Security Engineering in Windows Vista

Black Hat 2006

John Lambert
Security Group Manager, SWI

Contact: johnla at microsoft dot com
Intro – Who Am I?

- Joined Microsoft 6 years ago
  - There for Code Red, Nimda, SQL Slammer, Blaster
  - There for Windows Security Push, Windows XPSP2, SDL
  - At the billg meeting where Trustworthy Computing was kicked off

- Security Group Manager, Secure Windows Initiative
  - Security Development Lifecycle (SDL) at Microsoft
  - Manage an internal team of hackers and security program managers
Agenda

• Here to explain what we’re doing in Windows Vista
  o Overview of security engineering activities
  o Some detail on our major security initiatives
  o Overview of our mitigations work

• Here to listen to:
  o Any engineering focused feedback you have
  o How you think we’re doing

• What you WON’T hear in this presentation:
  o Security features
    • What’s changed in Kerberos, PKI, BitLocker, etc
Security Deployment Lifecycle Tasks and Processes

- **Product Inception**
  - Assign resource
  - Security plan

- **Design**
  - Design guidelines applied
  - Security architecture
  - Security design review
  - Ship criteria agreed upon

- **Implementation**
  - Guidelines & Best Practices
    - Coding Standards
    - Testing based on threat models
    - Tool usage
  - Threat Modeling
    - Models created
    - Mitigations in design and functional specs
  - Security Docs & Tools
    - Customer deliverables for secure deployment

- **Verification**
  - Security Push
    - Security push training
    - Review threat models
    - Review code
    - Attack testing
    - Review against new threats
    - Meet signoff criteria
  - Final Security Review (FSR)
    - Review threat models
    - Penetration Testing
    - Archiving of Compliance Info

- **Release**
  - Security Response
    - Feedback loop
      - Tools/ Processes
      - Postmortems
      - SRLs

- **Response**
  - RTM & Deployment
  - Signoff
Windows Vista Security Approach

Stop playing catch up - find & fix before ship
  o Use root cause analysis to ensure we’re solid against previous issues
  o Look forward to get ahead of new classes of issues
  o Apply all the lessons from Windows XP SP2, Windows Server 2003 SP1 to a mainline release

1. Apply least privilege throughout architecture
   o Harden services, applications, browser

2. Automate proven techniques
   o Buffer overruns and common coding mistakes
   o RPC and File parser fuzzing
   o Banned API removal

3. Methodically apply security expertise on whole product
   o Attack Surface Reduction, Threat Model reviews
   o Feature reviews
   o Penetration testing

4. Defense-in-Depth Mitigations
   o Firewall on by default
   o Enhanced protections for stack, heap, and more
Component Team

- Training
- Threat Models
- Component level code review and testing

Security

- PREfast
- Banned API Removal
- SAL Annotations
- FxCop
- Privacy, Reliability, …

High Risk Code

- Design & Attack Surface Review
- In Depth Threat Model Review
- Pentration Testing
- Mini-Security Push (if necessary)
- Network and File Parser testing
- Special Cleanup Projects

Quality Gates

“Winmain” Main Source Tree

Across All Code

Security Bug Tracking
PREfix, Default Permissions
PART 1: APPLY LEAST PRIVILEGE THROUGHOUT ARCHITECTURE
Pervasive Least Privilege

• Problem: If a program runs as SYSTEM or Administrator, any compromise is catastrophic

• Approach:
  – Applications
    • Enhance standard user
    • Administrators use full privilege only for administrative tasks or applications
    • Run some applications with heightened restrictions
  – Service Hardening
    • Minimize privilege user pervasively in applications and services
    • Define restrictions to ensure behavior conforms to expected activity
A Look at Service Hardening

• Motivation:
  o Services are attractive targets for malware
    • Sasser, Blaster, Slammer, Zotob, CodeRed
  o No need for user interaction
  o Often run in elevated identities
  o Many worms:
    • Tinker with the OS (drop a root kit)
    • Open network connections to propagate
Reduce Privilege Level of Services

