



Token Kidnapping

Cesar Cerrudo
Argeniss

Who am I?

- Argeniss Founder and CEO
- I have been working on security for 7 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...
- I have researched and created novel attacks and exploitation techniques
- I have spoken around the world at most important security conferences
- I have never written a book



Agenda

- Introduction
- What is impersonation and what are tokens?
- Windows XP and 2003 services security
- Windows XP and 2003 services security weaknesses
- Windows Vista and 2008 services security
- Windows Vista and 2008 services security weaknesses
- Token Kidnapping in action
- Conclusions



Introduction

- In the beginning all Windows services ran as Local SYSTEM account
 - Compromise of a service == full system compromise
- Then MS introduced NETWORK SERVICE and LOCAL SERVICE accounts
 - Compromise of a service != full system compromise
- Then with Windows Vista and 2008 new protections were introduced and some previous weaknesses were corrected
- But as we are going to see Windows is still not perfect...

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
 - Threads impersonate to run code under another user account, ACL checks are done against the impersonated users
 - Impersonation can only be done by processes with the following privilege:
 - “Impersonate a client after authentication” (SeImpersonatePrivilege)
 - When a thread impersonates it has an associated impersonation token



What is impersonation and what are tokens?

- An access token is an 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 ones are those that are assigned to processes
 - Impersonation ones are those that can be get when impersonation occurs
 - Four impersonation levels: SecurityAnonymous, SecurityIdentity, SecurityImpersonation, SecurityDelegation



Windows XP and 2003 services security

- Services run under
 - LOCAL SYSTEM, NETWORK SERVICE, LOCAL SERVICE and user accounts
- Services seemed to be armoured
 - Processes are created with “special” permissions
 - A service running under “X” account can't directly access another service running under the same account
 - Gentle Security found that services were improperly protected and that service account has WRITE_DAC permissions on service



Windows XP and 2003 services security

- All services can impersonate
 - If a service can get a SYSTEM impersonation token the game is over
 - This doesn't happen always in all services
 - Impersonation takes place mostly during Inter Process Communication (IPC) using Local Procedure Call (LPC), Named Pipes, etc.
 - Impersonation can be limited by clients by setting proper options in the used functions



Windows XP and 2003 services security weaknesses

- While service processes are not well protected, threads aren't either*
 - Service threads have default account permissions
 - A service running under X account can access threads of another services running under the same account
 - Service X can run arbitrary code on service Y
 - Service X can get impersonation tokens from service Y

*Brett Moore from Insomnia Security (www.insomniasec.com) independently found the services thread permissions issue



Windows XP and 2003 services security weaknesses

- While service processes are not well protected, threads aren't either
 - Threads from Rpcs service process (runs under NetworkService) can be accessed
 - This process always has impersonation tokens from many different accounts including SYSTEM
 - Services will need first to get NetworkService impersonation token and then use it to access Rpcs threads



Windows XP and 2003 services security weaknesses

- Calling APIs that interacts with a service ends up getting the service account impersonation token
 - Calling process only needs to be able to “impersonate”
 - If impersonation tokens have higher privileges then calling process can elevate privileges
 - Problem present in MSDTC (runs under NetworkService)
 - Call DtcGetTransactionManagerEx() to get NetworkService impersonation token
 - The function starts MSDTC if not running



Windows XP and 2003 services security weaknesses

- Both weaknesses combined lead to full system compromise just having Impersonation rights
 - Any service can run code as SYSTEM
 - Any ASP web page, CGI, etc. on IIS can run code as SYSTEM
 - Any SQL Server administrator can run code as SYSTEM
 - Etc.



Windows Vista and 2008 services security

- Huge improvements in latest Windows versions (at least in theory)
- Session 0 isolation
 - Not big deal, mostly protect against Shatter attacks
- Least privilege
 - Not big deal, most Windows services requires Impersonation privileges
- Per service SID
 - Nice feature, now the service process it's really protected and its resources can be armoured



Windows Vista and 2008 services security

- Per service SID
 - Service running under X account can't access other service resources no matter the service is running under same account
 - Threads are now properly protected
- 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



Windows Vista and 2008 services security

- Restricted network access
 - Nice feature
 - Services can only accept connections on specified ports and protocols
 - Services can only make connections to specified ports and protocols
 - Services can be restricted to have no network access
 - Implemented as firewall rules
 - Can't be disabled after service starts



Windows Vista and 2008 services security weaknesses

- Per service SID weaknesses
 - While regular threads are properly protected, threads from thread pools are not
 - Service running under X account can submit work to thread pools on other services running under same account
 - This means arbitrary code execution bypassing per service SID protection



Windows Vista and 2008 services security weaknesses

- Per service SID weaknesses
 - While service processes are protected some regular processes running under LOCAL SERVICE and NETWORK SERVICE are not
 - Service process running under X account can access regular processes running under same account
 - Services can execute arbitrary code on other processes
 - WMI processes have this problem
 - » They impersonate SYSTEM account



