security measures, remain hidden and often have the ability to survive reinstallation of an operating system.

The first bootkit



```
Windows

An exception 0E has occurred at 0028:C14953BA in VxD ---.  This was
called from 0028:C0003A7F in VxD ---.  It may be possible to continue
normally.

*  Press any key to attempt to continue.
*  Press CTRL+ALT+DEL to restart your computer.  You will
   lose any unsaved information in all applications.

                Press any key to continue _
```

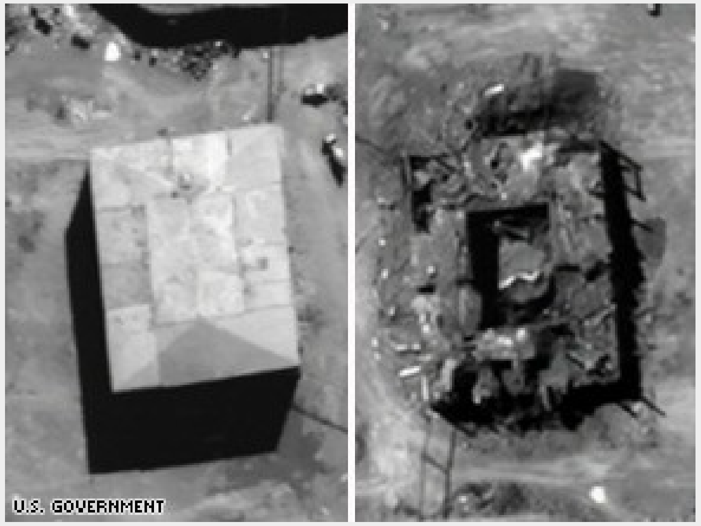
The first bootkit was **CIH (Chernobyl virus)**, which appeared in **1998**. It was developed by a Taiwanese student Chen Ing-hau (hence the name CIH) and targeted Windows 9x systems. The malware corrupted the MBR and overwrote parts of the BIOS, rendering the system unbootable.

Evil Maid attack

The technology of bootkits is often implemented in various governmental and commercial remote surveillance tools.

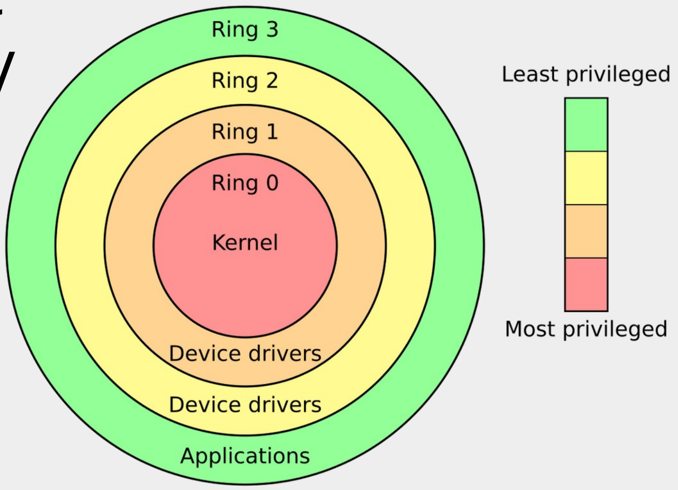
One of the reasons is, that bootkits can be used to bypass encryption, typically by intercepting passwords or encryption keys entered during boot.

- Joanna Rutkowska, Evil Maid proof-of-concept tool, 2009.
- Al-Kibar nuclear facility and Operation Orchard (also known as Operation Outside the Box) on September 2007.



Rootkits and protection rings

Protection rings, also called **hierarchical protection domains**, are mechanisms to protect data and functionality from faults (by improving fault tolerance) and malicious behaviour.



Rings in computer systems are arranged in a hierarchy from most privileged (most trusted, usually numbered zero) to least privileged (least trusted, usually with the highest ring number).

Modern operating systems are using only **Ring 0** (*kernel mode*) and **Ring 3** (*user mode*).

However, there are even lower rings...

Ring 3 rootkits

Ring 3 rootkits, also called **user mode rootkits** are running at the user-space level.

They run with the lowest level of privileges within the operating system and can perform a damage at the user space of the infected user.

Example, spyware running as a user program in Ring 3 should be prevented from turning on a web camera without informing the user, since hardware access is reserved for Ring 0 (*kernel mode*).

Ring 3 rootkits mitigation

Since Ring 3 rootkits do not have kernel-level access they are easier to detect and remove.

Good strategies:

- **application sandboxing** (in order to minimize damage within the infected user space) and
- **regular backuping/snapshoting** of the complete user space in order to prevent complete data loss.
- using **different isolated environments** for different tasks.

Ring 0 rootkits

Ring 0 rootkits, known also as **kernel mode rootkits** reside in the core of the operating system (so called *kernel space*).

They:

- have the highest level of privileges *within the operating system*;
- could be deeply integrated into the operating system (can hide files, processes, or network activity and modify system calls);
- are usually hard to remove (requires specialized tools or reinstalling the complete operating system).

