No-Go for Malware using Executable Watchdog

Cyber-Security based on new Paradigms

Based on / Summary of:

Erland Wittkotter, Roman V. Yampolskiy:

"No-Go for Malware using Independent Executable Watchdog"

https://asi-safety-lab.com/DL/No-Go-Malware_EW_22_03_13.pdf

• • •

Mar 14th, 2022

Erland Wittkotter, PhD
Erland.Wittkotter@gmail.com
+1 702 997 2475
Skype: ike2345

Highlights

- Goals: Proactive Cyber-Security
  - Why not eradicating Malware? **(Rootkits were eradicated via Hooksafe)**
  - Vision: fully automated, rapidly responsive Cyber-Security
  - Redundancy: watchdogs to diminish residual damage for all types of Malware
- Replacing CPU/OS for static/rigid security related operations
  - CPU for regular (dynamic/versatile) tasks
  - Software: Micro-Hypervisor sufficient against current cyber-crime
  - Hardware protection within devices or databus
- **Binary whitelisting** of trusted apps with updateable, stored **hashcodes**
  - Hardware as "Circuit Breaker" against omnipresent software vulnerabilities
  - Taking edge of from malware attacks (later: mitigation of Ransomware/Spyware)
- Software Vendors/Developers collaborating/volunteering allies
- Safe/Secure by Default via Executable Watchdogs (EWD)

Malware - Adversary

- **Malware:** Software for
  - Stealing/manipulating data
  - Damaging device's intended operation
  - Utilizing resources without permission
- **Cyber-Criminals – Nation-States**
  - AI used to lower cost of custom cyber attacks
  - Currently $1T (2020) damage, est. 2025: $10T
  - What if malware attacks keys used in SSL/TLS?
    - Damage to eCommerce, by Cyber-Warfare, …
- **Artificial (Super-) Intelligence (ASI)**
  - Feasible, but hypothetical: once emerged – then too late
  - ASI potentially: Super-Hacker, Digital-Ghost, Master-Thief, Super-***
    - Usage of Reverse Code Engineering at scale
    - Effortless access to all devices
    - Coordinated, diverse swarm attacks on large scale
    - Steals encryption keys, (+ computational, storage resources)

  → ASI potentially "worst adversary imaginable"

Need to Change Paradigms

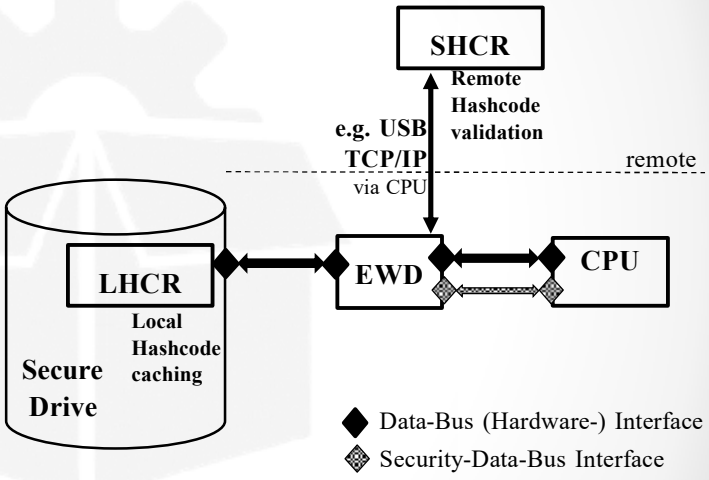
- **Old Ideas**
  - Trusting CPU (no other choice)
    - Sys-Admin rights implies users are responsible, aware, …
    - Requires removal every vulnerability
  - Don't trust developers (who are the good - or - bad guys?)
    - Everyone can provide software – scripts source of fileless malware attacks
- **New/Alternative Concepts**
  - CPU can not be trustworthy
    - CPU, OS, RAM not transparent, too complex
    - Type-1 Hypervisors good enough (for now) – better: combined with hardware
  - Software Vendors/Developers trustworthy – incentivize reputation
    - Legal (product) liabilities (→ Publishing exploit? → See you in court!)
    - Reputation essential
  - Creating (redundant) "Circuit" Breakers
    - Security Layer (preferably independent from main CPU – otherwise in Hypervisor)
    - Accepting software vulnerabilities; fixing them nonetheless (asap)

New Rules – Spelled-Out

1. **Detection Focus on "is known/harmless" not "is dangerous"**
   - Updateable binary app whitelisting – no blacklists with pattern/signatures
   - Hashcoding app files – compared to known/registered HC on server – cache locally
   - Only **known/harmless** software executable; **unknown/unregistered** is **dangerous**
2. **Separating security-related from regular (versatile) tasks**
   - Separation via Hypervisor or Hardware in device/databus
   - Security related code: rigid, static (simple)
3. **Software installation/update "outside" CPU**
   - Independence from OS - Overwriting OS activities **(distrusting OS/other apps)**
   - Preferably via independent hardware component in device/databus
4. **Trusting legitimate software vendors/developers:**
   - Don't fight them – consider them as partner
   - Vendors/developers are seeking reputation
5. **Scripts/Macros stored/checked before (covert) usage**
   - Fileless threats turned automatically into file-based scripts
     - Implemented by software vendors **(reputation preserving support)**

