Security Engineering in Windows Vista

Ian Hellen & Vishal Kumar
Security Program Managers, SWI

Contact: ianhelle@microsoft.com vishalku@microsoft.com
A Public Retraction…

Presentation Title: Security Engineering in Windows Vista
Presentation Details:

This paper will present a technical overview of the security engineering process behind Windows Vista. Windows Vista is the first end-to-end major OS release in the Trustworthy Computing era from Microsoft. Come see how we’ve listened to feedback from the security community and how we’ve changed how we engineer our products as a result. The talk covers how the Vista engineering process is different from Windows XP, details from the largest commercial pentest in the world, and a sneak peek at some of the new mitigations in Vista that combat memory overwrite vulnerabilities. It includes behind the scenes details you will only hear from Microsoft!

Why this talk rocks:

Reason 1: Microsoft has benefited immensely over the years from the feedback from the security community and our customers. This presentation is an opportunity to show them how we’ve listened and tried to apply the best of what we’ve heard to Windows Vista.

Reason 2: This talk describes security improvements made in Windows Vista that raise the bar in terms of difficulty of exploitability. Hack In The Box will be the first Asian venue to learn of these changes.

Reason 3: Dave Tamasi is a key member responsible for the security engineering going into Vista. HITB attendees expect to hear authoritative information about the most important things that are going to impact their world. Windows Vista will be one of these products.
Intro – Who Am I (are we)?

Ian Hellen
- Security PM in SWI (Secure Windows Initiative)
- SWI is part of SECD (Security Engineering & Comms Dept.)
- Part of team of security reviewers working on Microsoft product security
- Ran Vista security design reviews
- Joined Microsoft 8 years ago
- 7 years in MCS – security in enterprise customers

Vishal Kumar
- Security PM in SWI
- Responsible for Server and tools products for the past 2 years
- Now part of the design review team for the next Server Release
- Joined Microsoft 4 years ago
- Worked in the Core OS team prior to joining SWI
Goals of Talk

• 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 work on mitigations

• 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
Security Deployment Lifecycle
- Tasks and Processes

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Design</th>
<th>Implementation</th>
<th>Verification</th>
<th>Release</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design</td>
<td>Design guidelines applied</td>
<td>Security architecture</td>
<td>Security docs &amp; tools</td>
<td>Security push training</td>
<td>Feedback loop</td>
</tr>
<tr>
<td></td>
<td>Security architecture</td>
<td>Security design review</td>
<td>Customer deliverables for secure deployment</td>
<td>Review threat models</td>
<td>- Tools/ Processes</td>
</tr>
<tr>
<td></td>
<td>Ship criteria agreed upon</td>
<td></td>
<td>Security push</td>
<td>Review code</td>
<td>- Postmortems</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Design</td>
<td>Security push</td>
<td>Attack testing</td>
<td>- SRLs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>guidelines</td>
<td>Security push training</td>
<td>Review against new threats</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>applied</td>
<td>Review threat models</td>
<td>Meet signoff criteria</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Review code</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Attack testing</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Review against new threats</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Meet signoff criteria</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Windows Vista Security Approach

Stop playing catch up! - Find & fix before ship!

### Our Engineering Process:

<table>
<thead>
<tr>
<th></th>
<th><strong>Apply least privilege throughout architecture</strong></th>
<th>Harden services, applications, browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>Automate proven techniques</strong></td>
<td>Buffer overruns and common coding mistakes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RPC and File parser fuzzing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Banned API removal</td>
</tr>
<tr>
<td>2</td>
<td><strong>Methodically apply security expertise on whole product</strong></td>
<td>Attack Surface Reduction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Threat Model reviews</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Design reviews</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Penetration testing</td>
</tr>
<tr>
<td>3</td>
<td><strong>Defense-in-Depth Mitigations</strong></td>
<td>Firewall on by default</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Enhanced protections for stack, heap, and more</td>
</tr>
</tbody>
</table>
**Windows XP**

