Privilege Escalation in Windows OS

by Zohaib & Vlad

Follow the slides: goo.gl/bvmYgb
What is Privilege Escalation?

An act of exploiting a bug, design flaw or configuration oversight with the goal to gain elevated access to application resources

- Gives the ability to perform unauthorized actions in software, web apps, operating systems
What is Privilege Escalation?

Vertical
Accesses to functions that are reserved for higher privilege users or applications.
- gaining administrative privileges
- Jailbreaking Devices
- Lock Screen Bypass

Horizontal
Accesses functions that are accessible by other normal users.
- Accessing accounts on the same user level
- Stealing usernames/passwords
Vertical: Cross-zone scripting

A web browser exploit that takes advantage of a zone-based vulnerability

http://windowsupdate.microsoft.com%2f.example.com/
Windows Permission Structure

- root is “Local System” Account
- Windows UAC (User Account Control)
  - disabled admin account, instead uses UAC
- “sudo” is “runas” to run with privileges

Types of Accounts:
- Local User
- Domain User
- The LocalSystem
Windows with User Access Control

- All users run as an unprivileged user by default, even when logged on as an Administrator.

- Once running, the privilege of an application cannot be changed.

- Users are prompted to provide explicit consent before using elevated privilege, which then lasts for the life of the process.
Windows OS Privilege Escalation

- replacing “screensaver” binary
- scan the registry for
  - logon Information
  - network credentials
  - private keys
  - Many different tools that will do this task for you
Windows OS Privilege Escalation

- exploit design flaws
  - find processes that run as SYSTEM using GUI or tools

- retrieve user hashes
  - Retrieve a user hash from Local Security Authority Subsystem Service (LSASS)
  - Corrupt the memory and use the hash
Windows OS Privilege Escalation

- missing autorun programs
- service quoting
Windows OS Privilege Escalation

- **Internet Explorer Elevation Policy**

<table>
<thead>
<tr>
<th>Integrity Access Level (IL)</th>
<th>System Privileges</th>
<th>Value</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Administrative (Process can create thread)</td>
<td>3</td>
<td>Protected Mode silently launches the broker as a medium integrity process.</td>
</tr>
<tr>
<td>Medium</td>
<td>User (Process can create thread)</td>
<td>2</td>
<td>Protected Mode prompts the user for permission to launch the process.</td>
</tr>
<tr>
<td>Low</td>
<td>Untrusted (Process cannot create thread)</td>
<td>1</td>
<td>Protected Mode silently launches the broker as a low integrity process.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>Protected Mode prevents the process from launching.</td>
</tr>
</tbody>
</table>

Compromise IE

Start a server on localhost

Start a server on localhost
Windows OS Privilege Escalation

- services run under Local System or with Elevated flags through stolen access tokens

*Recent Vulnerability Demo*
Security Tokens in Windows

- Access Token
- Impersonation Token
- Impersonation

<table>
<thead>
<tr>
<th>Impersonation level</th>
</tr>
</thead>
<tbody>
<tr>
<td>SecurityAnonymous</td>
</tr>
<tr>
<td>SecurityIdentification</td>
</tr>
<tr>
<td>SecurityImpersonation *</td>
</tr>
<tr>
<td>SecurityDelegation</td>
</tr>
</tbody>
</table>
Exploit Details

- Allows to run services and programs with Elevated privileges
- Severity Rating: “Important”
- Proof of Concept: Disables UAC popup
- The token may allow you to:
  - inject DLLs into system processes
  - start up ASPNET / IIS server processes
  - get access to LOCAL SYSTEM
Writing an exploit

- Find an auto-elevated executable
  - Such as “ComputerDefaults.exe”
  - These executables set up a cache point in the registry (regsvr32.exe)
- Look up Application Compat DB: sysmain.sdb
- Capture the “Impersonation Token”
  - by using the vulnerability in the cache system
- Start a new process using “runas”
- Assign its impersonation token using “SetThreadToken” and it set a “SecurityImpersonation” level
Google’s Project Zero

- Aims to improve the security of any software
- Locating and reporting large number of vulnerabilities
- Issues are filed in an external database which is initially reported to vendor
- 90-day policy
Disclosing the vulnerability early

Google posts Windows 8.1 vulnerability before Microsoft can patch it

Source: Engadget

SECURITY microsoft, windows 8.1

Google outs unpatched Windows 8.1 vulnerability, and debate rages on both sides

Source: PCWorld
Disclosing the vulnerability early

- Who do you think is right in this issue Google or Microsoft?

- Do you think 90-days is fair for vulnerability disclosure?

- How much time should a vulnerability patch take? (90-days, 180-days?)