Ripples in the Gene Pool

Creating Genetic Mutations to Survive the Vulnerability Window

Chris Eagle

Defcon 14

Standard Disclaimer

The views expressed in this presentation are my own and do not necessarily reflect the official policy or position of my employer

Background

- Ideas born from CTF
  - Often no source code available
  - Stopping service not an option
  - Patch needs to survive < 3 days
- Techniques discussed using x86 examples
  - Easily adapted to other platforms

The Software Monoculture

- From Geer et al.
  - A monoculture of networked computers is a convenient and susceptible reservoir of platforms from which to launch attacks; these attacks can and do cascade.
  - This susceptibility cannot be mitigated without addressing the issue of that monoculture.

The Vulnerability Window

- Time of discovery to patch availability
- Two components
  - Discovery to disclosure
    - Hardest to pin down
    - Requires a cooperative discoverer
    - Effectively no defensive capability
  - Disclosure to patch availability
    - Defense via mitigation
    - Must be shorter than disclosure to automated exploit window or all hell breaks loose

Third Party Patching

- Discoverer provided patch
  - Rarely seen
- Independent researcher provided patch
  - Follows disclosure, precedes vendor patch
  - Also rare
    - Ilfak's WMF hotfix
    - eEye's IE patch
  - Controversial

Responsible Disclosure ☺

- I don’t care if you disclose or not
- I don’t care if you coordinate with a vendor or not
- IF you do choose to disclose please do all the grandmothers in the world a favor and publish ways to mitigate

What Mutations Are

- Simple changes to a binary to alter runtime characteristics sufficiently enough to foil automated exploitation attempts
  - Often easier than a proper fix
- Security Through Obscurity

What Mutations Aren’t

- Not un-exploitable
- Not a long term solution

Assumptions

- Automated exploits are generally built for specific target layouts
- Automated attackers simply move on to new targets when they do not achieve expected results

Binary Patching

- A bit of a black art
- Proper fixes generally require additional space
  - Compilers are usually concerned with size and don’t generally leave to much free space
  - May require extensive editing of file headers
- May require functions not originally imported

Simple Mutations

- Stack Mutations
  - Alter stack layout to something unexpected
  - Simplest to perform
- Heap mutations
  - Alter heap layout
- Format String Mutations
  - Add extra parameter
- Uninitialized Stack Variables
  - Alter stack layout to move variable

Stack Mutations

- Grab more stack space
- Typical function prologues

```
push    ebp
mov     ebp, esp
sub     esp, 34h      ; one byte constant

push    ebp
mov     ebp, esp
sub     esp, 414h     ; four byte constant
```

Stack Mutations (cont)

- After grabbing more stack space frame pointer offsets may need adjusting
  - esp based frames
    - No adjustment required for local variable offsets
    - Adjust all function argument offsets
  - ebp based frames
    - Adjust all local variable offsets
    - No adjustment needed for function arguments

Stack Mutation Example

```
-00000410 var_410           dd ?
-0000040C var_40C           dd ?
-00000408 var_408           dd ?
-00000404 var_404           dd ?
-00000400 var_400           dd ?
-000003FC var_3FC           dd ?
-000003F8 var_3F8           db 1016 dup(?)
+00000000  s                db 4 dup(?)
+00000004  r                db 4 dup(?)
+00000008 arg_0             dd ?
```

```
push    ebp
mov     ebp, esp
sub     esp, 414h       ; claim extra 1024
lea     edx, [ebp+var_3F8]
```

Stack Mutation Example (cont)

```
-00000810 var_810    dd ?    ; former var_410
-0000080C var_80C    dd ?    ; former var_40C
-00000808 var_808    dd ?    ; former var_408
-00000804 var_804    dd ?    ; former var_404
-00000800 var_800    dd ?    ; former var_400
-000007FC var_7FC    dd ?    ; former var_3FC
-000007F8 var_7F8    db 1016 dup(?)  ; former var_3F8
    ; 1024 bytes of padding here
+00000000  s                db 4 dup(?)
+00000004  r                db 4 dup(?)
+00000008 arg_0             dd ?

 push    ebp
 mov     ebp, esp
 sub     esp, 814h         ; NOTE CHANGE HERE
 lea     edx, [ebp+var_7F8]  ; AND HERE
```

Stack Mutation Example (cont)

```
-00000810 var_810    db 1016 dup(?)  ; former var_3F8
-00000410 var_410    dd ?   ; In this case no other
-0000040C var_40C    dd ?   ; variable offsets need
-00000408 var_408    dd ?   ; to be changed
-00000404 var_404    dd ?
-00000400 var_400    dd ?
-000003FC var_3FC    dd ?
    ; 1016 bytes of padding here
+00000000  s          db 4 dup(?)
+00000004  r          db 4 dup(?)
+00000008 arg_0       dd ?

 push    ebp
 mov     ebp, esp
 sub     esp, 814h              ; NOTE CHANGE HERE
 lea     edx, [ebp+var_810]     ; AND HERE
```

Stack Mutations (cont)

- Variations
  - Add padding to all functions, especially main
    - The effect is poor man's stack randomization
  - Reorder local variables
    - Place additional locals between buffers and saved return address
    - Poor man's canaries

Heap Mutations

- Allocations made using
  - Fixed size chunks for known size structs/arrays
  - Computed size chunks based on expected size of structs or array
- Mutation is made to increase requested size

Heap Mutation Example

- Simple static size mutation
  - Trades increased memory use for improved(?) security

```
push    16
call    _malloc
```

  - becomes

```
push    64
call    _malloc
```

Heap Mutation Example

- Computed size mutations
  - More difficult
  - Need to create space to adjust computed size upward
    - Need a gap of 5 or more bytes to insert an add instruction

Format String Mutations

- This is a more standard patch
- Usually need to push a valid format string
- Create space for extra push
  - At least 5 bytes required
- Create format string in binary
  - Overwrite some unimportant string like usage
- Modify post return stack adjustment

Unintialized Stack Variable Mutations

- Two options here
  - Create space to add initialization code
  - Adjust stack offsets to move variable to a less predictable location

Demonstrations

Questions?

Chris Eagle
cseagle@gmail.com

shoutz & greetz to the Sk3wl kr3w, Kenshoto, and all the Shmoo

References

- Geer, et al., “CYBER/*IN*SECURITY: THE COST OF MONOPOLY”, Sep 2003

  http://www.ccianet.org/papers/cyberinsecurity.pdf
- Arbaugh, et al., “Windows of Vulnerability: A Case Study Analysis”, IEEE Computer, 2000
  - http://www.cs.umd.edu/~waa/pubs/Windows_of_Vulnerability.pdf
- infectionvectors.com, “Just In Time: Microsoft’s Time to Exploit”
  - http://www.infectionvectors.com/vectorspaces.htm#jit