Ring 0 rootkits mitigation

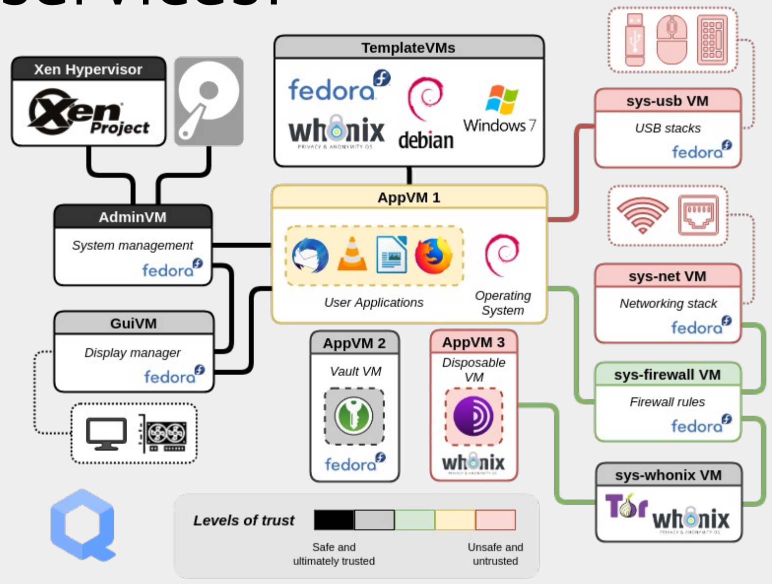
Good strategies against them:

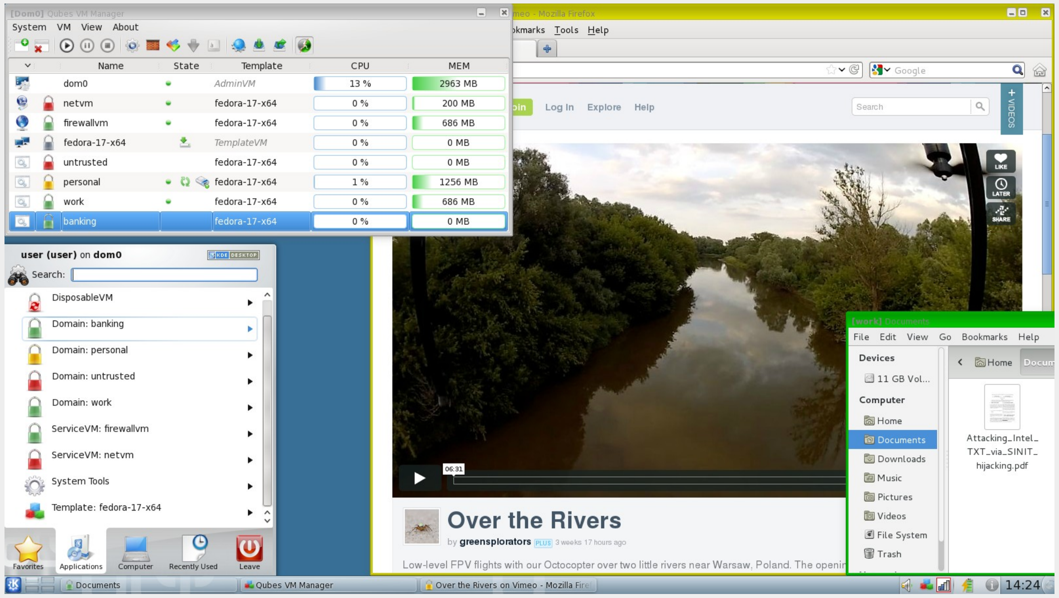
- running the operating system in virtual compartment;
- use different isolated environments for different tasks;
- perform virtual machine level snapshots.

In that case virtualisation technology provides the isolation of different virtual machines.

QubesOS

QubesOS is a security focused desktop operating system that provide security and segmentation of applications through isolation with virtualization services.




[Dom0] Qubes VM Manager

Name	State	Template	CPU	MEM
dom0	●	AdminVM	13 %	2963 MB
netvm	●	fedora-17-x64	0 %	200 MB
firewallvm	●	fedora-17-x64	0 %	686 MB
fedora-17-x64		TemplateVM	0 %	0 MB
untrusted		fedora-17-x64	0 %	0 MB
personal	●	fedora-17-x64	1 %	1256 MB
work	●	fedora-17-x64	0 %	686 MB
banking		fedora-17-x64	0 %	0 MB

The user's digital life is divided into security domains with different levels of trust. Unfortunately, QubesOS is quite complex and is less suitable for regular users, because it has quite steep learning curve.

And that is all...

...or not?

Ring -1 rootkits

Ring -1 rootkits, also known as **hypervisor rootkits** operate at the *hypervisor level*, below the operating system.

Basically they create virtual environment and confine operating system into it, while the compromised operating system believes it is running directly on the hardware.

They are extremely challenging to detect (and remove), because they can manipulate the operating system from outside its own context.

BluePill

Joanna Rutkowska, 2006: concept of a Ring-1 malware, called Blue Pill.

It exploited virtualization extension AMD-V.

Is able to place the operating system in a virtualized environment without the operating system being aware of it. OS thinks it is operating on bare-metal hardware, in reality, it's running in a hypervisor and is being being monitored and manipulated by the Blue Pill rootkit.

Other researchers have shown, that Intel VT-x virtualisation extension could also be exploited.