Windows Vista and 2008 services security weaknesses

- Write restricted token weaknesses
 - Just a couple of services are restricted by default
 - These restricted services can and do Impersonate SYSTEM account and administrative accounts
 - eg.: when an administrator configures Windows Firewall, the Windows Firewall service impersonates the administrator and SYSTEM account
 - No sense in make them restricted since they can own Windows after impersonating SYSTEM
- Restricted network access weaknesses
 - A service can easily bypass all restrictions by executing code under another process



Token Kidnapping in action

- Windows XP & 2003
 - Since threads are not protected they can be easily manipulated
 - Using `SetThreadContext()` the thread can execute any code in target process
 - Need to have the some code already on target process
 - Brett Moore cool technique using WLSI to build a call stack and then set proper thread context
 - Using thread manipulation techniques from c0de90e7
 - Code can be executed without putting any code on target process
 - Techniques needs to find proper op codes



Token Kidnapping in action

- Windows XP & 2003

- An APC can be submitted to a thread

- QueueUserAPC() can be called with ImpersonateSelf() as parameter
 - Thread starts to impersonate service account
 - Impersonation token is get by OpenThreadToken()
 - Token is used to access the process
 - Token handles are brute forced in target process until SYSTEM token is found
 - SYSTEM token is used to run code



Token Kidnapping in action

- Windows XP & 2003
 - RpcSs service is the best target for getting SYSTEM token
 - Attacker must have a NetworkService impersonation token
 - Attacker can get NetworkService impersonation token just calling DtcGetTransactionManagerEx()
 - SQL Server exploit demo
 - IIS 6 exploit demo



Token Kidnapping in action

- Windows Vista & 2008
 - Unprotected thread on pools don't resume execution unless work is submitted to the pool
 - We have to wait in order to manipulate the thread, it can take arbitrary time unless we can trigger some action to get a thread executing
 - APC can be used to get code executed
 - APC on a thread from a pool can't be manipulated by `SetThreadContext()`
 - Calling `ImpersonateSelf()` crashes target process, an APC in a thread from a pool can't end impersonating



Token Kidnapping in action

- Windows Vista & 2008
 - APC can be used to get code executed
 - Need to call a useful function that allows to execute code in order to elevate privileges
 - LoadLibrary() can be called to get code executed
 - We only need to find a pointer to a letter in memory for dll name
 - » .dll extension is automatically appended
 - Dll must be in dlls search paths or full path must be provided
 - » We need permissions to copy dll or we need a way to put a string in target process
 - Code can be executed in this way but there is an easier way...



Token Kidnapping in action

- Windows Vista & 2008
 - Getting SYSTEM token from WMI process (WmiPrvSE.exe)
 - This process runs under NetworkService, LocalService or SYSTEM accounts
 - This process is not protected and it impersonates SYSTEM account
 - Services running under NetworkService and LocalService can get SYSTEM token from it
 - Invoke WMI functionality
 - Patch CloseHandle() and OpenThreadToken() on WMI process
 - Brute force token handles until SYSTEM token is found



Token Kidnapping in action

- Windows Vista & 2008
 - RpcSs Dll injection demo
 - A Dll is injected into RpcSs service from an ASP .NET web page, the site is running under NetworkService account (default)
 - Bypass per service SID
 - RpcSs injects the same Dll into IIS service (runs as SYSTEM), this service then runs a reverse shell
 - Bypass least privilege
 - Bypass restricted network access
 - IIS 7 exploit demo
- *All demos are with Windows 2008 default firewall settings, just World Wide Web Services (HTTP Traffic-In) enabled



Recomendations

- Windows XP and Windows 2003
 - On IIS 6 don't run ASP .NET in full trust and if classic ASP is enabled don't allow users to execute binaries
- On Windows Vista and 2008
 - On IIS 7 don't run ASP .NET in full trust or don't run web sites under NetworkServer or LocalService accounts
 - Don't run services under NetworkService or LocalService accounts
 - Use regular user accounts to run services



Conclusions

- On Windows XP and Windows 2003
 - If a user can impersonate then game is over
 - User can execute code as SYSTEM
- On Windows Vista and 2008
 - LocalService == SYSTEM
 - NetworkService == SYSTEM
 - New services protections are almost useless



References

- Impersonate a client after authentication

<http://support.microsoft.com/kb/821546>

- Access tokens

<http://msdn2.microsoft.com/en-us/library/aa374909.aspx>

- Thread manipulation

http://www.rootkit.com/vault/c0de90e7/gw_ng.c

- The weakness in the Windows impersonation model

<http://www.gentlesecurity.com/04302006.html>

- Process explorer

<http://www.sysinternals.com>





Fin

- Questions?
- Thanks
- Contact:
cesar>at<argeniss>dot<com

Argeniss – Information Security

WE BREAK ANYTHING

www.argeniss.com