- Get services out of SYSTEM to LOCAL SERVICE or NETWORK SERVICE
- Reductions from Windows XP SP2
  - Eight (8) SYSTEM services now run as LOCAL SERVICE
    - Windows Audio
    - DHCP Client
    - Windows Event Log
    - COM+ Event System
    - Workstation Service
    - Windows Time
    - Security Center
    - Windows Image Acquisition
  - Four (4) SYSTEM services now run as NETWORK SERVICE
    - Cryptographic Services
    - Policy Agent
    - Telephony
    - Terminal Services
- And 48% of new services in Windows Vista run under a low privilege account
Compartmentalize Resources with Service SIDs

- **Per Service SIDs**
  - Derived from service name in SCM
  - S-1-5-80-xxxx

- **ACL objects such that only your service can manipulate them**

- **Integrated into:**
  - LookupAccountSid
  - LookupAccountName

Example:
S-1-5-80-242729624-280608522-2219052887-3187409060-2225943459
Resolves to NT SERVICE\CryptSvc
Eliminate Unnecessary Privileges

- Describe the privileges your service needs and all others are removed
- Process that host multiple services get union of required privileges

```c
// Set up the required privileges
SERVICE_REQUIRED_PRIVILEGES_INFORMATION servicePrivileges;
servicePrivileges.pmszRequiredPrivileges =
    (L"SeChangeNotifyPrivilege\0"
    L"SeCreateGlobalPrivilege\0"
    L"SeImpersonatePrivilege\0");

fRet = ChangeServiceConfig2(
    schService,
    SERVICE_CONFIG_REQUIRED_PRIVILEGES_INFO,
    &servicePrivileges);
```
Restricted Network Behavior

• Define the network requirements for your service…
  – You describe and OS enforces network access policy
  – Eg: foo.exe can only open port TCP/123 inbound
    • |Action=Allow|Dir=In|LPORT=123|Protocol=17|App=%SystemRoot%\foo.exe
  – If foo.exe has a bug, the rogue code cannot make outbound connections

• …enforced by the firewall
Group services to take advantage of restrictions

- **New svchosts**
  - **LocalServiceNoNetwork**
    - Runs under local service account. This group has no network access and has a write-restricted token
  - **LocalServiceRestricted**
    - Runs under local service account. This group uses a fixed set of network ports on the system and a write-restricted token
  - **LocalServiceNetworkRestricted**
    - Runs under local service account. This group uses a fixed set of network ports on the system but not a write-restricted token
  - **NetworkServiceRestricted**
    - Runs under network service account. This group uses a fixed set of network ports on the system and a write-restricted token
  - **NetworkServiceNetworkRestricted**
    - Runs under network service account. This group uses a fixed set of network ports on the system but not a write-restricted token
  - **LocalSystemNetworkRestricted**
    - Runs under local system account and accesses fixed set of network ports.
## Case Study: DHCP Client Service

<table>
<thead>
<tr>
<th></th>
<th>Windows XP SP2</th>
<th>Windows Vista</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Account</strong></td>
<td>SYSTEM</td>
<td>LOCAL SERVICE</td>
</tr>
<tr>
<td><strong>Privileges</strong></td>
<td>24</td>
<td>4</td>
</tr>
<tr>
<td><strong>Network Identity?</strong></td>
<td>Yes (Machine Account)</td>
<td>No</td>
</tr>
<tr>
<td><strong>Uses Fixed Set of Ports?</strong></td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Data accessible only by service? (Service SID)</strong></td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
PART 2: AUTOMATE PROVEN TECHNIQUES
A Brief Introduction to the Standard Annotation Language (SAL)

• Tools can only find “so much” without more contextual information
  o SAL helps bridge the gap by providing interface contract information to the tools
    • SAL leads to improved analysis (More bugs, Less noise)
  o The concept is not new: think IDL in RPC

• Prime focus is finding buffer overrun bugs
  o Given the code buff[x] = y;
  o How big is buff, and what is the value of x?
  o Problem is C/C++ don’t associate buffer to their sizes
SAL Example

```c
void FillString(
    TCHAR* buf,
    size_t cchBuf,
    char ch)
{
    for (size_t i = 0; i < cchBuf; i++)
    {
        buf[i] = ch;
    }
}
```