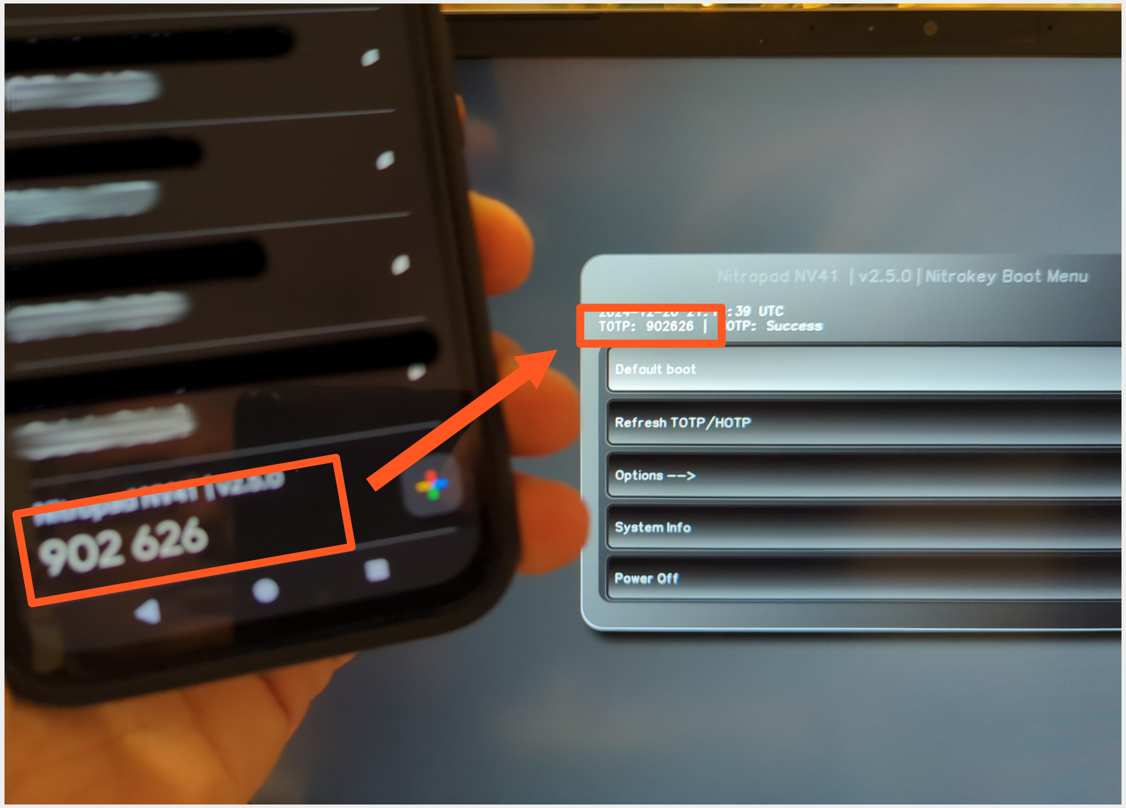
Ring -1 rootkits mitigation

One solution against Ring-1 rootkits is to **disable hardware virtualization in BIOS/UEFI**.

However in that case user will be limited running virtual environments on their system.

Ring -1 rootkits mitigation

Another solution is to use **trusted boot mechanisms**. Those mechanisms perform hypervisor integrity checks (by verifying its cryptographic signature) and can help to ensure that unauthorized hypervisors cannot load during the boot process.


902 626
Nitropad NV41 | v2.5.0 | Nitrokey Boot Menu
TOTP: 902626 | HOTP: Success
Default boot
Refresh TOTP/HOTP
Options -->
System Info
Power Off

Ring -2 rootkits

Ring -2 rootkits are a mix of so called **SMM rootkits** and **BIOS/UEFI bootkits** (also called **UEFI implants**).

Usually Ring-2 rootkits utilize SMM and UEFI compromise. While SMM rootkits operate dynamically within the CPU’s SMM environment they usually use UEFI rootkit technology to embed malicious code in the firmware layer to achieve persistence.

No surprise: SMM and UEFI contain security vulnerabilities and are loaded with lot of bloat (for instance some SMM's contained complete USB stack).

SMM rootkits

SMM rootkits run at the **System Management Mode (SMM)**. SMM operates in a protected memory space called SMRAM (System Management RAM), which is inaccessible to the operating system and most security tools.

SMM is the most privileged mode in the modern x86_64 processors. SMM can directly interact with hardware, it is bypassing the operating system and hypervisors. Also, it cannot be interrupted by normal hardware/software interrupts.

All this allows completely stealth code execution!

SMM rootkit examples

Shadow Walker by *Sherri Sparks* and *Jamie Butler* in 2005:

- was capable of hiding both its own code and changes to operating system's components and was able to fool both signature and heuristic based scans.

SMM rootkit by *Shawn Embleton, Sherri Sparks* and *Cliff Zou* in 2008:

- a chipset level keylogger and a network backdoor capable of directly interacting with the network card to send logged keystrokes to a remote machine via UDP.

SMM rootkit examples

Injecting shellcode from SMM to a Ring0/Ring3 context by *Jussi Hietanen* in 2020:

- capability to infect a Windows usermode process, access the full memory space and persist between OS reinstalls.

BIOS/UEFI rootkits

Another class of Ring -2 rootkits is called **BIOS/UEFI rootkits**, because they specifically target the BIOS (*Basic Input/Output System*) or its modern equivalent, UEFI (*Unified Extensible Firmware Interface*).

- The first rootkit/bootkit targeting BIOS was **CIH (Chernobyl virus)** in 1998.
- **IceLord** proof-of-concept bootkit (**ICLord Bioskit**) in 2007 demonstrated that BIOS rootkits were feasible and powerful.
- **Rakshasa** by Jonathan Brossard in 2012 - proof-of-concept firmware rootkit was able to persist in UEFI/BIOS firmware.

