Chip related security and availability

Markku-Juhani O. Saarinen
<markku-juhani.saarinen@tuni.fi>

2025-12-03 Tampere ENDR

“Military Grade Cryptography” ?

When I started as a cryptographer in the late 1990s (at SSH Communications Security), we avoided calling even SSH “dual-use” due to export controls..

Observations in 2025:

Everything about, say, **CNSA 2.0** is **public**. Anyone can implement these.

(CNSA 2.0 = A set of quantum-secure crypto algorithms, used up to TOP SECRET.)

Challenge: Making sure that the chips running your crypto don't leak secrets.

This has become very difficult (with ***bugdoors*** in advanced-node chips).

In the past: **SANLA M83/90** vs **Philips UA-8295**

Developed by NOKIA in the early 1980s.

Approximately all Finnish males ±10 years from my age know this device ..

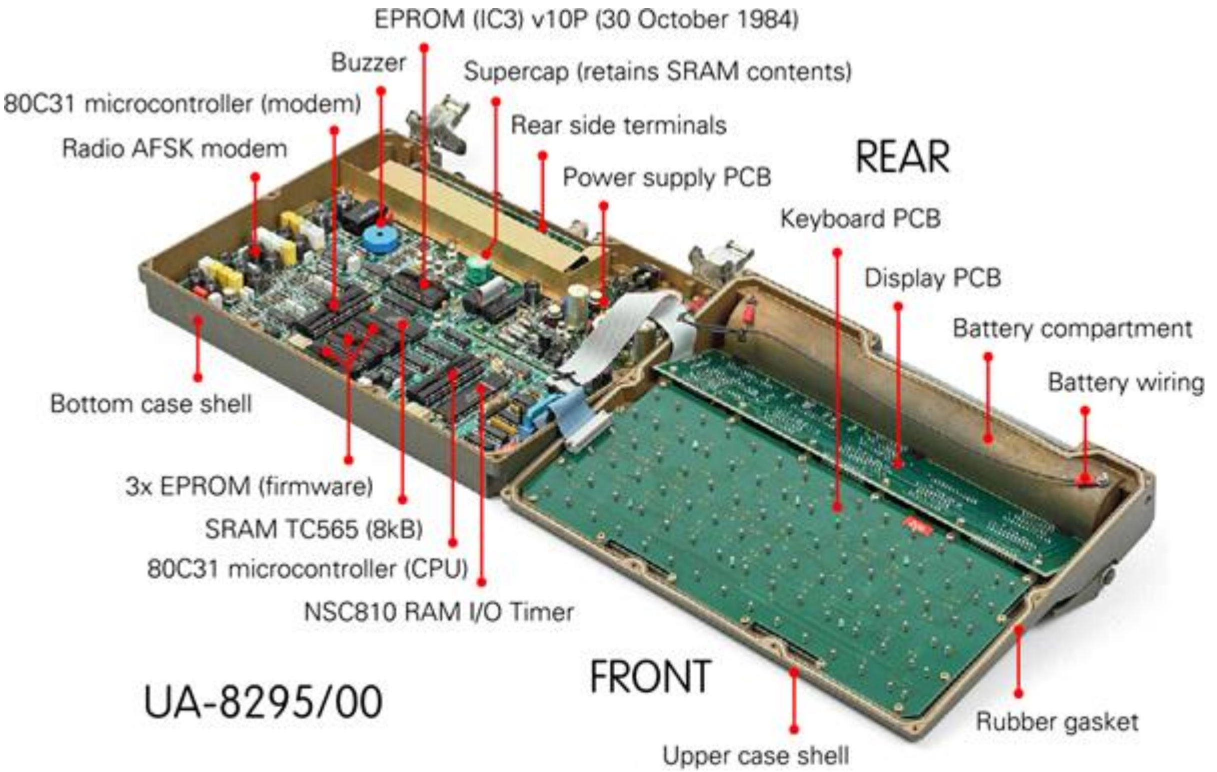
Licensed to Philips Usfa in 1984: UA-8295.

Standard version used **DES** in **OFB** mode.

Philips Export versions were modified to use even weaker encryption, called **SBT**.

