SECURITY IMPROVEMENTS IN WINDOWS VISTA

Michael Howard
Principal Security Program Manager
Microsoft Corp.
mikehow@microsoft.com
Who is this Guy?

- Microsoft employee for >15 years
- Always in security
- Editor for IEEE Security & Privacy
- A pragmatist!
Agenda

- Core Design Assumptions
- Security Development (SDL) Process security contributions
- Isolation
- Service Hardening
- Memory defenses
Code is never perfect
Designs are never perfect
Remember, security is “Man vs. Man”
  - Security is a never-ending arms race
  - You can never be “done” with security
Individual protections may fail
  - Windows Vista includes numerous, layered defenses
  - All enabled by default
  - Each protection raises the bar
But, we must protect customers
High Level Windows Vista Engineering Process

- Prescriptive Guidance
- Mandatory Education
- "Quality Gates"
- Central analysis
- Threat analysis
- External Review

Software Security Science
SDL In Action For Windows Vista

- Weak Crypto banned in new code
  - No use of MD4, MD5 or SHA1.
  - No use of RC4.
  - No symmetric keys smaller than 128 bits allowed.
  - No RSA keys smaller than 1024 bits allowed.

- Threat Modeling
  - Training and tools provided to engineering teams
  - 1,400+ Threat models developed for Windows Vista
  - Security team reviewed models
Mandatory Use of Compiler Security Options
- /GS flag (runtime stack BO detection)
- /SAFESEH (runtime exception checking)
- /NXCOMPAT (NX support)
- /DYNAMICBASE (ASLR support)
- /ROBUST switch for MIDL compiler

Safe Libraries Developed
- 120+ Banned functions
- IntSafe (C safe integer arithmetic library)
- SafeInt (C++ safe integer arithmetic template class)
- Secure CRT (C runtime replacements for strcpy, strncpy etc)
- StrSafe (C runtime replacements for strcpy, strncpy etc)
**Tool Utilization in SDL**

- **TOOLS ARE NOT A PANACEA**
- PREfast – Static code analysis (used by /analyze)
- FxCop – Static analysis of managed code and assemblies
- Standard Annotation Language (SAL)
  - Majority of C Runtime library has been annotated
  - Windows SDK functions have been annotated
Sidebar: What’s SAL?

- Tools can only find “so much” without more contextual information
- SAL helps bridge the gap by providing interface contract information to the tools
- SAL leads to dramatically improved static analysis
  - More bugs
  - Less noise
- The process of adding annotations can find bugs!
- The concept is not new: think IDL
- Included in Visual Studio 2005
void FillString(
    char* buf,
    size_t cchBuf,
    char ch) {

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

   for (size_t i = 0; i < cchBuf; i++) {
      buf[i] = ch;
   }
}
More Extensive Security Testing

- Identify and fuzz all file formats consumed by the operating system
  - Minimum 100,000 malformed files per parser
  - Fuzz many networking protocols, including RPC
- Internal Penetration Testing
- External Penetration testing (thanks to):
  - Code Blau Security Concepts
  - Cybertrust
  - iSec Partners
  - IOActive
  - Matasano
  - Password Consultancy
  - Net-square
  - NGS
  - n.runs
  - Security Innovation
Some Early Results
Security Bulletins that do not affect Windows Vista

- MS06-078 Windows Media Player
  - Banned API removal (wcsncat)
- MS06-069 Flash 6
  - Installed by default in Windows XP, not shipped with Windows Vista
- MS06-066 NetWare Client
  - Installed by default in prior OS’s, removed in Windows Vista
- MS06-055 VML
  - Found through fuzzing
- MS06-050 Windows Hyperlink Object Library
  - Found and fixed because of SAL
- MS07-004 VML
  - Integer overflow calling ::new caught by compiled code
Some Early Results
Interesting figures

- Analysis of 63 buffer-related security bugs that affect Windows XP, Windows Server 2003 or Windows 2000
  - but not Windows Vista
- 82% removed through SDL process
  - 27 (43%) found through use of SAL
  - 26 (41%) removed through banned API removal
Windows Vista Engineering Process (from 35,000ft!)