BIOS/UEFI rootkits

- Probably the first known Ring -2 rootkit used in the wild was **Mebromi**, discovered in 2011.
- *Andrea Allievi* who developed one of the first UEFI bootkit concepts (for Windows 8) in 2012.In 2013, *Sebastien Kaczmarek* from *Quakerslabs* presented **Dreamboot** (probably based on Allievi's work, also targeting Windows 8).

BIOS/UEFI rootkits

Rootkit/bootkit technology is often used by government spying tools and for cyberespionage.

- Hacking Team group tool **Remote Control System** infected UEFI/BIOS to keep their malware persistent (2015).
- **FinSpy** (also known as **FinFisher** or **Wingbird**), is also used for cybersepionage, in 2021 they employed UEFI bootkit technology.
- **MoonBounce** rootkit (discovered in 2021 and linked to Chinese APT41 hacker group) injected its malicious code into the SPI flash chip on computer motherboard, targeting UEFI firmware.

BIOS/UEFI rootkits

- **LoJax**, discovered in 2018, embedded itself into UEFI firmware to execute at system startup and was operating in SMM, bypassing OS-level detection. LoJax can persist in the UEFI even if the operating system is reinstalled or its hard drives are replaced. Used for track the system's location, remotely access the system and install additional malware on it. LoJax targeted organizations in the Balkans and countries in Central and Eastern Europe.

BIOS/UEFI rootkits

UEFI bootkits are not exclusively targeting Windows.

In 2012 a security researcher Loukas K., "snare", presented **Mac EFI rootkit**.

In 2017 Wikileaks published information about CIA's Vault 7 hacking tools, containing Mac OS X EFI implant, **QuarkMatter**.

QuarkMatter used an EFI driver stored on the EFI system partition to provide persistence to an arbitrary kernel implant.

BIOS/UEFI rootkits

- In 2024 the first UEFI bootkit designed for Linux systems appeared, named Bootkitty. It was a proof of concept tool that disables the kernel's signature verification feature to load unsigned boot code.
- **BlackLotus** (discovered in 2022) was first that integrated **Secure Boot bypass** and is probably the first UEFI rootkit that was "commercially" sold on cybercrime forums. It also implemented several detection evasion features, for instance code obfuscation, anti-virtualization, disabling Windows Defender antivirus software, bypassing User Account Control (UAC), etc.

What about Secure Boot?

Secure Boot is a security standard to help make sure that your PC boots using only software that is trusted (*by the PC manufacturer*).

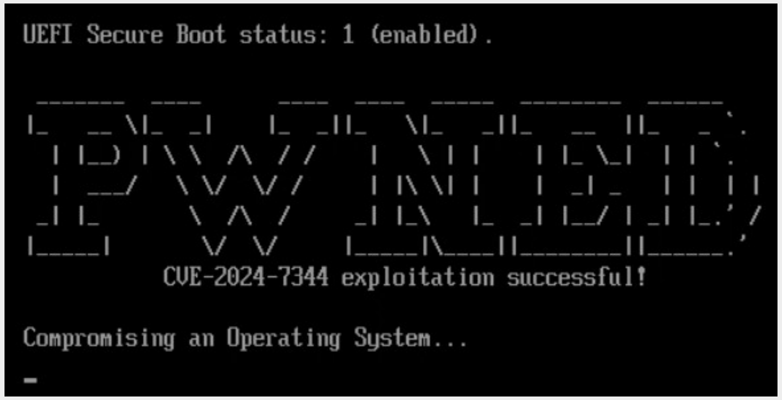
See the problem here?

Unfortunately, **Secure Boot can be bypassed**:

- LogoFAIL attack;
- "Baton drop" Secure Boot bypass;
- Eclypsium bootloader;
- "Backdoor" vulnerability that allowed **disabling** Secure Boot in Lenovo and Acer laptops (probably it was a debug feature);

What about Secure Boot?

- CVE-2024-7344, that enabled an attacker to **load any UEFI binary**, even an unsigned one and regardless of the UEFI Secure Boot state.



```
UEFI Secure Boot status: 1 (enabled).

                CVE-2024-7344 exploitation successful!

Compromising an Operating System...
_
```

- in 2024 security researchers found out, that Secure Boot was completely compromised on more than 200 device models sold by Acer, Dell, Gigabyte, Intel, and Supermicro, because someone **mistakenly published the cryptographic key** that forms the root-of-trust anchor between the hardware device and the firmware that runs on it (so called *Platform Key*).

Ring -2 rootkits mitigation

Enable hardware protections like **BIOS lock** (a security feature designed to prevent unauthorized access when the computer is booting) and **SMM lock** (hardware protection to prevent unauthorized access to SMRAM).

Using **Secure Boot** to prevent unauthorized firmware or bootloader modifications is also an option, however some rootkits (for instance *BlackLotus* and *Bootkitty*) can **bypass** Secure Boot protection.

Prevent physical access to the system, because BIOS/UEFI rootkits could also be installed via direct hardware access (this requires special hardware device called BIOS firmware programmer).

However...

Security researchers found several vulnerabilities in closed source BIOS firmware code.

- LogoFAIL attack.
- Secure Boot and Intel Trusted Boot in traditional BIOS'es are vulnerable to rollback attack.
- In 2024 security researchers found out, that **Secure Boot was completely compromised** on more than 200 device models sold by Acer, Dell, Gigabyte, Intel, and Supermicro, because someone **mistakenly published the cryptographic** key that forms the root-of-trust anchor between the hardware device and the firmware that runs on it (so called platform key).