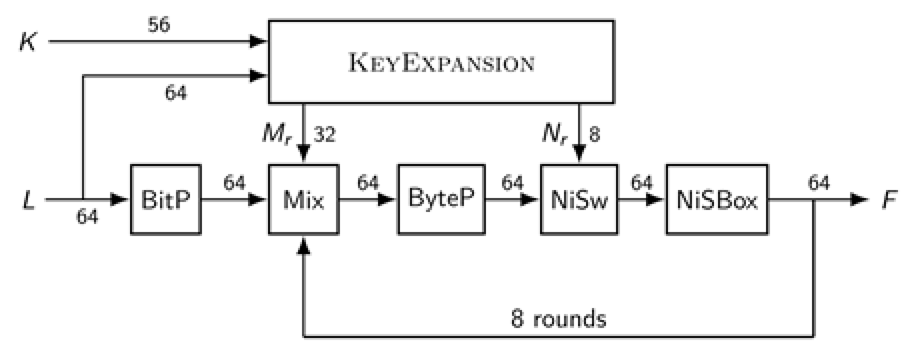
👉 Fun academic project to look at what a 1980s **NSA** crypto backdoor looked like!

Reversing 1980s crypto (with a screwdriver!)



Stijn Maatje & Marc Stevens (CWI, NL) 2025

- Interviewed former Philips Usfa Head of Advanced Development to confirm NSA - Philips deal. Had access to an UA-8295 with both “Nokia” and “NSA” Firmware.
- Disassembled code (Philips 80C31 MCU; Intel MCS-51 ISA) of modified ROM parts.
- Reconstructed mode-of-operation & NSA’s intentionally weakened **SBT cipher.**
- Found & documented several weaknesses, developed two attacks against SBT..

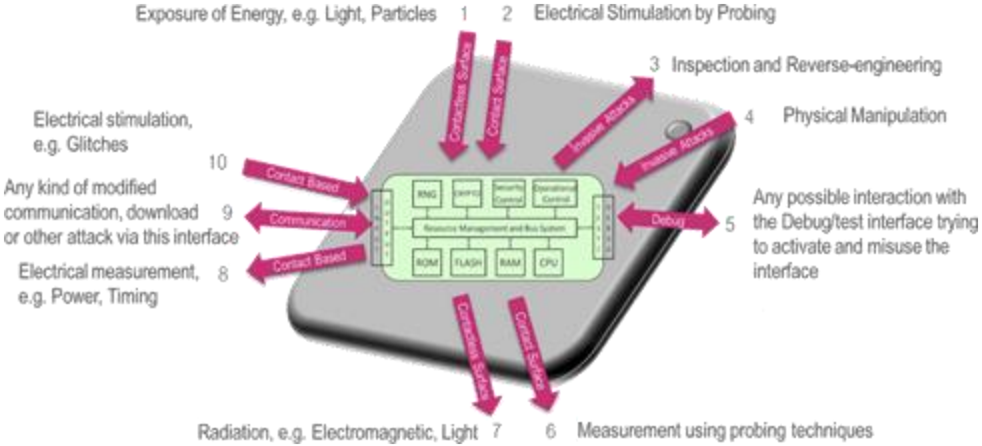


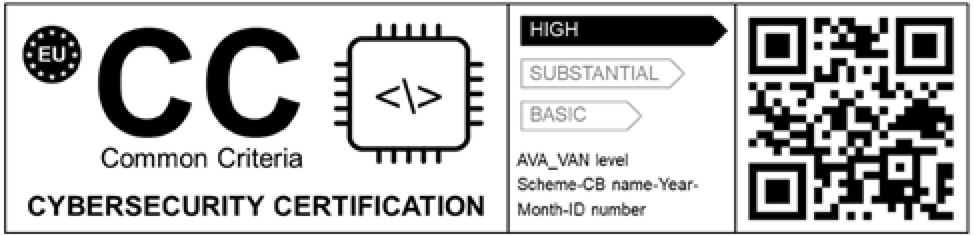
2025: AVA_VAN SoC/RoT Security Evaluation

Secure Elements are a strength for the European Semiconductor industry (ST, NXP, Infineon, etc.)