Executable Watchdog (EWD) Simplified

- Server-sided Hashcode (HC) Repository (**SHCR**)
  - Remote HC validation
  - Additional Security Data
- Local HC Repo (**LHCR**)
  - Locally cache HC (binary whitelist)
- **EWD** located in
  - CPU (Hypervisor) or
  - drives or
  - data bus: cable/connector



EWD Operation Modes
Activities

1. **Initial validation** of all executable files (LHCR, creating whitelist)
2. **Managing access** to executable files
   - DMA hashcodes all executables (i.e. "protected DMA")
   - Only known/harmless apps allowed in RAM
3. Regular **check of change in additional data**
   - Software's trustworthiness or
   - Security data changes
4. Regular **check/installation of updates**
   - Auto-updating all installed apps
   - Regular updates of binary whitelist of accepted apps
5. **Installation** of new software **(without support of CPU)**

EWD Versions
Examples

- **Basic Version**
  - 5 Operation Modes
  - EWD "transparent" for CPU/OS operations
- **Extended Basic Version**
  - Incl. software OS extensions **(against fileless malware)**
  - Validation: no script/macro taken from RAM covertly
- **Enhanced Version**
  - Incl. software OS extensions **(against new malware threats)**
  - Incl. software/apps specially compiled
    - Providing: Additional Information on Executables
  - Detection of unknown software/usage scenarios

Software Vendor/Developer Participation

- Expected to register hashcodes for all executables
  - Links to new updates
  - Starting with statistical inferences without vendor participation
- Providing basic information: Trustworthiness Categories: 1.-12.
  - Auto-assigned to basic categories: (1.-5.)
  - With evidence → warning (6.-9.), alert (10., 11.), ASI (12.) -- see Appendix
- Providing additional information on hardware/resource usage by software
  - confidential sharing of use of 3rd party components
- Software Vendors need public incentive to provide software fixes
  - Public acknowledgement → Reputation

→ **Software Vendors/Developers: Natural Allies to Cyber-Security**

Summary

- Executable Watchdog (EWD)
  - Creating unpassable hurdles for malware (binary whitelist of trusted apps) → Attack/Circuit Breaker
  - Different levels of protection (software, hardware) or detection
  - Against file-based/file-less malware
- Watchdog Technology extendable to fight
  - Ransomware
  - Spyware, Backdoors
  - ASI
- CPU not our trustworthy friend
- Software Vendors/Developers natural allies
  - Participating partners

Appendix

1) Trustworthiness Categories
2) Trustworthy Encryption/Decryption
3) Watchdog Technologies

Appendix:
(1) Trustworthiness Categories

1. Software (app, script, macro) - known/trusted sources
2. Software from sources with insufficient reputation
3. Scripts/macros used by a reputable distributor
4. App, script/macro used by less reputable distributors
5. Non-commercial, private apps/scripts/macros

Warning

6. Software with miss-use potential
7. Software that can modify other software (incl. scripts)
8. Software that could threaten users' data or privacy
9. Software with annoying features

Alert

10. Software with hidden features
11. Software with real damage potential

ASI

12. Assumed to be ASI software

Appendix:
(2) Trustworthy Encryption/Decryption

- ASI could steal every cleartext key – Unacceptable
- → Keys never shown in cleartext:
  - Cleartext keys in main CPU are compromised
  - No openly published public keys
  - Referencing keys via Hashcodes (otherwise similar to SSL/TLS/PKI)
    - Intentionally incompatible with existing PKI (and SSL/TLS)
- Keys in tamper-proof hardware: Key-Safes
  - Keys processed in dedicated Encryption/Decryption Units
  - Key-Exchange: hardware Key-Safe to hardware Key-Safe only
- Redundancy: Auto-detection of stolen keys
  - Data Exchange Protocol contains sequence numbers:
    - detect utilization of stolen keys reliably
  - Misuse detection (doesn't trust CPU)

→ **Unbreakable Data-Privacy with Redundancies**

Appendix:
(3) Watchdog Technologies

- Physically Task Separation in/via **Data-Bus**:
  - **Regular tasks** in main CPU (no change)
  - **Security-critical tasks** in separate "Watchdog" (WD)
    - Protected Storage (→anti-malware, →anti-ransomware)
    - Network (→anti-spyware, →anti-backdoor)
    - RAM/CPU (→ protected VM - ASE), GPU, etc. …
  - Only "well-known" code/OS allowed in WD – validated via known hashcodes
  - (Opt.) Data-Bus (cable, connectors, USB as backchannel) cheaply/easily retrofits
- Update of Software from trustworthy remote sources
  - Main CPU prevented from updating software (done by WD exclusively)
  - Separated/Remote Hashcode Validation
  - "Protected DMA": Local hashcode validation before loading into RAM
    - Reject data/code with unknown hashcodes (or known malware)

→ **Circuit-Breaker- / Hardware-based Cyber-Security**