Fortunately...

- System Management RAM locking.
- SMM BIOS write protection.
- BIOS/UEFI lock.
- Secure Boot ***and*** Measured Boot to detect unauthorized changes to firmware and SMM code (and also operating system's boot scripts!).
- Utilizes external hardware security module for verification of system integrity (firmware, kernel, and bootloader).

Ring -3 rootkits

Ring -3 rootkits operate in the **Management Engine** (ME) or **Platform Controller Hub** (PCH) firmware, such as **Intel's Management Engine (ME)** or **AMD's Platform Security Processor (PSP)**.

These are embedded microcontrollers **within the CPU** chipset, designed for out-of-band system management and security features. Since those rootkits reside in firmware, they are also called **firmware rootkits**.

They can access host memory via DMA (direct memory access), they can directly access network interface, can boot the system from the emulated CDROM and are active even in so called S3 sleep (*System Power State S3*).

Ring -3 rootkits

Ring -3 rootkit concept was first presented by Alexander Tereshkin and Rafal Wojtczuk in 2009.

They found out that Intel vPro chipsets had an **independent CPU**, access to dedicated DRAM memory, special interface to the network card and execution environment called Management Engine (ME).

Intel Q35 chipset had a **standalone web server**.

So this is a little computer inside computer, that can execute programs independently from the main CPU.

Ring -3 rootkits

Intel ME / Intel Active Management Technology (AMT) is exploitable:

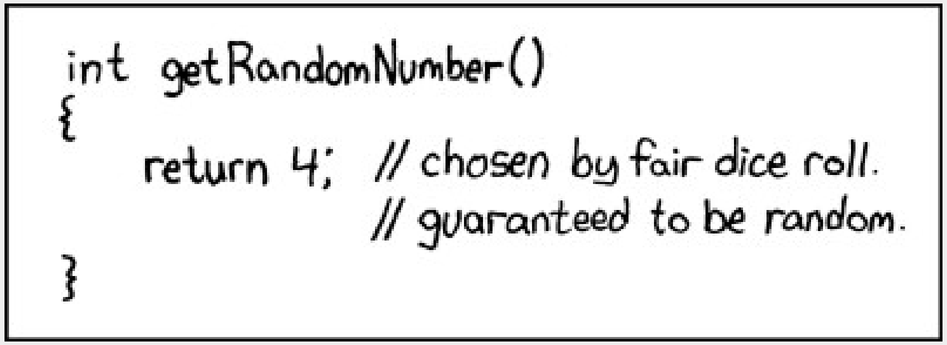
- In 2010 Vassilios Ververis described several fundamental security weaknesses in Intel's AMT that allow the attacker to **remotely control the target machine** (over the Internet or a mesh networking) and enables the installation and control of a botnet on the hardware level.
- In 2017 Mark Ermolov and Maxim Goryachy presented a talk titled *How to Hack a Turned-Off Computer, or Running Unsigned Code in Intel Management Engine*, where they have shown how to **execute unsigned code even on a powered-down system** by exploiting Intel ME.

Ring -3 rootkits

- "**Silent Bob is Silent**" - CVE 2017-5689 from 2017 - allowed an attacker to gain system privileges remotely (through the Internet). This vulnerability was present in Intel CPUs from 2008 (**9 years**).
- In June 2017, the **cybercrime group PLATINUM** started to exploit Intel's AMT Serial-over-LAN functionality, which allows them to remotely access computers, bypassing the host operating system and its firewalls. The cybercrime group exploited AMT to perform data exfiltration of stolen documents.
- in June 2022, the **Wizard Spider ransomware group**, developed proof-of-concept code targeting Intel firmware to carry out persistent, hard-to-detect attacks.