- If cchBuf doesn’t correspond to the actual size of buf, the loop will walk off the end of buf
Annotations – A Closer Look

```c
void FillString(
    __out_ecount(cchBuf)
    size_t cchBuf,
    char ch)

__out_ecount(cchBuf)
```

Out parameter (will be written to) and is non-null

Buffer size is an Element count

Buffer is cchBuf elements in size
Adding Annotations

```c
void FillString(
    __out_ecount(cchBuf) TCHAR* buf,
    size_t cchBuf,
    char ch)
{
    for (size_t i = 0; i < cchBuf; i++)   {
        buf[i] = ch;
    }
}
``` 

```c
void main() {
    TCHAR *buff = malloc(200 * sizeof(TCHAR));
    FillString(buff,210,'x');
}
``` 

1. Warning C6386: Buffer overrun: accessing 'argument 1',
2. The writable size is ‘200*2' bytes, but '420' bytes might be written
3. The specification for the function. warning C6387: 'argument 1' might be '0': this does not adhere to 'FillString’ __out
Remember this Buffer Overrun?

- Buffer overrun found in IE7 Beta 2 on Jan 31, 2006.
  - `<BGSOUND SRC=file://-------------------------------------------------------------------------------------------
    *----------------------------------------------------------------------------------------------------------------------------
    *----------------------------------------------------------------------------------------------------------------------------
    ---- >`  

- Workaround: Mozilla Firefox 😞

```c
WCHAR pwzTempPath[MAX_PATH];
PathCreateFromUrlW(
    pwzPath,
    &cchPath,
    0);
```
### PREfast & SAL in Action

<table>
<thead>
<tr>
<th>LWSTDAP!</th>
<th>WCHAR pwzTempPath[MAX_PATH];</th>
</tr>
</thead>
<tbody>
<tr>
<td>PathCreateFromUrlW(</td>
<td>PathCreateFromUrlW(</td>
</tr>
<tr>
<td>LPCWSTR pszIn,</td>
<td>pwzPath,</td>
</tr>
<tr>
<td>__out_ecount(*pcchOut) LPWSTR pszOut</td>
<td>(LPWSTR) pwzTempPath</td>
</tr>
<tr>
<td>__inout LPDWORD pcchOut,</td>
<td>&amp;cchPath ,</td>
</tr>
<tr>
<td>DWORD dwFlags )</td>
<td>0);</td>
</tr>
</tbody>
</table>

#### 11/24/2005 5:50 AM Bug # ______ Opened by PREfast

1. **Description:** Potential overflow using expression '& pwzTempPath'  
2. **Buffer access is apparently unbounded by the buffer size.**  
3. **In particular:** cchPath'3485a is not constrained by any constant  
4. **Buffer is of length 260 elements** (2 bytes/element) [size of variable or field]  
5. **Annotation on function PathCreateFromUrlW@16 requires** that {parameter 2} is of length \( \geq \) *{parameter 3} elements (2 bytes/element)  
6. **where** {parameter 2} is & pwzTempPath; {parameter 3} is & cchPath
Did SDL succeed or fail?

- Root cause analysis leads to tools Improvement
  - After PnP RPC bug (Zotob worm), PREfast was improved (warning #2015)
- Process improved to auto-file bugs on #2015
- IE bug identified immediately and filed by PREfast toolset in Nov 2005
  - Caught by PREfast due to SAL annotation on PathCreateFromUrlW API
- Found through internal fuzzing efforts 8 days after public vuln report
- Reported through Windows Error Reporting 1 day later
- Bug was found and would have been fixed by RTM
  - Focus on root cause analysis, continuous tools and process improvements in SDL pays off
File Parsers: Under Attack

- **MS05-036**: ICM (JPG, PNG, BMP)
- **MS05-050**: AVI
- **MS05-002**: 3 ANI
- **MS06-002**: EOT
- **MS05-012**: OLE/COM
- **MS05-014**: CDF
- **MS05-018**: Fonts
- **MS05-022**: GIFs
- **MS05-026**: .ITS
- **MS06-005**: BMP
- **MS06-003**: TNEF
- **MS05-009**: PNG
- **MS05-005**: BMP
- **MS05-010**: MSRatings .RAT
- **MS05-020**: MSRatings .RAT
- **MS05-025**: PNG
- **MS06-004**: WMF
- **MS05-036**: 9 ICM (JPG, PNG, BMP)
- **MS05-053**: EMF
Multi-Prong Approach on Parsers

