Bypassing Windows services protections

Cesar Cerrudo
Argeniss

Who am I?

- Argeniss Founder and CEO (23 people company)
- I have been working on security for +9 years
- I have found and helped to fix hundreds of vulnerabilities in software such as MS Windows, MS SQL Server, Oracle Database Server, IBM DB2, and many more...
- +50 vulnerabilities found on MS products (+20 on Windows operating systems)
- I have researched and created novel attacks and exploitation techniques

Agenda

- Introduction
- What is impersonation and what are tokens?
- Windows 7, Vista and 2008 services hardening
- Session 0 isolation
- Least privilege
- Per service SID
- Write restricted token
- Restricted network access
- Conclusions

Introduction

- In the beginning all Windows services ran as Local SYSTEM account
  - Compromise of a service==full system compromise
- Later Windows services run under Local Service, Network Service and Local System accounts
- Since Windows Vista new services protections were introduced and previous weaknesses were corrected
  - Compromise of a service!=full system compromise
- But Windows is still not perfect and most new services protections can be easily bypassed...

What is impersonation and what are tokens?

- Impersonation is the ability of a thread to execute using different security information than the process that owns the thread
  - ACL checks are done against the impersonated users
  - Impersonation APIs: ImpersonateNamedPipeClient(), ImpersonateLoggedOnUser(), RpcImpersonateClient()
  - Impersonation can only be done by processes with "Impersonate a client after authentication" (SeImpersonatePrivilege)
  - When a thread impersonates it has an associated impersonation token

What is impersonation and what are tokens?

- Access token is a Windows object that describes the security context of a process or thread
  - It includes the identity and privileges of the user account associated with the process or thread
  - They can be Primary or Impersonation tokens
    - Primary are those that are assigned to processes
    - Impersonation are those that can be get when impersonation occurs

Windows 7, Vista and 2008 services hardening

- Services hardening consists of new services protections implemented as a defense in depth mechanism
  - ***Session 0 isolation***
  - ***Least privilege***
  - ***Per service SID***
  - ***Write restricted token***
  - ***Restricted network access***
- All protections working together **"could contain"** exploitation by limiting attacker actions when exploiting a vulnerability

Session 0 isolation

- System and Services processes runs on Session 0 and user processes on Session 1, 2, etc.
- If an interactive service displays a window, user is asked to switch to Session 0 by "Interactive Services Detection" service (UI0Detect)
  - UI0Detect displays a window telling the user about a process trying to display a message
- User sessions can't send windows messages to Session 0
- *Goal:* ***Protect against Shatter attacks***

Session 0 isolation

- Protection bypass
  - None known
  - Interactive services not common

Least privilege

- Services run only with required privileges
  - For services sharing a single svchost processes privileges are accumulated
- Privileges can be added with
  - sc privs [service name] [Privileges]
- Privileges can be queried with
  - sc qprivs [service name]
- *Goal:* ***if a service is compromised attacker actions will be restricted due to few available privileges***

Least privilege

- Protection bypass
  - Most services need and have impersonation privilege
    - When impersonating, service process gets privileges from impersonated account and could elevate privileges
  - Windows tasks can be created from a service and then run with full privileges
    - Services can run any program with full service account privileges
  - Demo
    - Creating a Windows task with full privileges from a service

Per service SID

- Nice feature, now service processes are well protected, unique SID assigned to process ACL
- Service running under “X” account can't access other services running under same “X” account
- Setting a Per service SID
  - sc sidtype [service name] unrestricted (or restricted)
- IIS worker processes and WMI process have similar protection
- Goal: ***restrict access to processes running under same account blocking privilege elevation***

Per service SID

- Protection bypass
  - ACLs checks are performed against service SID and also service account
    - Shared registry keys and files break and weaken this protection
  - Some processes aren't properly protected
    - SQL Server WMI process
    - Windows task processes
  - Demo
    - Elevation of privileges exploit (new Token Kidnapping exploit)

Write restricted token

- Nice feature, service can have write access to resources only if explicitly granted to the service SID, logon SID, Everyone SID or write-restricted SID
- Service account gets restricted from writing to resources
  - Good protection for registry keys and files
  - Needs proper ACLs for used resources
- Doesn't restrict reading
- Windows Firewall service runs with write restricted token (sc qsidtype MpsSvc)
- Goal: ***if a service is compromised attacker will be restricted to write to resources***

Write restricted token

- Protection bypass
  - Just a couple of services are write restricted by default (Windows Firewall service)
    - These services can and do impersonate SYSTEM account and administrative accounts
      - No sense in making them restricted since they can compromise Windows after impersonating high privileged accounts
      - After exploiting service just wait for an administrator to log in in order to elevate privileges
  - Demo
    - Windows Firewall service impersonating an administrator

Restricted network access

- Services can be restricted to only make or accept connections on specified ports and protocols
  - They could have no network access at all
- Implemented as Windows Service Hardening (WSH) rules through Windows firewall
  - WSH rules evaluated before firewall ones
  - Can't be disabled after service starts
  - Works no matter firewall is disabled
  - Processes created by a service are restricted too
- Goal: ***if a service is compromised attacker will be restricted to accept or make connections***

Restricted network access

- Protection bypass
  - On Win2008 Local Service account has full control over non native services WSH rules registry key
    - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\SharedAccess\Parameters\FirewallPolicy\RestrictedServices\Configurable
    - Restrictions can be removed while service is still running if attacker can run code as Local Service account
  - All network restrictions can be bypassed by executing code under another process not directly created by the network restricted service such as Windows task process or SQL Server WMI process
- Demo

Conclusions

- New services protections can be easily bypassed
  - Just require an additional exploit step
- More Windows design changes must be done to make protections stronger

References

- Session 0 isolation

http://www.alex-ionescu.com/?p=59

http://blogs.technet.com/b/voy/archive/2007/02/23/services-isolation-in-session-0-of-windows-vista-and-longhorn-server.aspx

- Windows Services Hardening

http://blogs.technet.com/b/askperf/archive/2008/02/03/ws2008-windows-service-hardening.aspx

- Least Privileges

http://blogs.technet.com/b/voy/archive/2007/03/21/least-privilege-for-services.aspx

References

- Per Service SID

http://blogs.technet.com/b/voy/archive/2007/03/22/per-service-sid.aspx

- Write Restricted Token

http://blogs.technet.com/b/voy/archive/2007/04/01/write-restricted-token.aspx

- Network Restrictions

http://blogs.technet.com/b/voy/archive/2007/04/02/network-restrictions-for-service-hardening.aspx

Fin

- Questions?
- Thanks

- Contact: cesar>at<argeniss>dot<com

Argeniss – Information Security & Software

WE BREAK ANYTHING

www.argeniss.com