Ring -3 rootkits

- In February 2025 Google Security Team published details about **AMD Microcode Signature Verification Vulnerability** (for Zen 1 through Zen 4 CPUs). It allows an adversary with local administrator privileges to load malicious microcode patches. Vulnerability can compromise *AMD Secure Encrypted Virtualization* (and *SEV-SNP extension* that provides additional protection against side-channel attacks) and *Dynamic Root of Trust Measurement*.
- They created a PoC that makes the RDRAND instruction always return number 4.



```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

Source: https://xkcd.com/221/

Ring -3 rootkits mitigation

A possible mitigation (for specific Intel CPUs only) is **to disable ME functionality**.

(Not completely, because this would destroy the CPU).

ME Cleaner:

- HECI method (soft-disabling; Host Embedded Controller Interface), which is not fully trusted by the security community and it also only partially disables Intel ME.
- HAP disabling method - sets a special HAP bit that acts like a kill-switch. This method completely turns off all Intel ME parts that can be disabled.

Ring -4 rootkits

Ring -4 rootkits are more theoretical, however there are some proofs that they can be successfully deployed.

The term Ring -4 is used to describe emerging threats in the privilege hierarchy below known Ring -3 systems.

Those rootkits would target components even deeper within the system, such as the System on Chip (SoC) or physical hardware devices themselves.

Ring -4 rootkits

Exploits on baseband processors:

- In 2011 Ralf-Philipp Weinmann has shown how to set up fake base station, attract nearby phones to join the fake network, where he was then able to inject a malicious firmware update into the baseband processor. His malicious firmware would then switched on the phones' auto-answer feature, which would have let the researcher to **silently dial into the phone and remotely listen to nearby conversations**.
- In 2019 two vulnerabilities called QualPwn (CVE-2019-10538 and CVE-2019-10540) impacted devices with Qualcomm chipset. They allowed the attacker to **remotely run code with kernel privileges on the target device**. Attack was carried out through WiFi.

Ring -4 rootkits

Exploits on baseband processors:

- Simjacker vulnerability (2019) allows the attacker to send a special crafted SMS to the victim's device, which instructed the SIM card within the phone **to take over the mobile phone and perform sensitive commands**. The Simjacker attack was exploited by surveillance companies for cyberespionage operations.

Ring -4 rootkits

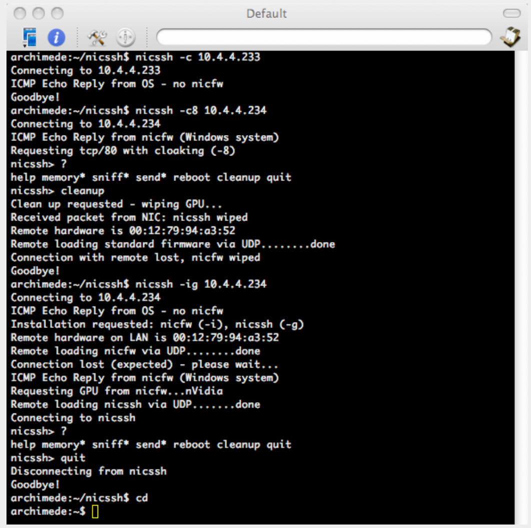
Exploits on storage controllers:

- Jeroen Domburg (in 2013) - **malware on a hard disk controller** that was able to modify data when reading from the hard disk. He demonstrated how to "inject" replacement password to a target system. Malware is activated with »magic string«.
- Marcus Hutchins (in 2015), who created a firmware rootkit that could be stored on hard drive’s memory chip, and can intercept and modify data being sent back to the host computer. This allows the rootkit to **trick the host system into executing arbitrary code**.
- IRATEMONK (NSA’s exploit tool) provided software application persistence by implanting the malware in the hard drive firmware to **gain execution through Master Boot Record (MBR) substitution**.

Ring -4 rootkits

Exploits on network interface cards:

- Arrigo Triulzi, 2008, Project Maux Mk.II. Proof-of-concept hardware rootkit on a network card, called NIC SSH. The tool allows him to connect directly to compromised network, completely bypassing the operating system (and the firewall) to access the computer.



