The Evolution of Sandboxing and Anti-Sandboxing

2020-02-24

Carsten Willems, Ralf Hund
- Studied computer science at University of Mannheim
- Did my Ph.D. at Horst-Görtz-Institute in Bochum in 2013
  - Chair for systems security lead by Prof. Thorsten Holz
  - Researched various areas of applied system computer security
  - Strong focus on malware analysis
- Transformed academic findings into VMRay in 2013
  - CTO of the company ever since
- Presentation done together with Carsten Willems
  - Co-founder and CEO of VMRay
Introduction Into Malware Sandboxing
Why do we Need Sandboxing?

- **Malware** has been a huge issue for decades
- Used for fraud, espionage, sabotage, blackmail, etc.
- Attackers have become increasingly **professional**
- **Nation state actors** have been involved for a long time
- Ongoing **arms race** between attackers and defenders
- **Malware sandbox** is a proven concept to deal with **sophisticated** malware
- **First** malware sandbox (academic and commercial) are more than a decade old
- Lots of **technological progress** in this area since then
What Are Our Use Cases?

- Three different uses cases exist in the field of malware defense

  - **Analysis**
    - Manually analyze potential malware sample and **investigate** functionality
    - **Examples**: Disassembler

  - **Detection**
    - Automatically analyze malware in **high volume** and generate **alerts** in case of detections
    - **Examples**: Email and web detection solutions

  - **Protection**
    - Detect and block sample before being deployed to user
    - **Examples**: Anti-virus on end host, email and web protection solutions

  - Sandbox can be applied to **all** use cases
Two approaches to analyze and detect malware samples

**Static analysis**: Sample *is not* executed while analyzing
- Examples: Pattern matching (AV signatures, YARA), disassembler analysis (IDA Pro, Ghidra)

**Dynamic analysis**: Sample *is* executed while analyzing
- Examples: Debugger, malware sandboxing
Why do Both Approaches Exist?

- Static analysis arrived **first**
- While it has been very successful, attackers began to exploit its **weaknesses**
- Static analysis **cannot handle** detection of certain types of malware well
  - Ineffective against **polymorphic** malware
  - Ineffective against **new** malware
  - Ineffective against **targeted** malware
- Bottom line: Many solutions only effective at detecting **known** malware
- **Malware sandboxing** addresses this problem
So what is a Sandbox?

**MALWARE SANDBOX**

- **Detonate**
- **Monitor Behavior**
- **Determine if Malicious**

- File types: docx, exe, pdf, xlsx
Upload sample
Choose target environments
VTI SCORE: 100/100

Monitored behavior

Severity score
What is The Most Important Component of a Sandbox?

Detonation Environment

- User Log
- PCAP & Network Activity
- Behavior Log
- Behavior Monitor
- User Monitor & Simulator
- Network Monitor & Simulator
- Files System Monitor
- Memory Monitor
- UI Monitor
- Videos & Screenshots
- Dropped & Modified Files
- Process & Memory Dumps

Reputation Lookups

Sandbox Analysis

Static Analysis

Analysis Report

Pattern Matcher

Total Risk Score

malicious pattern database
What Monitoring Technologies Exist?

• **Three** monitoring technologies exist

• **Hooking**
  ▪ **Overwrite** instructions or code pointers in memory to detours sample execution
  ▪ Comparatively **easy** to implement, open source libraries exist

• **Full System Emulation (FSE)**
  ▪ **Emulate** entire detonation environment including **CPU**
  ▪ Can be for example based on QEMU or BOCHS

• **Hypervisor Monitoring (HVM)**
  ▪ Hardware assisted monitor located in **hypervisor/VMM** outside of target environment
  ▪ Can be for example based on x86 or ARM virtualization extensions (Intel VT, AMD-V, ...)

• There are **hybrids**
What Are The Essential Properties of The Behavior Monitor?

- There are **two** essential properties of a behavior monitor
- They have great **impact** on **detection** and **report quality**
- **Granularity**
  - What is actually monitored?
- **Evasion resistance**
  - How good can the sandbox **counter evasion** attempts?
- These two properties are the major **drivers** in the **evolution** of sandboxing
- **Performance** is also important but we will exclude it in this presentation
Granularity
What is Monitored?

- **Three** granularity levels exist
- **CPU** level
  - Monitor every **CPU instruction**
  - Done by **FSE**
- **Downstream API** level
  - Monitor **some** APIs in various levels
  - Done by **Hooking**
- **Upstream API** level
  - Monitor **all API functions** directly called by sample
  - Done by **HVM**
What Has Experience Told us?

- CPU level monitoring has heavy performance **overhead**
  - Questionable if advantages outweigh disadvantages
  - To analyze internal sample behavior, **static** analysis might be the better choice

- Downstream API monitoring can be **sketchy** and **noisy**
  - Important behavior is **missed** because it does not result in syscall
  - **Noise** is induced by libraries not related to sample

- Upstream API monitoring is the **golden mean**
  - Sample must call API to do something meaningful
  - Monitor catches **every** API call as **close** to the sample as possible
Evasion Resistance
What is Sandbox Evasion?

- Sandboxing has enjoyed widespread support
- Attackers have adopted their techniques to evade sandbox
- **Goal**: Do not reveal malicious behavior in sandbox, but only in actual target
- Sandboxing evasion attempt exist that can be put in three categories
  - **Monitor detection**: Detect presence of sandbox monitor
  - **Jamming**: Flood monitoring engine with useless behavior
  - **Context awareness**: Tie malicious behavior to certain triggers
What Are Examples For Monitoring Detection?

• Detect if running inside a **virtual machine**
  - Early and mostly **outdated** approach. Target can be virtualized too
  - Applies to Hooking, FSE and HVM. FSE and HVM can **conceal presence** of VM

• **Directly** detect monitoring engine
  - Check if certain **DLLs or kernel drivers** are loaded
  - Check if code or pointers have been **overwritten** to install hooks
  - Check for other artifacts (kernel driver in certain directory, etc.)
  - **Effective** against Hooking. **Futile** against FSE and HVM

• **Indirectly** detect monitoring engine
  - Use **performance** measurements
  - Can work against **any** technology, but very prone to **FPs**
What are Examples for Jamming?

- Target **common** properties of all sandboxes
  - Monitoring engine must **log** behavior
  - Monitoring engine naturally induces **overhead**

- **Call loop** with millions of iterations during sample initialization
  - Produces tons of **unnecessary** behavior when loop behavior is logged
  - Sample may run drastically **slower** in sandbox than on target
  - Can work against **any** technology. Monitoring can be **switched off** temporarily
What Are Examples For Context-Awareness?

• Install application **startup** scripts
  - Malicious behavior only revealed when Word is executed

• Only reveal malicious behavior after **reboot**
  - Many sandboxes cannot handle reboots

• Many **similar** approaches exist

• Can work against **any** sandboxing technology
Thanks for listening