ABOUT ME

• Senior Researcher, working on prevalent threats
• Joined F-Secure in 2004
• Background in R&D (engine dev) and malware analysis
• Reflash background: what’s the CVE?
• Flash platform: rich graphics for web, games, videos, music
• Been around since 90’s (FutureSplash Animator -95, sold to MacroMedia)
• Current platform: ActionScript 3, ActionScript Virtual Machine 2 (AVM2)
• Flash Player 9 (AS3, AVM2) released June 2006
• Plagued by bugs, several major zero days 2010-2016
• Flash still around, mainly because of money in the ads
PROBLEMS WITH FLASH RESEARCH

- Lack of flash debugging tools
- Obfuscations, protection systems
- Inline loading of embedded flash content (usually several layers)
REFLASH?

- Framework for assembly-level flash analysis
- Record-and-replay debugging
- Works by instrumenting flash files “on the wire”

Exploit Kit

HTTP(S)

Reflash proxy

HTTP(S)

Stack frames, inline SWF

Injected bytecode

Record analysis
AVM2

- Runs ActionScript3
- Open sourced (Tamarin / Mozilla)
- Stack VM
- Runs Just-in-Time compiled bytecode (JIT major problem)
- No official tools for assembly-level debugging
Closure call (opcode 0x41)

• stack: function, receiver, args …
• function: closure being called
• receiver: object for “this” value
• args: arguments passed for the function
• After function call: stack is cleared, and result is pushed onto the stack
• Example: flash.display.Loader::loadBytes
INSTRUMENTATION

• Works by injecting code to the flash opcode stream
• Three types of code injection:
  • Method entry hook
  • Generic opcode hook for collecting stack frames
  • Generic post-opcode hook
• Flash files are disassembled, and assembled back after instrumentation
• Analogous to Intel x86 inline hooking (a'la Detours static PE modification)
• No signatures or other integrity measures in flash! (like java)
INSTRUMENTATION

- Injected opcodes change branch targets
- Targets are *absolute* within the method frame
  - If target is before injection, it remains the same
  - If target is after injection, it needs to be adjusted

![](image)
PROXY

- Driver for instrumentation, result gathering, database creation
- Implementation: python mitmproxy
- Track incoming flash files real-time
- Flash files coming from two directions: from the network and from the loadBytes hooks
GENERIC STACK FRAME HOOK

- Arbitrary opcode can be hooked
- Hooks pack stack frames prior to opcode execution
- Also the type of stack frame data (as known by AVM2) is packed
- Data is sent to proxy with asynchronous TCP connection
LOADBYTES HOOK

• Application of generic stack frame hook
• All *call* hooks are examined further:
  • Called object: `flash.display::Loader`?
  • First argument: `ByteArray` with flash file?
• If inline flash file is detected, it is sent back to proxy *synchronously*
• Stack argument replaced by received `(re)flash(ed)` file
• Stealthy instrumentation of inline flash files
POST-OPCODE HOOK

• Opcodes can also be hooked for manipulating resulting stack values

• Reflash implements one specific post-opcode hook, `getproperty`:
  • Modify flash.system::Capabilities: flash version, example: “WIN 15,0,0,203”
  • Modify flash.system::Capabilities: OS version, example: “Windows XP”

• Purpose of `getproperty` post-hook is to add flexibility for Exploit Kit testing
RECORD-AND-REPLAY

- Generic hooks send stack frames back to the proxy
- Stack frames are packed with AMF (ActionScript Message Format)
- Proxy records stack frames in SQL database
- Static stack frame meta-data also written to the database (disassemblies)
- With the database, *coherent view of execution* can be formed afterwards
CVE DETECTION: HYBRID APPROACH

- Usually exploits abuse function arguments
- Reflash features running YARA against stack frame data
- Fully generic solution is not possible (what’s the CVE?)
- Static CVE signatures cannot be used without heavy dynamic processing (protection systems, inline flash files...)

F-Secure
SO WHAT’S THE CVE?
RELATED WORK

- FlashHacker (Jeong Wook Oh)
- JPEXS debugging support
- Timo Hirvonen’s Sulo