(Un) Fucking Forensics
Active/Passive memory hacking/debugging

K2 / Director @IOACTIVE

https://github.com/K2
About me?

• Hacker for a while

  • inVtero.net
    • Memory analysis framework for Windows
    • Super fast/GBPS throughput
    • Memory integrity checking of VM’s/CrashDumps/Memory
    • Type aware memory hacking tool

• EhTrace
  • Binary trace tool
    • Uses hook/patch-less technique for in-process debugging

• Lots of other stuff
Outline / areas

• How to forensic, how to Fuck forensics and how to un fuck it.

• Intx80 AF technique on header wipe/non-resident code/trim()
  • How to deal with that

• RoP background, how it’s used in attacks
  • Gargoyle attacks & how to protect against them

• CloudLeech – twist on UlfFrisk DMA attacks / PCILeech
  • Demo of open source memory integrity platform for Windows!
Can you even forensic?

• In general: **Determine what happened.** Make a **timeline** of known events.

• “**Artifacts**” disk & memory (often incomplete/fragmented) used to build timeline.

• How good can we do? How do we know if were done?
 Artifact sources

How attestable are they?

- Time stamps from all sources to derive **timeline** (event logs/syslog/firewall/filesystem time, etc...)

- **Wevtutil** - Windows Events Command Line Utility. Configure more than 1189 event log sources

- SysMon (from **SysInternals**/Mark Russinovich) / neat config: https://github.com/SwiftOnSecurity/sysmon-config

- Linux (osquery https://github.com/facebook/osquery )
Handling memory

- Forensics meets Reverse Engineering
  - Dump / disassemble determine what the extent of capability the attacker possesses
  - I want to at least clear this guy out & find out how much damage he did

- Volatility/Rekall python forensic engines

- Stephen Ridley’s RE memory hacking tool: https://github.com/s7ephen/SandKit
  - Paper: Escaping the sandbox

- GAME HACKING! 😊
  - Let’s look at what people do to cheat sometime?
How to F’it?

• Hide really well
  • Wipe/Destroy logging/leave no trace/Stenography/Encrypt

• Misdirect
  • Flood/Annoy/Make analysis so costly$$$/Obfuscate/Spoof/

• Direct Attack
  • DefCon 15: Breaking Forensics Software: Weaknesses in Critical Evidence Collection Chris Palmer, Alex Stamos
Anti-forensics: Furthering digital forensic science through a new extended, granular taxonomy:

[Diagram showing a taxonomy of anti-forensics techniques, including categories such as Steganography, Encryption, Data Hiding, and Attacks against forensic tools and methods.]
Foreshadowing: normalize you’re operations

• A great way to operate undetected is to ensure you are not an anomaly.
  • Use the resources of you’re target to conduct you’re operations.

• “Configuration” attacks
  • Enable IPV6 tunneling & VPN access
    • Attacker has trusted CA capability (added their privkey to trusted list)

• The more “normal” the method will be very hard to dig up
Int0x80’s AF counter

Attack against a tool: Rekall
- Prevent dumping for working

More ways to get it:
- Use VAD (kernel source)
- Use PageTable (ABI)
- Use inVtero.net
  - dump.py - VADDump (VAD) or Dump (PageTable)
RoP: More normal

• RoP is an often discussed topic used mostly for exploitation
  • RoP uses the CPU stack semantics to execute as if it were a really large set of return statements.
    • This uses the code that’s already on the system more “normalized” than if you had to inject an executable payload that did not originate from the target

• RoP is used by Gargoyle (Josh Lospinoso) as an example of a persistence technique that evades memory analysis systems

• There is hope, we can detect RoP attacks through call chain evaluation
RoP is not perfect

Gargoyle persistence

• Leverages a timer and blocking wait that moves it into the “active state”

• Once active, stages page protection +X

• Then uses this page to invoke it’s primary payload

• It then mask’s the +X bit back off and goes inactive
Tools too defend against RoP attack?

• Analysis: **ROPEMU**: A Framework for the Analysis of Complex Code-Reuse Attacks
  • Dump a complex RoP execution trace into an ELF!! Wow!

• Detection: inVtereo.net can perform a stack checking function against the memory dump.
  • Similar tools for monitoring RoP at run time (EhTrace, RoPGuard, etc...)
  • From the inVtereo output, you really do NOT want to see the Gargoyle gadget, or anything that looks like a stack pivot
Injection techniques

• Many variegations to achieve the same goal;
  • 10 Process Injection Techniques (Ashkan Hosseini/Endgame)

• LoadLibrary, Hallowing, Thread hijacking, Windows Hooks, Registry keys, APC, SetWindowLong, Shims & IAT shims

• Flame was a sort of hallowing attack
  • “hid” inside of ntdll, remained undetected for years
Enter DMA with PCILeech

- Ulf Frisk Direct-Memory-Attack-the-Kernel:
  - PCILeech attacks and utility for forensics (memory) collection

- Uses a variety of (very cool) techniques to execute payloads

- One of the simplest is the “unlock” functionality
  - It’s an inline patch however
  - Hard to detect w/o manual reversing
Integrity validation

• Full validation at any point in time must be able to be conducted
  • System state should/must be static
    • CPU execution will allow attackers to play games / evade read’s

• RDMA

• LiveMigration / Snapshotting
Tie it together

Un F’d Memory Forensics – Remove the guesswork

• Leverage wide range of information sources

• Have a comprehensive global view of the data set

• Appropriate countermeasures for most attackers (RoP)

• Integrity checking of memory (in line patch protection)

• Symbols and context for analysis of pointers
  • Pointer tracking becomes more significant as we can qualify their address/vector
Demo’s & Thank you

• Check out the tools

Github.com/K2