

SYMTEX-Explosive Firewall Subversion

Lin0xx
Noxusfiles.com
Offensive Vulnerability Research

Agenda

- Reversing Symantec NIS 2007
- Next month – Norway
- XPFW_BIND shellcode is now in metasploit
- This is the last firewall I do for a while ;)

Introduction

- NIS is a full 'internet security suite'
- Anti-virus, anti-phishing, firewall
- Anti-debug techniques, of course
 - That's what I always go after
- Unique way of anti-debugging
- Mainly about the process of discovery rather than subverting it

Anti-Debug

- Rootkit Hook Analyzer shows only 4-5 calls being hooked
 - Rootkit Hook Analyzer is retarded.
- I didn't know this at the time, so I started debugging usermode APIs

Debugging the Debugger

- Start off with the obvious – `OpenProcess`
 - It's successful!
- Better go further
- `DebugActiveProcess` fails
 - Why?

DebugActiveProcess

- Disassemble DebugActiveProcess and peel the onion
 - Pretty much converts the PID into a handle and then calls other native APIs
- DbgUiDebugActiveProcess
- NtDebugActiveProcess?
 - No, works – again, quite surprising
- DbgUiIssueRemoteBreakin
 - Bingo
 - <shift to IDA view>

DbgUiIssueRemoteBreakin

- Let's think – what is a remote break in?
- Remember when you click 'attach to process' in a debugger and then you've broken somewhere?
 - This is the call that does it
- RtlCreateUserThread is called...
 - push offset _DbgUiRemoteBreakin@4 ; start address of thread
 - What does that do?

DbgUiRemoteBreakin

```
.text:7C950787      mov     eax, large fs:18h
.text:7C95078D      mov     eax, [eax+30h]
.text:7C950790      cmp     byte ptr [eax+2], 0
.text:7C950794      jnz     short loc_7C95079F
.text:7C950796      test   byte ptr ds:7FFE02D4h, 2
.text:7C95079D      jz      short loc_7C9507BF
.text:7C95079F
.text:7C95079F loc_7C95079F:      ; CODE XREF:
    .text:7C950794j
.text:7C95079F      and     dword ptr [ebp-4], 0
.text:7C9507A3      call   _DbgBreakPoint@0 ; DbgBreakPoint()
```

Continued

```
if(isDebugged()){ //if process is debugged..
    DbgBreakPoint();
}
DbgBreakPoint(){
    __asm{
        int 3          //It breaks.
        ret
    }
}
```

RtlCreateUserThread

- All threads need a stack right?
- RtlpCreateStack is the first thing called
 - Dies
- 0 was passed to the 'stack commit' section of RtlCreateUserThread
 - This gets you 1 page of memory by default
- NtAllocateVirtualMemory
 - NT_STATUS == 0xc0000022 == ACCESS_DENIED

* @ # (- Now What?

- Since the rootkit analysis program lied, assume there's some sort of uber 31337 dkom crap
- Rolf recommended I try olly advanced – no worky
 - But interesting possible method of anti-debugging!
- What about windbg?
 - Let's test...

Debugger War

- So, it's obvious that windbg is more 'determined' than olly
- Has a way of suspending the threads so you can halt the process
- But...
 - ed esp 0...
 - die.

Lost Again

- Debug WriteProcessMemory
- Pretty much wraps NtWriteVirtualMemory
 - ..which gets an NT_STATUS of ACCESS_DENIED
- So, we're screwed
 - We pretty much can't touch the process memory

Windbg Wins Again

- !chkimg
 - “...detects corruption in the images of executable files by comparing them to the copy on a symbol store or other file repository”
 - Windbg was sent from God
 - And so was skywing :)

```
kd> !chkimg -ss .text -d nt
```

```
80501060-80501067 8 bytes - nt!KiServiceTable+30
```

```
[ 62 96 5c 80 12 96 5c 80:f8 58 7a 82 78 3c 7a 82 ]
```

```
80501074-80501077 4 bytes - nt!KiServiceTable+44 (+0x14)
```

OMGWTFBBQ PWN!

- !chkimg -f nt for the win!
- KiServiceTable means SSDT/syscall hooks
 - Why RK Hook Analyzer didn't see this, I have no idea

dd poi(KeServiceDescriptorTable) L11c

80501070 8060a212 nt!NtAllocateUids

80501074 82711ea8

80501078 805a44be nt!NtAreMappedFilesTheSame

- Compared to..

80501074 8059c8f4 nt!NtAllocateVirtualMemory

Now What?

- Write up a driver to do the unhooking
- I tried to be more intelligent this time and resolve the addresses without prior knowledge
 - But, `MmGetSystemRoutineAddress()` only finds addresses exported by `ntoskrnl`
 - `NtWriteVirtualMemory` isn't exported!
 - So, I just hardcoded :P

En La Moda Más Elegante

- Usermode section
- No need to disable the firewall when you can inject shellcode/DLLs into it
 - Hooks are gone – we can do what we want
- Insert a stager into NIS (it's trusted)
 - Allows for greater flexibility with metasploit – send down any payload
- That part is easy – what about getting a DLL to run without a connection to the attacker?

Phantom Loader

- Thanks to pusscat for the help! :D
- File format vulns can house VERY large payloads
 - ..enough to embed a DLL ;)
- Metasploit DLL injection stuff wants a remote connection to receive it
 - What if we can just load it locally?
- Egg hunting instead of recv()'ing
 - Search for the egg, load the dll at that address – requires minimal modification

End

- Thanks to the following people:
 - Pusscat, Rolf Rolles, Richard Johnson, Skape, and **especially** Skywing
- References:
- http://www.openrce.org/articles/full_view/24
 - Debugger internals for windows
- The DDK docs
- Wear your cyberterrorism turban with pride! @:)
 - Thanks for that, pancho :)