Dynamic Hooks: Hiding Control Flow Changes within Non-Control Data

Sebastian Vogl, Robert Gawlik, Behrad Garmany, Thomas Kittel, Jonas Pfoh, Claudia Echkert, Thorsten Holz

Motivation

- In general, malware needs to intercept events within the system
- Event interception requires us to divert the control flow at runtime
- This is accomplished by installing hooks into the control flow

Motivation

- Types
  - Change code(Code Hooks)
  - Change function pointer(Data Hooks)
- Researchers have presented effective detection mechanisms for both types
- How can we evade existing detection mechanisms?

Definition

- Control data
  - Target location of a branch instruction
- Non-control data
  - Never contains the target address for a control transfer
- Transient
  - We consider control data to be transient when it cannot be reached through a pointer-chain originating from a global variable
- Persistent
  - We consider control data that is reachable through a global variable as persistent

Attacker model

- We already control the target application
  - We are not affected by most protection mechanisms
  - We can modify internal data structures and attack internal functions
  - We can prepare our shellcode in advance

High-Level Overview

- Fundamental idea
  - Hide desired control flow change within non-control data such that there is no clear connection between the changes that the malware conducts and the actual control flow change

Example

- [next+8] = prev
- [prev] = next
- We focus on 8-byte writes
- Mov [rax], rbx

```
1  struct list_head {
2          struct list_head *next;
3          struct list_head *prev;
4  };
5
6  static void list_del(struct list_head *entry)
7  {
8           entry->next->prev = entry->prev;
9           entry->prev->next = entry->next;
10 }
```

Program Slicing

- Mov [<destination>], <source>
- Backwards breadth-first search on the assembly-level
- Extract path if destination and source originate from a global variable
- Implementation: IDA Pro

Symbolic Execution

- Transform extracted path into VEX IR(pyvex)
- Map VEX statements into Z3 expressions
- Check satisfiability of conditional branches
- Generate detailed information about controlled registers

Experiments

- Implemented three prototypes of dynamic hooks
- Control Hook: Interception of system calls(Linux)
- Data Hook: Backdoor(Linux)
- Control Hook: Interception of process termination(Windows)

OS	*Size*	*Instructions*	*8-byte moves*	*Slices*	*Paths*
Linux 3.8 64-bit (`vmlinux`)	18.8 MB	1,976,441	42,130 (2.1%)	1753 (4%)	**566 (32%)**
Windows 7 SP1 64-bit (`ntoskrnl.exe`)	5.3 MB	1,330,791	26,694 (2.0%)	5450 (20%)	**379 (07%)**