```
archimede:~/nicssh$ nicssh -c 10.4.4.233
Connecting to 10.4.4.233
ICMP Echo Reply from OS - no nicfw
Goodbye!
archimede:~/nicssh$ nicssh -c8 10.4.4.234
Connecting to 10.4.4.234
ICMP Echo Reply from nicfw (Windows system)
Requesting tcp/80 with cloaking (-8)
nicssh> ?
help memory* sniff* send* reboot cleanup quit
nicssh> cleanup
Clean up requested - wiping GPU...
Received packet from NIC: nicssh wiped
Remote hardware is 00:12:79:94:a3:52
Remote loading standard firmware via UDP........done
Connection with remote lost, nicfw wiped
Goodbye!
archimede:~/nicssh$ nicssh -ig 10.4.4.234
Connecting to 10.4.4.234
ICMP Echo Reply from OS - no nicfw
Installation requested: nicfw (-i), nicssh (-g)
Remote hardware on LAN is 00:12:79:94:a3:52
Remote loading nicfw via UDP........done
Connection lost (expected) - please wait...
ICMP Echo Reply from nicfw (Windows system)
Requesting GPU from nicfw...nVidia
Remote loading nicssh via UDP.......done
Connecting to nicssh
nicssh> ?
help memory* sniff* send* reboot cleanup quit
nicssh> quit
Disconnecting from nicssh
Goodbye!
archimede:~/nicssh$ cd
archimede:~$ 
```

Other exploits on hardware components:

- Firewire interface,
- malware on a Apple Aluminium Keyboard,
- malware on a PCI card, etc.

Ring -4 rootkits mitigation

Depends on a type of a component that rootkit resides on.

In general:

- firmware validation (not feasible in practice),
- secure supply chain practices (also not feasible in practice),
- physical security (also not always feasible in practice),
- hardware components with open source and verified firmware (Guess what? Also not really feasible in practice).

Ring -4 rootkits mitigation

- Exploits on baseband processors could be mitigated by baseband isolation.
- Blob-free network cards for computers (non-modifiable pre-installed firmware that is part of the hardware).
- Malware on storage controllers could be defeated by software level full disk encryption (+data integrity algorithms).
- Keeping the operating system secure, can also help defending against firmware attacks (because malware can not communicate with host OS).
- For other hardware components the threat level should be evaluated.

Can we go deeper?

Processors with malicious design

Illinois Malicious Processor (presented in 2008), proof-of-concept research project demonstrating how malicious functionality can be embedded directly into a **processor's design**.

- an attacker can design a hardware to support general purpose attacks; malicious hardware design can bypass traditional software-based security mechanisms.
- Illinois Malicious Processor included a hidden operational mode, that was designed to be undetectable by traditional hardware and software monitoring tools. This mode allows the malicious processor to execute hidden instructions and access reserved parts of the cache memory for storing attack payloads.

Processors with malicious manufacturing

CPU manufacturing process:

- Silicon is **purified** to a high degree (99.9999%).
- It is **sliced** into thin wafers.
- Photolithographic and chemical processes are used to create the actual circuit on the silicon wafer.
  - The layer of **photoresist** (light-sensitive material) is applied to the silicon wafer
  - The circuit is **illuminated with UV** light through a photomask with a picture of a circuit.
  - Illuminated photoresist is hardened, while other parts of photoresist could be removed. This creates **image** of a circuit on the silicon wafer.
  - Exposed silicon is then **etched away** (chemically or with plasma).

Processors with malicious manufacturing

- **Doping**. Doping refers to the process of intentionally introducing impurities into a semiconductor to modify its electrical properties (create areas that can conduct or block electricity).