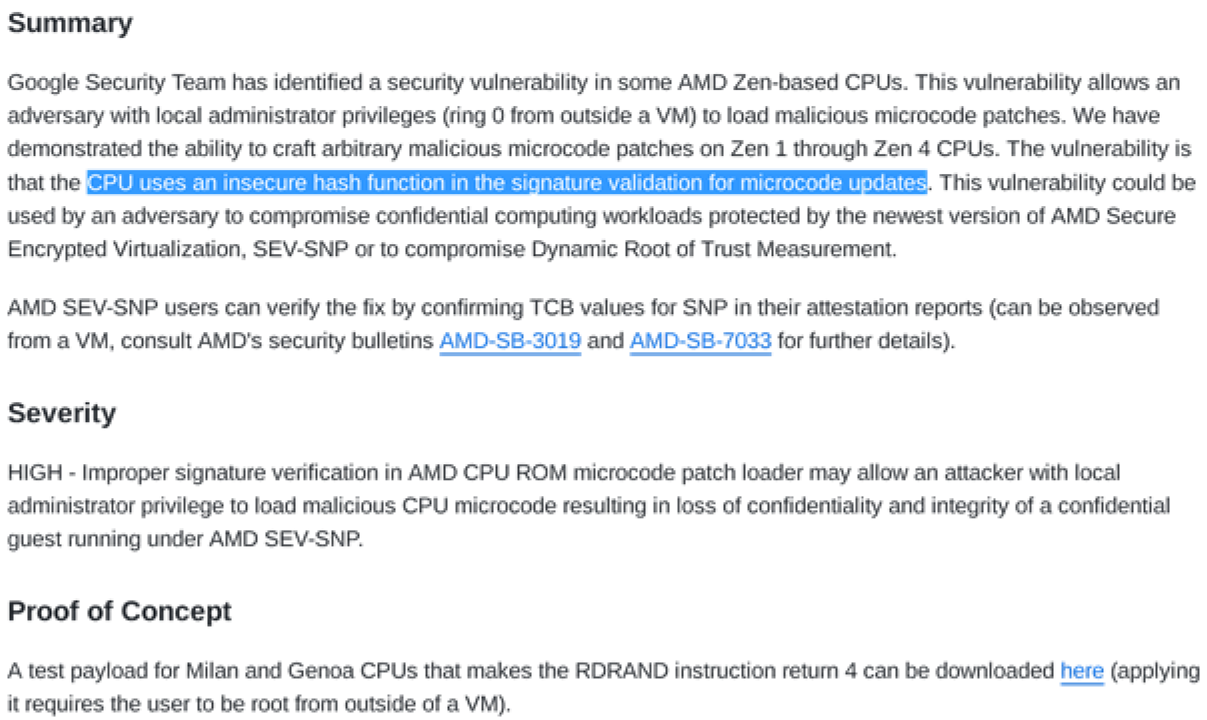
EUCC and its **AVA_VAN** is a rigorous “penetration test.”

EU Cyber Resilience Act (**CRA**) requires high AVA_VAN for Critical-class products (e.g. encryption products) by 2027.


Exposure of Energy, e.g. Light, Particles 1
2 Electrical Stimulation by Probing
3 Inspection and Reverse-engineering
4 Physical Manipulation
Any possible interaction with the Debug/test interface trying to activate and misuse the interface 5
6 Measurement using probing techniques
Radiation, e.g. Electromagnetic, Light 7
Electrical measurement, e.g. Power, Timing 8
Any kind of modified communication, download or other attack via this interface 9
Electrical stimulation, e.g. Glitches 10
Contactless Surface
Contact Surface
Invasive Attacks
Invasive Attacks
Debug
Contact Surface
Contactless Surface
Contact Based
Communication
Contact Based
RNG
CRYPTO
Security Control
Operational Control
Resource Management and Bus System
ROM
FLASH
RAM
CPU


EU
CC
Common Criteria
<\>
CYBERSECURITY CERTIFICATION
HIGH
SUBSTANTIAL
BASIC
AVA_VAN level
Scheme-CB name-Year-Month-ID number

