Reverse Engineering and the ANI Vulnerability

Alexander Sotirov
alex@sotirov.net
Introduction

• Security researcher at Determina
• Vulnerability analysis and reverse engineering Microsoft patches
• Exploit development experience
• Speaker at CanSecWest, REcon, SyScan and BlackHat
• Vista vulnerabilities
Exploit Demo
Part I

Reverse Engineering
Microsoft Patches
Patch Statistics

- More than 500 bulletins since 1998
- Most updates fix multiple vulnerabilities
  - 5 vulnerabilities in the latest IE patch
- Fixed release schedule
  - second Tuesday of the month
Skeletons in Microsoft's Closet

• Security issues are often fixed silently
  ○ security researcher reports a vulnerability
  ○ Microsoft audits the affected code and discovers 5 related bugs
  ○ 6 bugs are fixed in the patch
  ○ security bulletin describes only the first bug

• Service packs silently fix bugs
Withholding information

• Security bulletins omit technical details:

  There is a privilege elevation vulnerability in Windows 2000 caused by improper validation of system inputs. This vulnerability could allow a logged on user to take complete control of the system.

• Reverse engineering is the only way to really understand vulnerabilities
Patch Analysis

• The security industry relies on reverse engineering patches for:
  ○ attack vectors and packet signatures
  ○ vulnerability analysis
  ○ remote detection of the vulnerability
  ○ exploit development
Reverse Engineering Tools

• IDA Pro
  ○ great plugin API

• BinDiff
  ○ function level diffing of binaries

• PaiMei
  ○ allows tracing and visualization of execution paths, guides static analysis

• VMware
  ○ backwards debugging with multiple snapshots
Patch Analysis Demo
Part II

Exploitation
Protection Mechanisms in Vista

• /GS stack cookies
• Address Space Layout Randomization
• Data Execution Prevention
static_cookie = rand();
void foo(char* input)
{
    int cookie = random_cookie;
    char buf[256];
    strcpy(buf, input);
    if (cookie != random_cookie)
        abort();
}
Bypassing /GS

- No need to bypass /GS for ANI exploit
- There is no stack cookie in our function:
  - /GS protects only functions with arrays
  - ANI header data is read into a structure
ASLR

• Address Space Layout Randomization
  ○ stack and heap addresses
  ○ base addresses of executables and libraries

• Blocks the use of jmp esp trampolines
  ○ we need a fixed location
Bypassing ASLR

• Find something that's not randomized
  ○ executables
  ○ ntdll.dll and kernel32.dll

• Write our shellcode at a known location
  ○ vulnerability specific

• Heap spraying
  ○ great for browser exploits
Heap spraying

Used by most browser exploits since 2004

```javascript
var x = new Array();

// Fill 200MB of memory with copies of the
// NOP slide and shellcode

for (var i = 0; i < 200; i++) {
    x[i] = nop + shellcode;
}
```
Normal heap layout

- Used memory:
- Free memory:
After heap spraying

used memory:  
free memory:  
shellcode:  

Any address around 200MB is likely to contain shellcode.
Data Execution Prevention

- CPU support for non-executable data
  - x86 architecture did not support it
  - introduced by AMD and Intel in 2004
- Prevents code injection
- Opt-in on Windows
  - IE not protected by default even on Vista
Bypassing DEP

• Return-into-libc attacks
  
  system("/bin/sh")

• Disabling DEP
  
  ○ jump to code in ntdll.dll that disables DEP

• VirtualProtect
  
  ○ change the protection of the heap to allow execution
Bypassing DEP

- ASLR is supposed to stop DEP bypasses

- LoadAniIcon function has an exception handler that catches access violations

- Send multiple ANI files
  - guess the address of ntdll.dll (only 256 locations)
  - disable DEP and execute shellcode
Part III

Secure Development
Security from the ground up

• Use the right language and platform
  ○ Java and Python eliminate buffer overflows
  ○ PHP encourages insecure programming
  ○ C++ is a bad choice in almost any case
Designing secure software

- Isolate components along trust boundaries
  - authenticated / non-authenticated
  - root / non-privileged user
  - user data / trusted data

- Narrow, well defined interfaces

- Validate all data that crosses a trust boundary
Know when to give up

• Some things are just really bad ideas
  ○ ActiveX
  ○ Google Desktop Search web integration
  ○ PHP register_globals setting

• Adding security on top of an existing insecure system
  ○ Windows and Oracle legacy codebases
  ○ WordPress vs. MediaWiki
Exploit mitigation

- All software has bugs

- Assume that all software you write will ship with critical security vulnerabilities

- Make exploitation harder
  - /GS cookies and ASLR are great examples
  - SSH privilege separation
  - Avoid single sign-on for web services
<table>
<thead>
<tr>
<th></th>
<th>Vista</th>
<th>XP SP2</th>
<th>2000</th>
<th>RHEL</th>
<th>Open BSD</th>
<th>OSX</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ASLR</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Executable Randomization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Library Randomization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stack Randomization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heap Randomization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Stack Protection</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stack Cookies</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable Reordering</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-executable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Heap Protection</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heap Metadata Protection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-executable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Questions?

alex@sotirov.net