  - If elements of chemical **group V** (such as **phosphorus**), which have more electrons than silicon, are added to the silicon, the result is weakly bound and very mobile electrons. We get an **n-type semiconductor**.
  - If we dope silicon with elements chemical of **group III** (such as **boron**), we create a deficit of electrons, so we get **p-type semiconductors**.
- Finally, thin layers of materials like copper, aluminium, or insulating oxides are deposited on the wafer in order to get the multi-layered structure of the chip.

Processors with malicious manufacturing

Theoretical attack.

Theoretically doping could be used to introduce hardware vulnerabilities or even inject malware-like behaviour into a chip.

For instance, malicious actor could create regions in the chip with altered electrical properties. This might cause the chip to malfunction, leak data, or execute unintended instructions under specific conditions.

Doping could also be used to create hidden circuits that are not part of the original design. That would in fact create hardware Trojan on a chip.

Processors with malicious manufacturing

Practical attack.

In 2013 researchers have shown that hardware Trojans can be implemented completely undetectably **on consumer grade processors**:

- **Malformed random number generator**: they were able to arbitrarily reduce the range of random numbers from 2^128 to 2^32, the RNG **passed** the NIST test.
- **malicious hardware implementation of the AES encryption functions**, so that they were not resistant to a side channel attack any more. But - integrated circuit still performs its task - protecting against the ***all other*** side channel attacks. No functional testing can detect a hardware Trojan horse.

Processors with malicious manufacturing

Practical attack.

Those malicious hardware modifications could not be detected neither by optical inspection (the metal and polysilicon wiring of the modified chip is unchanged), or by performing a BIST test (build-in-self-test, a hardware self-testing process), or by checking with a reference chip, so called gold chip.

Also.

A similar process is already commercially used to obfuscate the operation of integrated circuits!

Mitigation?

General mitigation strategies:

- secure supply chains
- third-party verification

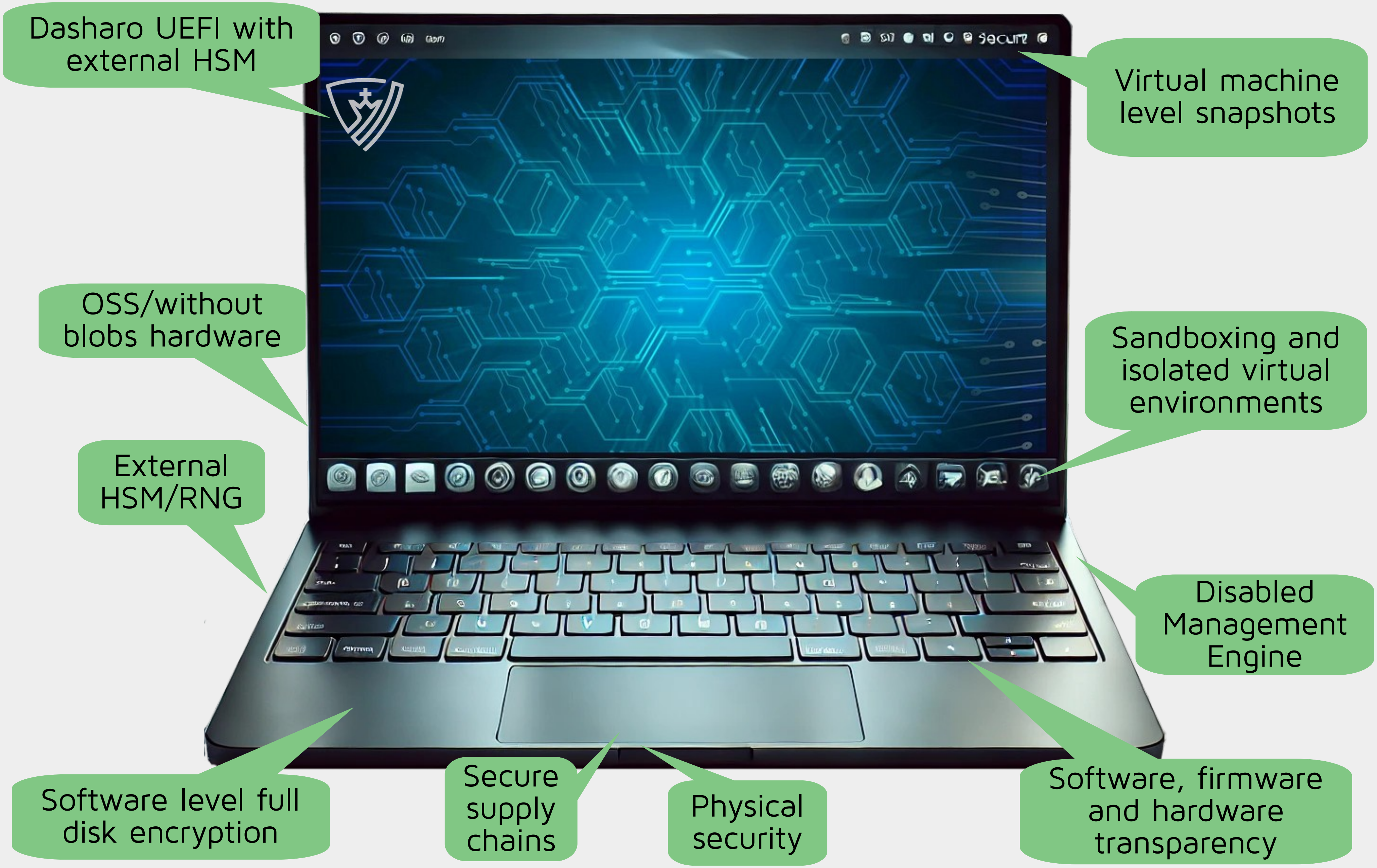
In reality: not really feasible.

However:

- external random number generators;
- hardware security modules for handling encryption keys.

What can be done?

Security hardening


Dasharo UEFI with external HSM
Virtual machine level snapshots
OSS/without blobs hardware
Sandboxing and isolated virtual environments
External HSM/RNG
Disabled Management Engine
Software level full disk encryption
Secure supply chains
Physical security
Software, firmware and hardware transparency

Questions?

Matej Kovačič

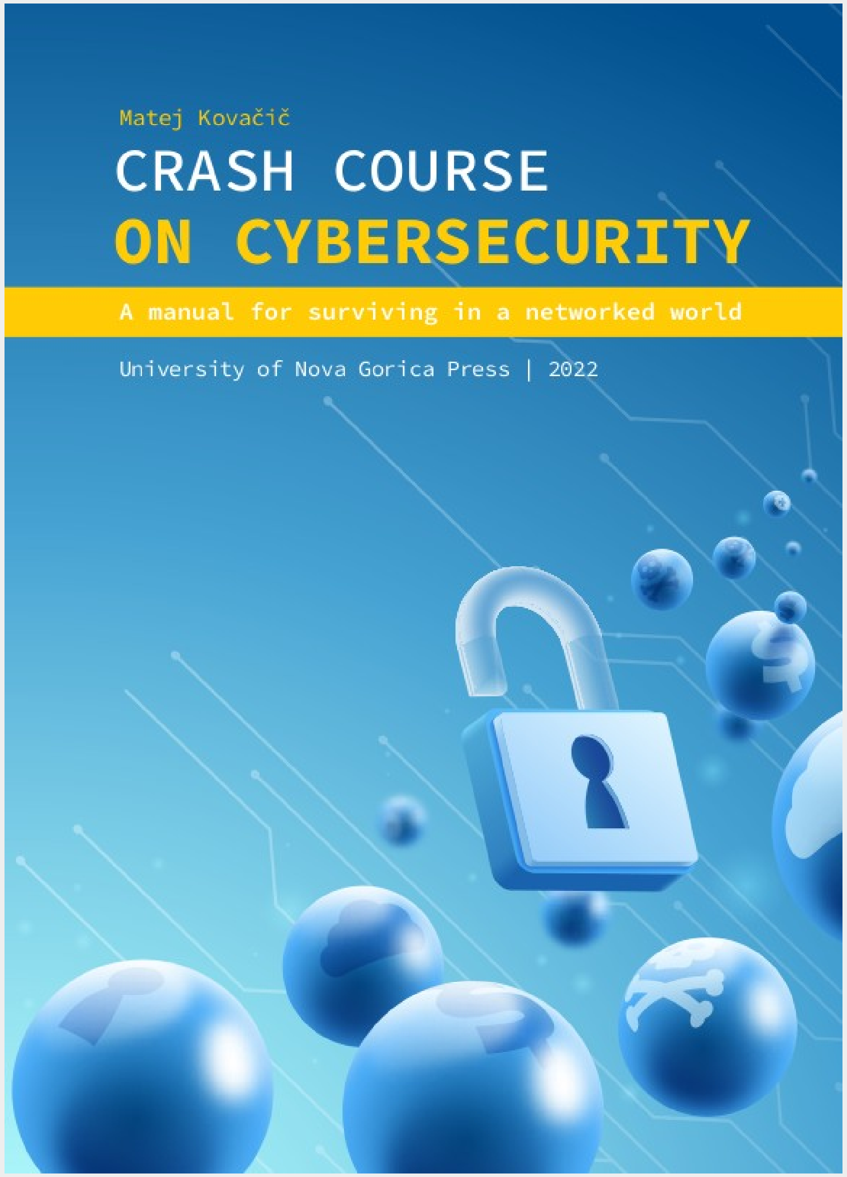
https://telefoncek.si

Some further reading...

Matej Kovačič. 2022. Crash course on cybersecurity: a manual for surviving in a networked world. ISBN: 978-961-7025-24-8 (PDF)

The book tries to explain the complex area of cybersecurity in an understandable way, to help to grasp the essential information on how to protect yourself and/or your company from cyberattacks and to provide technologically neutral advice for the implementation of protection against cyberattacks.

The book is available under a Creative Commons license and PDF is freely available online at <https://telefoncek.si>.


Matej Kovačič
CRASH COURSE
ON CYBERSECURITY
A manual for surviving in a networked world
University of Nova Gorica Press | 2022