But.. Case of AMD CPUs (CVE-2024-56161)



Summary

Google Security Team has identified a security vulnerability in some AMD Zen-based CPUs. This vulnerability allows an adversary with local administrator privileges (ring 0 from outside a VM) to load malicious microcode patches. We have demonstrated the ability to craft arbitrary malicious microcode patches on Zen 1 through Zen 4 CPUs. The vulnerability is that the CPU uses an insecure hash function in the signature validation for microcode updates. This vulnerability could be used by an adversary to compromise confidential computing workloads protected by the newest version of AMD Secure Encrypted Virtualization, SEV-SNP or to compromise Dynamic Root of Trust Measurement.

AMD SEV-SNP users can verify the fix by confirming TCB values for SNP in their attestation reports (can be observed from a VM, consult AMD's security bulletins AMD-SB-3019 and AMD-SB-7033 for further details).

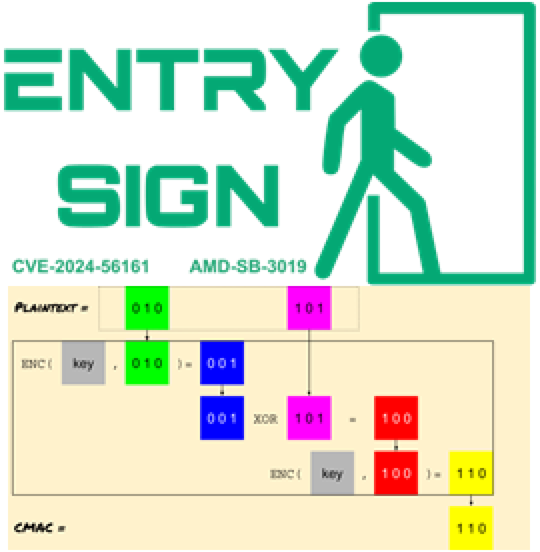
Severity

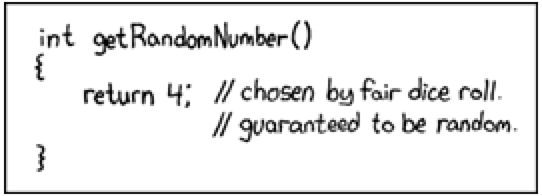
HIGH - Improper signature verification in AMD CPU ROM microcode patch loader may allow an attacker with local administrator privilege to load malicious CPU microcode resulting in loss of confidentiality and integrity of a confidential guest running under AMD SEV-SNP.

Proof of Concept

A test payload for Milan and Genoa CPUs that makes the RDRAND instruction return 4 can be downloaded here (applying it requires the user to be root from outside of a VM).

https://bughunters.google.com/blog/5424842357473280/zen-and-the-art-of-microcode-hacking





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

Let’s think about the AMD vulnerability

- An adversary can change the behavior of the processor in a way that is very difficult to detect in forensic analysis (we can’t trust attestation reports after microcode mod.)

- **Example:** Replace random bit generator with, say, AES in counter mode; impossible to statistically detect. But an adversary with key can predict any random bit / secret key.

- The “Zen” microarchitecture chips are in 14nm (Zen 1) to 3nm (Zen 5) – well beyond our reverse engineering abilities (SOTA 40nm with luck, also way too many transistors.)

- **Forever question:** How do we even know that this is a “bug” ? No professional cryptographer would use CBC-MAC as a hash function for digital signatures.

Provenance? Attack vectors in Supply Chain

Finding intentionally inserted security backdoors in modern semiconductors is near-impossible. One needs ways to control and mitigate all attack vectors:

1. Design and Engineering: RTL Models and Software/Firmware Source Code.
2. Electronic Design Automation (EDA) Tools, Software compiler toolchains.
3. Manufacturing ("semiconductor fabs") – multiple facilities in process.
4. Sales channels & physical transport of components and finished products.
5. System integration steps and user provisioning mechanisms.
6. Malware, direct hacking attacks, software update mechanisms. *(.. etc ..)*

.. THANK YOU!