- Automate what you can:
  - Code coverage helps in “template reduction” & to show basic block coverage
    - 19,154 JPGs were optimized down to 47 with the same block coverage. Fuzzed using 101 templates (3 optimized sets)
  - All parsers: Internally developed general purpose fuzzer
    - Over 83 Million manipulations by Beta 2
  - Highest risk parsers: get Data-Definition-Language extensions
  - Hard targets: Smart fuzzers (Examples: EMF, HTML)
  - Run under debugger with “fail fast” options enabled;
  - Review all Access Violations & Memory Spikes

- Apply security expertise where you need it:
  - Manual code review + detailed program analysis on “problem parsers”
  - Extended SAL annotations for struct members
  - Emit runtime stack protections more aggressively in “attack path”
PART 3:
METHODICALLY APPLY SECURITY EXPERTISE
Feature Reviews

• Features prioritized using multiple risk factors
  o Listens to the network
  o Historical MSRC issues? Legacy code?
  o Accepts anonymous or untrusted input
  o Makes security guarantees
  o Runs with high privilege, and more…

• Feature Reviewer analyzes threat models, design, and attack surface
  o Weak areas referred to pentest for deep inspection

• Internal reviews augmented with security consultants
  o Affinitize reviewer to area of expertise where possible
  o Each reviewer has a MS “driver” to assist with process, pushback
Penetration Testing

• Largest Pentest in MS History
  o Internal team of hackers (remember LSD?)
  o Multiple simultaneous pentests (e.g. TCP/IP)
  o “Blue Hat hackers” :^)
    • 20+ security consultants (aka hackers) in a room
    • Access to Full Source + Symbols, specs, threat models
    • Access to members of product teams, SWI experts
    • All necessary expertise is within 1 building radius

• Modus Operandi
  o Target selection
  o Diagnostic in nature: engineering remediation as necessary
    • You will never pentest your way to a secure product
  o Spend anywhere from 1 week to 2 months per target

• Nothing is out-of-scope
Sampling of Findings

• Process tended to yield “rabbit holes”
• Contradiction in Security Assumptions
  o TM #1: “We have no risk because we don’t parse anything, we just pass things down.”
  o TM #2: “We have no risk because our input is normalized; we just receive validated content.”
• Process handicaps
• Failures of Imagination
• Unwise filenames
  o wls0wndh.dll
  o Winlogon Session0 Viewer Window Hook DLL
It Came from the Codebase…

```
n -= (e = (e = 0x4000 - ((d &= 0x3fff) > w ? d : w)) > n ? n : e);
```

“It’s actually quite beautiful. Almost like a Haiku or something.”

Found by Felix Von Leitner (n.Runs)
How to Identify Truly Old Code?

• Pentesters wanted to focus some time on “legacy code”
• How to identify what’s been around a while?

• “I know, can we find old copies of MSDN?”
How to Identify Truly Old Code?
Windows 95 Was a Simpler Era

Safe String handling circa 1995:

```
LPTSTR SafeStrcpy(LPTSTR lpszString1, LPTSTR lpszString2) {
    try {
        return strcpy(string1, string2);
    }
    except (GetExceptionCode() == EXCEPTION_ACCESS_VIOLATION ?
        EXCEPTION_EXECUTE_HANDLER :
        EXCEPTION_CONTINUE_SEARCH ) {
        return NULL;
    }
}
```

We’ve come a loooooooooooooooong way....
PART 4: DEFENSE-IN-DEPTH MITIGATIONS
Mitigations

• /GS improved in MSVC 8.0 (Visual C++ 2005 aka Whidbey)
  o Significantly improved /GS stack protection from Whidbey compiler
  o Annotations force more aggressive protection in forward facing areas

• Hardened Heap: Many Defense in Depth changes:
  o Lookasides no longer used
  o Arrays of free lists no longer used
  o Early detection of errors due to block header integrity check
  o Dynamic adjustment of algorithms based upon the usage (target attacks)
  o Pseudo-random base address
  o Heap TerminateOnCorruption
    o On by default for 64bit
      o On for services & most apps on x86. still fine tuning
  o See in depth talk by Adrian Marinescu later today
Function Pointer Encoding

- Function pointer encoding
  - Overwriting a function pointer in a predictable location is a common technique to gain control of EIP
  - Encode function pointer with secret ; Decode prior to dereference
  - Decreases reliability of exploit by increasing chances of process termination due to AV on EIP, NX exception, invalid instruction, etc.