- Prescriptive Guidance
- Mandatory Education
- "Quality Gates"
- Central analysis
- Threat analysis
- External Review
- Software Security Science
Isolation

- UAC: Users are no longer admins by default
  - Even an admin is not an admin
- Integrity levels help contain damage
  - IE7 runs in low integrity (by default)
    - Protected Mode
  - Most parts of the operating system are medium integrity
  - Restricts “Write-Up”
  - Helps defend integrity of the operating system
DEMO: Integrity Levels
Service Hardening

- Many existing services moved out of SYSTEM
- Describe the privileges you need
- Per-service identity (SID)
  - Protect objects for just that service
  - S-1-5-80-xxxx
- Stricter service restart policy
- Restrict network behavior
  - Eg: foo.exe can only open port TCP/123 inbound
    - Action=Allow | Dir=In | LPORT=123 | Protocol=17 | App=%SystemRoot%\foo.exe
Memory Defenses

- Stack BO detection (aka /GS, enabled by default)
  - Detects many stack-based overruns at runtime
  - Re-arranges the stack so buffers are in higher memory (helps protect variables)
  - Moves various arguments to lower memory
- Exception handler protection (aka /SAFESEH, enabled by default)
  - Exception addresses are verified at runtime
Memory defenses

- Data Execution Prevention (aka NX/XD, enabled by default*)
  - Harder to execute data
- In Windows Vista, DEP cannot be disabled once turned on for a process

*Most CPUs today support DEP, but make sure it’s enabled in the BIOS
Sidebar: Memory defenses and IE7

- By default IE7 does not enable DEP/NX :-(
  - Because too many controls break
  - Many controls use just-in-time compilation
  - They try to run data
  - Fix is to use VirtualProtect(..., PAGE_EXECUTE_READ,...)
- We **will** enable DEP/NX in a future release of IE
Memory Defenses

- Heap defenses (all enabled by default)
  - Lookasides gone 🤖
  - Arrays of free lists gone
  - Early detection of errors due to block header integrity check
    - ENTRY->Flink->Blink == ENTRY->Blink->Flink == ENTRY
  - Heap terminate on corruption
- Integer overflow calling `operator::new` automatically detected at runtime (by default)
Image randomization (ASLR)
  ▪ System images are loaded randomly into 1 of 256 ‘slots’
  ▪ Changes on each boot
  ▪ **To be effective ASLR requires DEP**
  ▪ Enabled by default
  ▪ Link with /DYNAMICBASE for non-system images

Stack is randomized for each new thread (by default)

Heap is randomized (by default)

Long-lived pointers are encoded and decoded
  ▪ A successful pointer overwrite must survive the decoding process (XOR with a random number)
DEMO: Memory Defenses
The coding vulnerability was in the code

The attacker had to:
  - Get passed the firewall
  - Bypass /GS
  - Bypass SafeSEH
  - Bypass NX
  - Bypass ASLR
  - Bypass stack randomization
  - Bypass service hardening

And the attacker has only two attempts
  - Because of service restart policy
Windows Vista Vulnerability Reduction to Date

- **Vulns**: 70 (53% Vulnerability Reduction)
- **Bulletins**: 41 (66% Bulletin Reduction)

Comparison between Windows XP SP2 and Windows Vista.
Software Security Science

- Security is “Man vs. Man”
- We must continue to innovate
- We must continue to learn more about attackers
  - And how to thwart them
- We perform root-cause analysis of each security bug
- We analyze bugs from around the industry
- We work closely with security researchers
- Feeds back into the SDL twice a year
Call to Action

- **Process**
  - Evaluate the SDL (it works!)
  - Build threat models
  - Utilize all available tools (eg: compiler, /analyze, SAL etc)
  - Perform fuzz testing
  - Hire expert pentest help

- **Engineering**
  - Remove banned APIs
  - Compile with /GS
  - Link with /NXCOMPAT, /SAFESEH and /DYNAMICBASE
Questions?