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

Buddy Code Review

“Winmain” Main Source Tree

Security Bug Tracking
PREfix, Default Permissions

Across All Code
Windows Vista

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

Quality Gates

“Winmain” Main Source Tree

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

High Risk Code

Across All Code

Security Bug Tracking
- PREfix, Default Permissions
- Threat Models Reviewed
- Legacy Banned API Removal & SAL Annotations
- Weak Crypto Removal, Service Hardening
- 3rd Party Code SDL Tracking
- Mitigations (/GS, etc)
- Privacy Review
- Design & Attack Surface Review
- In Depth Threat Model Review
- Penetration Testing
- Mini-Security Push (if necessary)
- Network and File Parser testing
- Special Cleanup Projects
PART 1: APPLY LEAST PRIVILEGE THROUGHOUT ARCHITECTURE
Pervasive Least Privilege

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

• Approach:
  – Run Applications with Least Privilege
    • Enhance the standard user account
    • Administrators use full privilege only for administrative tasks or applications
    • Run some applications (browser) with heightened restrictions
      – See the IE7 Blackhat talk by Rob Franco for more information
  – Harden Services
    • Minimize privilege user pervasively in 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:
    • Alter the OS
    • Open network connections to propagate
Reduce Privilege Level of Services

- Get services out of SYSTEM to LOCAL SERVICE or NETWORK SERVICE
  - LS & NS are not members of the Administrators group and aren’t assigned the most powerful privileges like SeDebug, SeTcb, etc.
- 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_INFOW 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
  o **LocalServiceNoNetwork**
    • Runs under local service account. This group has no network access and has a write-restricted token
  o **LocalServiceRestricted**
    • Runs under local service account. This group uses a fixed set of network ports on the system and a write-restricted token
  o **LocalServiceNetworkRestricted**
    • Runs under local service account. This group uses a fixed set of network ports on the system but not a write-restricted token
  o **NetworkServiceRestricted**
    • Runs under network service account. This group uses a fixed set of network ports on the system and a write-restricted token
  o **NetworkServiceNetworkRestricted**
    • Runs under network service account. This group uses a fixed set of network ports on the system but not a write-restricted token
  o **LocalSystemNetworkRestricted**
    • Runs under local system account and accesses fixed set of network ports.
## Case Study: DHCP Client Service

<table>
<thead>
<tr>
<th>Account</th>
<th>Windows XP SP2</th>
<th>Windows Vista</th>
</tr>
</thead>
<tbody>
<tr>
<td>Privileges</td>
<td>System 24</td>
<td>Local Service 4</td>
</tr>
<tr>
<td>Network Identity?</td>
<td>Yes (Machine Account)</td>
<td>No</td>
</tr>
<tr>
<td>Uses Fixed Set of Ports?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Data accessible only by service? (Service SID)</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. Example:
  - Given the code `buff[x] = 5;`
  - How big is buff, and what is the value of x?
  - C/C++ doesn’t associate buffers to their sizes

- SAL helps bridge the gap by providing interface contract information to the tools
  - The concept is not new: think IDL in RPC
  - Prime focus is finding buffer overrun bugs
SAL Example

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
void FillString(
  __out_ecount(cchBuf)
  TCHAR* buf,
  size_t cchBuf,
  char ch)

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

Buffer size is an Element count

Buffer is cchBuf elements in size
Adding Annotations

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

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,  
(LPWSTR) pwzTempPath,  
&cchPath,  
0);
```
**PREfast & SAL in Action**

<table>
<thead>
<tr>
<th>Description: Potential overflow using expression ' &amp; pwzTempPath'</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer access is apparently unbounded by the buffer size.</td>
</tr>
<tr>
<td>In particular: cchPath`3485a is not constrained by any constant</td>
</tr>
<tr>
<td>Buffer is of length 260 elements (2 bytes/element) [size of variable or field]</td>
</tr>
<tr>
<td>Annotation on function PathCreateFromUrl@16 requires that {parameter 2} is of length &gt;= *{parameter 3} elements (2 bytes/element)</td>
</tr>
<tr>
<td>where {parameter 2} is &amp; pwzTempPath; {parameter 3} is &amp; cchPath</td>
</tr>
</tbody>
</table>