- APIs
  - EncodePointer - XOR with a per process cookie
  - EncodeSystemPointer - XOR with a shared cookie in SharedUserData
    - Use to share a pointer cross process
    - Does not require a kernel transition

- Example usages:
  - PEB Lock/Unlock routines
  - Pointers internal to NT heap
  - Vectored exception handler pointers
  - ~300 other usages as of Beta 2, typically long lived function pointers
Data Execution Protection aka NX

• Most Windows Vista PCs will have hardware support for NX
• Default Mode: Opt-in
  o EXEs linked with /NXCOMPAT:YES have NX turned on permanently
  o All Windows services and most EXEs will be opted in
    • Still battling compat issues with IE & Explorer and oddities with ATL thunk emulation
• Customers can put Windows in Opt-out mode
  o Everything has NX turned on
  o Exception list is configurable in registry
However DEP isn’t Good Enough by Itself

As can be seen, the technique described in this document outlines a feasible method that can be used to circumvent the security enhancements provided by hardware-enforced DEP...

First and foremost, the technique depends on knowing the location of three separate addresses...The first dependency could be broken by instituting some form of Address Space Layout Randomization that would thereby make the location of the dependent code blocks unknown to an attacker.

http://www.uninformed.org/?v=2&a=4
mmiller at hick.org, Skywing at valhallalegends.com
Mitigations - ASLR

- Address Space Layout Randomization (ASLR)
  - Powerful complement to NX (aka Data Execution Protection)
- Final design still being tweaked, but currently:
  - Images must opt-in via bit in PE header: DYNAMIC_BASE
    - New linker adds support; also emits reloc in EXEs
  - Limited # of bits available for randomness on 32bit. Main trade-off:
    - How difficult do you want the guess to be?
    - How much contiguous virtual address space do you want to be available to apps?
  - Currently set so that 99.6% of the time, your first guess will fail
  - Still working through issues: service restart, CLR
- Impact:
  - No major hit on performance. Some wins. Some minor losses.
  - Appcompat is looking good with current design (600 apps passed)
- On by default in Beta 2; All of Windows Vista is Opted-in 😊
## SDL Case Study: Zotob Worm

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 2000</td>
<td>Remote unauthenticated code execution possible (No SDL prior to ship)</td>
</tr>
<tr>
<td>Windows XP SP1</td>
<td>Attacker requires authentication to exploit (ACL restricted)</td>
</tr>
<tr>
<td>Windows Server 2003</td>
<td>No remote security threat (Security RPC Callback added)</td>
</tr>
<tr>
<td>Windows XP SP2</td>
<td>No remote security threat (Reviewed and implemented Windows Server 2003 changes)</td>
</tr>
</tbody>
</table>

Even if we had missed it in Windows XP SP2

Blocked by firewall that is on by default
And In Vista?

Windows Vista

And if that failed…

Improved PREfast & PREfix code scanners

And if that failed…

RPC Fuzzing and Penetration Testing

And if that failed…

Protected by an improved version of /GS & SafeSEH

And if that failed…

Protected by NX and ASLR

And if that failed…

And still blocked by firewall that is on by default
Still to Come!

11:15 – 12:30  The NetIO Stack: Reinventing TCP/IP in Windows Vista  Abolade Gbadegesin

13:45 – 15:00  WiFi in Windows Vista: A Peek Inside the Kimono  Noel Anderson & Taroon Mandhana

15:15 – 16:30  Windows Vista Heap Management Enhancements – Security, Reliability and Performance  Adrian Marinescu

16:45 – 18:30  Case Study: The Security Development Lifecycle and Internet Explorer 7  Tony Chor
Questions?