Standing on the shoulders of the **Blue Monster:**

**Hardening Windows Applications**

olleB
olle@toolcrypt.org

The Toolcrypt Group
www.toolcrypt.org
Agenda

• Introduction to Windows security model
• Windows security-related features
• Strategies for hardening Windows Apps
• Question time
Intro to Windows security model

• Security Identifiers
• Security Descriptors
• Access Control Lists
• Objects and Handles
• Tokens and Privileges
Intro to Windows security model

- Security Identifiers (SIDs)
  - Authority, $n \times$ Sub-Authority, Relative ID

Example: S-1-5-32-544

<table>
<thead>
<tr>
<th>Revision</th>
<th>Authority</th>
<th>Sub Authority</th>
<th>RID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>32</td>
<td>544</td>
</tr>
<tr>
<td>First</td>
<td>“NT”</td>
<td>Builtin</td>
<td>Administrators</td>
</tr>
</tbody>
</table>
Intro to Windows security model

- Security Descriptors

<table>
<thead>
<tr>
<th>SECURITY_DESCRIPTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header (revision number and control flags)</td>
</tr>
<tr>
<td>Owner SID</td>
</tr>
<tr>
<td>Group SID (used for POSIX compatibility)</td>
</tr>
<tr>
<td>DACL (Discretionary Access Control List)</td>
</tr>
<tr>
<td>SAACL (System Access Control List)</td>
</tr>
</tbody>
</table>
Intro to Windows security model

- Access Control Lists (ACLs)
  - Lists of Access Control Entries (ACEs)
    - DACLs list “access permissions”
    - SACLs list system info (auditing, etc.)

<table>
<thead>
<tr>
<th>ACL Contents:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revision</td>
</tr>
<tr>
<td>ACE Count</td>
</tr>
<tr>
<td>ACE [0]</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>ACE [n]</td>
</tr>
</tbody>
</table>
Intro to Windows security model

• Access Control Entries (ACEs)
  – Type and Flags determine meaning
  – Checked in order (first match, default deny)

ACE Contents:

<table>
<thead>
<tr>
<th>Type (Allow, Deny, Audit, etc.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flags (inheritance, etc.)</td>
</tr>
<tr>
<td>Access Mask (e.g. GENERIC_READ)</td>
</tr>
<tr>
<td>Trustee SID</td>
</tr>
</tbody>
</table>
Intro to Windows security model

- Objects and Handles
  - Kernel Objects are ref-counted structs
    - Common header (with type, refcounts, etc.)
    - Contains Security Descriptor => “Securable Object”

<table>
<thead>
<tr>
<th>OBJECT_HEADER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pointer Reference Count</td>
</tr>
<tr>
<td>Handle Reference Count</td>
</tr>
<tr>
<td>Type (Pointer to type object)</td>
</tr>
<tr>
<td>Misc. Flags and Control Structures</td>
</tr>
<tr>
<td>Security Descriptor</td>
</tr>
</tbody>
</table>
Intro to Windows security model

• Objects and Handles
  – Kernel ref by pointer, usermode by handle
  – Handles are kept per-process in kernel tables
  – Many different processes can have “open” handles to same Kernel Object
  – Handles closed by CloseHandle() or Exit()
  – Objects destroyed when refcounts reach 0
Intro to Windows security model

- Tokens and Privileges
  - An Access Token is a Securable Object
  - Describes security context of process (or thread)

<table>
<thead>
<tr>
<th>TOKEN (abridged)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOKEN_SOURCE</td>
</tr>
<tr>
<td>Privileges</td>
</tr>
<tr>
<td>User/Group SID  count</td>
</tr>
<tr>
<td>User/Group SID list</td>
</tr>
<tr>
<td>Impersonation Level</td>
</tr>
</tbody>
</table>
Intro to Windows security model

• Tokens and Privileges
  – Processes get a “Primary Token” at creation
  – Process or thread can temporarily have different Token assigned by “Impersonation” (or delegation)
  – Privileges can be assigned to users / groups
  – Privileges stored in Token, must be “enabled”
Windows security-related features

- Restricted Tokens
- Desktop Objects and Window Stations
- Job Objects
- MIC / UAC / UIPI
- Memory protection
- Exploit mitigations
Windows security-related features

• Restricted Tokens
  – CreateRestrictedToken()
    • Remove Privileges from Token
    • Prevent SIDs from granting accesses
    • Restrict SIDs list to a certain subset
  – CreateProcessAsUser()
    • Normally requires SeTokenPrivilege
      – not with Restricted version of callers’ Primary Token
      – which becomes the Primary Token of new process!
Windows security-related features

- Desktop Objects and Window Stations
  - Session => Window Station => Desktop
    - Winsta0 only interactive Window Station
    - Interactive Desktop selected by SwitchDesktop()
  - Processes assigned to a Window Station
  - Threads assigned to a Desktop
  - Desktop is container for UI objects
    - Windows, message queues, etc.
Windows security-related features