```c
LWSTDAPI
PathCreateFromUrlW(
    LPCWSTR pszIn,
    _out_ecount(*pcchOut) LPWSTR pszOut,
    __inout LPDWORD pcchOut,
    DWORD dwFlags )
```

```c
WCHAR pwzTempPath[MAX_PATH];
```

```c
PathCreateFromUrlW(
    pwzPath,
    (LPWSTR) pwzTempPath,
    &cchPath ,
    0);
```
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
Multi-Prong Approach on Parsers

• Automate what you can:
  o **All parsers**: Internally developed general purpose fuzzer
    • Over 100 Million manipulations by Beta 2
  o **Highest risk parsers**: get Data-Definition-Language extensions
  o **Hard targets**: Smart fuzzers (Examples: EMF, HTML)
  o Code coverage helps in “template reduction” to improve efficiency
    • Library of >19,000 JPGs were optimized down to 47 with the same block coverage

• Apply security expertise where you need it:
  o Security code review + detailed program analysis on “problem parsers”
  o Extended SAL annotations for struct members
  o Emit runtime stack protections more aggressively in “attack path”
PART 3: METHODOCALLY APPLY SECURITY EXPERTISE
The Human Touch

Windows Vista Features (>600)
Selected Features
Externally Commissioned Pen-tests
Pen-tests

Remaining Features
Selection of High-risk Features (>200)
Design Review
Implementation Review

Horizontal Investigations
RPC / GS Network Listeners
Product Security Improvements
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
  
  o Spend anywhere from 1 week to 2 months per target
  o 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 trusted; we just receive validated content.”
• Failures of Imagination
• Unwise filenames

\[\text{wls0wndh.dll}\]

\textit{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)
PART 4:  
DEFENSE-IN-DEPTH MITIGATIONS
Mitigations - /GS Flag

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

• Many Defense in Depth changes:
  o Looksides 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 Adrian Marinescu’s talk
    (Blackhat US 2006)
## 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

- **Examples:**
  - 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
  - EXEs linked with /NXCOMPAT:YES have NX turned on permanently
  - All Windows services and most EXEs will be opted in
    - Still battling compatibility issues with IE & Explorer and oddities with ATL thunk emulation
- Customers can put Windows in Opt-out mode
  - Everything has NX turned on
  - Exception list is configurable in registry
But 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 #1

- Address Space Layout Randomization (ASLR)
  - Powerful complement to NX (aka Data Execution Protection)

- Current (and probably RTM) design:
  - 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? vs.
      - 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 some issues: service restart, CLR
Mitigations – ASLR #2

- Impact:
  - No major hit on performance. Some wins. Some minor losses.
  - Application compatibility is looking good with current design (600 apps passed)
- On by default from Beta2
- All of Windows Vista is Opted-in 😊
**SDL Case Study: Zotob Worm**

**Windows 2000**
Remote unauthenticated code execution possible (No SDL prior to ship)

**Windows XP SP1**
Attacker requires authentication to exploit (ACL restricted)

**Windows Server 2003**
No remote security threat (Security RPC Callback added)

**Windows XP SP2**
No remote security threat (Reviewed and implemented Windows Server 2003 changes)

Even if we had missed it in Windows XP SP2

Blocked by firewall that is on by default
Zotob Worm in Vista?

Windows Vista

And if that failed…

And if that failed…

And if that failed…

And still blocked by firewall that is on by default

Improved PREfast & PREfix code scanners

RPC Fuzzing and Penetration Testing

Protected by an improved version of /GS & SafeSEH

Protected by NX and ASLR
Don’t Miss...

- 13:15 Tomorrow

Pen-Testing Windows Vista BitLocker
Drive Encryption from the Inside

Douglas MacIver
Questions?