Windows HIPS evaluation with Slipfest

Julien Tinnes, Yoann Guillot

France Telecom R&D

What is a HIPS?

- Host intrusion prevention system
  - Tries to generically prevent the exploitation of security flaws
  - Tries to mitigate their consequences
- State of the art on Linux with PaX (and GrSecurity)
  - Emulates non-executable pages semantic using splitted TLBs or segmentation (when not supported in hardware)
  - Prevents the injection of arbitrary code in the address space using this semantic
  - Prevents reusing of existing code by an exploit with address space layout randomization (ASLR)
  - Restricts the privileges of a process with mandatory access control (GrSecurity)

What is Slipfest (SF)?

- A tool developped by France Telecom R&D to help the evaluation of Windows HIPS products
- The name is basically a french joke meaning "Panty's party"
- Officially it's an acronym for "System-Level Intrusion Prevention Framework Evaluation Suite and Toolkit"
  - Who would believe that? :)
- It can be used to:
  - understand how your HIPS works
  - see its limitations

Windows HIPS

- There are a lot of products
- Some big companies bought a smaller one for their HIPS product
  - Cisco CSA, McAfee Entercept, Symantec client security . . .
- Wehntrust, Ozone, Eeye Blink, Ossurance, Prevx, Geswall, Buffer-Shield, NGSEC's StackDefender . . .
- Windows XP SP2 and Vista include HIPS-like features
- Features
  - Prevents shellcode execution (NOEXEC)
  - Address space layout randomization (ASLR)
  - Mandatory access control (MAC)

NOEXEC, technique #1, NX emulation

- Some products are doing PaX-like NX emulation using pentium's splitted TLBs
  - They are rare and quite hackish
  - Significant impact on performances (most people use segmentation instead in PaX)
  - SecureStack, BufferShield, StackDefender
  - If supported by the processor, XP SP2 uses NX, but not at its full potential
- Slipfest can test whether a non-executable semantic is present

NOEXEC, technique #2, behavioral analysis

- Instead of relying on non-executability:
  - Most products will let you run your shellcode
  - But they'll try to catch you when you call an API
  - Some vendors call this "behavioral analysis"
- You can get arbitrary code execution
  - They use SDT or userland (in libraries) hooks
  - When a hook catches you, the HIPS runs a heuristic
    - If something seems wrong (e.g. you're returning to the stack) it kills your process
    - If everything seems fine, it lets you run

It’s a bad solution

Behavioral analysis is by essence very weak

- When you've got arbitrary code execution all bets are off
  - You can do whatever the process can do
  - The only real remaining line of defense is privilege limitation (e.g. with MAC)
- You can bypass kernel and library based BA:
  - Fool the heuristic: make it think you're the legitimate program
- Additional techniques for library-based BA:
  - CPL-3 code is not more privileged than you are, you can emulate the hooked library
  - You can "jump above" the hook or even unhook it (depends on MAC)

SLIPFEST against behavioral analysis

- You can use Slipfest to find out what is hooked
  - List hooks in SDT, and see which module they are pointing to
  - List hooks in Libraries, and see where they're pointing to (SF resolves calls and jumps)
  - You can unhook libraries
    - This allows to easily find out if the interesting hook is in the SDT or in a library
- You can use Slipfest to find out how smart the heuristic is
  - Inject different shellcodes in the process
  - The shellcodes will try to fool the heuristic by proxying through existing code

HIPS Internals, ASLR

- Some products implement some address space layout randomization
  - ASLR makes it harder for your exploit to rely on fixed addresses
  - Wehnus Wehntrust, Ozone, Eeye Blink . . .
  - Windows XP SP2 and Vista add some randomization (TEB, stack, heap)
- Slipfest can tell you what is randomized and how much it is randomized
  - Creates a bunch of processes reporting back through a pipe

HIPS Internals, mandatory access control

- Mitigates the consequences of an exploitation by making a process less privileged
- Example: shareware.exe cannot modify a file in WINDIR
- You need to write a policy
  - It's hard
  - Most of the time there is no automatic consistency checking mecanism
  - For instance if you forbid a process to write to WINDIR you must also forbid it to access the NtWriteProcessMemory service
    - Otherwise it could inject a shellcode in another process which would write to WINDIR

HIPS Internals, mandatory access control (2)

- Most products (if not all) have some inconsistancies that you cannot fix, especially if the vulnerable process runs with administrator privileges
  - Slipfest can test some of them: loading a driver the "other" - undocumented - way, writing to physical memory device . . .
- Access control is implemented using hooks too
  - Kernel hooks (the correct way)
  - Userland hooks (the bad way)
    - They can be bypassed (it's CPL-3 code)
  - Slipfest can help you understand how it is implemented (see BA)

Why userland hooks are bad

Demo

- ASLR detection
- List SDT and userland hooks
  - See where they are pointing to
- Shellcode generation and injection
  - Hash based find_proc with forwarders support
- Bypass the heuristic by proxying API return through existing code
  - e.g. we return to an address in the PE just after a call and gain control back with the next 'ret'

Conclusion

- Future
  - Make the GUI actually usable (help appreciated)
  - Document the hidden features so that you don't have to read the source (almost 6000 lines) to find them
  - Port to Metasploit stagers some of the shellcodes and bypass techniques (WIP)
- Thanks for attending!
- Any questions ?
- Available in a few minutes at http://slipfest.cr0.org with a GPL licence