- **Job Objects**
  - Container for processes
    - Processes can be associated to Job Object
    - Processes created inherit Job Object association
  - Imposes limits on associated processes
    - Memory / CPU usage limits
    - Prohibit access to SwitchDesktop() 
    - Prohibit access to UI objects (e.g. clipboard)
    - Prohibit access to sensitive APIs
Windows security-related features

• MIC – Mandatory Integrity Control
  – “Mandatory Label” new ACE in SAACL
    • RID in SID of ACE defines “Integrity Level”
    • ACE attributes define a policy
      – NoWriteUp, NoReadUp, NoExecuteUp
  – Label defaulted if not explicitly present
    • Objects default to “Medium” and NoWriteUp
    • Processes to “Medium” and NoWriteUp / NoReadUp
  – Anyone with WRITE_OWNER can set lower IL
    • Need SeRelabelPrivilege to set higher IL than own
Windows security-related features

- **UAC** – User Account Control
  - Admin users run as Standard by default
  - “Elevation” required to use Admin rights
  - Privilege separation by “Linked Tokens”
  - New service “AppInfo” controls Elevation
  - Apps request Admin rights using Manifest
Windows security-related features

• UIPI – User Interface Privilege Isolation
  – Blocks windows messages between windows of processes with differing Integrity Level
  – “Message Filter” is list of allowed messages
  – ChangeWindowsMessageFilter()
    • Processes at or below “Low” IL cannot use
Windows security-related features

- Memory protection
  - Hardware can enforce access permissions on “pages” of virtual memory space
  - Permission bits in PTE => R / W / X
  - VirtualProtect(), VirtualAlloc()
Windows security-related features

- Exploit mitigations
  - Stack overwrite protection
  - Heap overwrite protection
  - Safe SEH, SEH Overwrite Protection
  - Data Execution Prevention
  - Address Space Layout Randomization
Windows security-related features

• Exploit mitigations

  – Stack overwrite protection (or “/GS”)
    • Inserts “cookie” value into stack frame
      – Check integrity of cookie before returning
      – Protects return address and stack variables
    • Default compiler option since VS 2003
Windows security-related features

- Exploit mitigations
  - Heap overwrite protection
    - Check forward / back links when unlinking lists
      - In all Windows versions since XP SP2
    - XORing / checksumming to “detect” overwrites
      - Since XP SP3, increasing protection in Vista
        - HeapSetInformation(HeapEnableTerminationOnCorruption)

- Don't use third-party dynamic memory managers!
Windows security-related features

• Exploit mitigations

  – Safe SEH
    • Linker inserts table of known exception handlers
    • “/SAFESEH” option available since VS 2003

  – SEH Overwrite Protection (SEHOP)
    • Checks integrity of exception handler chain
    • Available since Vista SP1, Server 2008
    • Disabled by default on client systems
Windows security-related features

• Exploit mitigations

  – Data Execution Prevention (DEP)
    • Makes stack and heap non-executable by default
    • Modes: OptIn/OpOut/AlwaysOn/AlwaysOff
    • SetProcessDEPPolicy() or “/NXCOMPAT”

  – Address Space Layout Randomization (ASLR)
    • Complement to DEP prevents simple bypasses
    • Available in Vista and later, for supporting modules
    • Link all modules with “/DYNAMICBASE” to enable
Strategies for hardening Windows Apps

• Standing on Microsoft's shoulders
• Securing your application boundaries
• Partitioning your application code
• Wrapping the onion (in tin foil)
Strategies for hardening Windows Apps

• Standing on Microsoft's shoulders
  - Use exploit mitigations
    • always build with latest version of toolchain
  - Read and adopt from the SDL
    • Architecture review and Threat Modelling
    • Secure coding guidelines! (Musts and Don'ts)
  - Use safe libraries and templates
    • SafeInt / intsafe.h
    • Banned APIs / Secure Template Overload
Strategies for hardening Windows Apps

• Securing your application boundaries
  – Architect using modular components
  – Make sure components aren't too large
  – Identify interfaces to other components
    • Data flows
    • Execution flow
  – Apply safe default DACLs on resources
Strategies for hardening Windows Apps

• Partitioning your application code
  – Sandbox parsers and data processing
    • Reduces complexity required in core code
    • Verify syntax and semantics of simplified input
  – Examine each app component
    • Make sure interfaces are simple and clear
    • Apply the principle of least privilege
    • Redesign if complex or requires too many privs

Strategies for hardening Windows Apps

• Wrapping the onion (in tin foil)
  – No “Silver Bullets” or “Magic Fairy Dust”
    • Goal: Raising the costs of the attacker!
  – Build security controls in layers
    • Overlapping controls build resilience
  – Each component is “autonomous”
    • Responsible for its own security
    • Assumes nothing of input, validates
Summary

- Build security in using overlapping checks
- Architect components and review them
- Divide your application into partitions
- Use Windows features to raise the bar
- Adopt what parts of the SDL make sense
Questions?
Corrections?
Additions?
Thank You for listening!

http://www.toolcrypt.org/

mailto:olle@toolcrypt.org

follow @toolcrypt on twitter