Hack Microsoft
by using Microsoft signed binaries

Pierre-Alexandre Braeken aka giMini
Cyber Security Architect
About Me

- I am from Belgium
- Senior security consultant / researcher @ Deloitte.
  - Incident Response, Compromise Assessment, Red Team
- 13+ years experience in information technology and security
- Starcraft 2 player
Why PowerMemory?

• I wanted to
  • Create a PowerShell tool using Microsoft signed binaries to play and hack Microsoft operating systems
  • Get a lot of money fun
1. What is PowerMemory?
2. Debug all the things
3. Let’s get technical
4. Weaponization: integrated to Empire
5. Mitigations
TOC

1. What is PowerMemory?
2. Debug all the things
3. Let’s get technical
4. Weaponization: integrated to Empire
5. Mitigations
What is PowerMemory
PowerMemory is a Minesweeper solver!
“That’s all Folks!”
Land

With **PowerShell** that is a **Microsoft tool** and a **Microsoft signed debugger**, PowerMemory can do what you want into the:

• User land
• Kernel land
• Wonderland
Send and receive TXT

PowerMemory send and receive text to and from the debugger.

And it is enough to do pretty much what you want.
PowerMemory is a user land attacker

- Get **Windows Passwords** from the memory
- **Inject and execute a shellcode** in a remote process
- Can **modify the memory** of a process (Minesweeper)
Kernel land is fun too...
PowerMemory is a kernel land attacker too (DKOM)

- Hide/Unhide a process
- Protect a process
- Inject all privileges in a process with SYSTEM identity
- Pass-The-Token attack

@giMini - PA Braeken
PowerMemory is an Active Directory Recon and Attack tool

- SPN scan (passive nmap)
- Get **GPP passwords** of all connected forests
- Assess **servers share** of all connected forests
  - Report place where the **authenticated** user can write
- Draw the AD topology with Visio and make a complete AD report
Elevate Your Rights, Bro!

- Auto escalation (Power-Escalate)
- Break and reveal passwords (Get-MacAfee)
- BSOD on vulnerable systems and get passwords from the dump
- ByPass UAC (Elevate-YourRightsMan)
- LOL : Check Point Software Firewall-1 3.0/1 4.0 - Session Agent Impersonation (Get-FirewallCredential)
Main Menu

Follow the white Rabbit :-)  
pbraeke@gmail.com

What do you want assess?
1) Reveal memory passwords
2) Local escalation attempt
3) Get McAfee passwords :-)
4) Active Directory assessment
5) Scan services network
6) Get all the Ticket (to be cracked with kerberoast)
7) Fun with Winmine
0) Exit

Enter menu number and press <ENTER>: |
1. What is PowerMemory?
2. Debug all the things
3. Let’s get technical
4. Weaponization: integrated to Empire
5. Mitigations
Debug all the things
Yeah Jeffrey, let’s automate the debugger!

AUTOMATE ALL THE THINGS!
Why using the Microsoft debugger?

• Because it’s a **Microsoft signed** application!
First steps
Symbols loading...

@giMini - PA Braeken
Get-FreeSymbols

• **Symbols are free!**
  [http://msdl.microsoft.com/download/symbols](http://msdl.microsoft.com/download/symbols)

• LIST_ENTRY which contains domain, user and password informations →
  • `L_LogSessList`

• Key (nt5) →
  • `g_pDesXKey` : DES-X key
  and `g_Feedback`

• Key (nt6,nt10) →
  • `h3DesKey` : Triple DES key
  • `AesKey` : AES key
  and `InitializationVector`  

@giMini - PA Braeken
1. What is PowerMemory?
2. Debug all the things
3. Let’s get technical
4. Weaponization: integrated to Empire
5. Mitigations
Let’s get technical
Passwords!

User land
The **Digest** Security Support Provider is one of the defaults component that interact with the Security Support Provider Interface Architecture (SSPI). As Microsoft tell to us, “Digest Authentication is an industry standard that, beginning with Windows 2000, is used for Lightweight Directory Access Protocol (LDAP) and web authentication. Digest Authentication transmits credentials across the network as an MD5 hash or message digest. Digest SSP (Wdigest.dll) is used for the following:

- **Internet Explorer (IE) and Internet Information Services (IIS) access**
- **LDAP queries**

  Location: `%windir%\Windows\System32\Digest.dll`


It is used everywhere for **Single-Sign-On** (SSO) in your corporate company.
Steal the bytes

• Dumping lsass (locally or remotely)
• Convert hiberfil.sys to dump file
• BSOD! and get the crash dump file
• Leverage the Hypervisor! (works for Hyper-V and VMWare)
• Access lsass process in kernel mode
Let’s manipulate the bytes

PowerMemory:
1. Calls the debugger and sends a command to execute
2. Retrieves the bytes
3. Parses them
4. Sends a new command with bytes to write at an address
Can you see the password?

- 0:000> dd 0252e020
- 00000000`0252e020 0252e4a0 00000000 fc7812c0 0000007fe
- 00000000`0252e030 00000001 00000000 0252e020 00000000
- 00000000`0252e040 91e505e3 00000000 00001001 0000000a
- 00000000`0252e050 000e000c 00000000 03350500 00000000
- 00000000`0252e060 00120010 00000000 03350b40 00000000
- 00000000`0252e070 00180014 00000000 033503c0 00000000
- 00000000`0252e080 00180016 00000000 03350c40 00000000
- 00000000`0252e090 00260024 00000000 025bfe00 00000000

@giMini - PA Braeken
Where is Wally?
Find Wally!

0:000> dd 0252e020
00000000`0252e020 0252e4a0 00000000 fc7812c0 000007fe
00000000`0252e030 00000001 00000000 0252e020 00000000
00000000`0252e040 91e505e3 00000000 00001001 0000000a
00000000`0252e050 000e000c 00000000 03350500 00000000
00000000`0252e060 00120010 00000000 03350c40 00000000
00000000`025e070 00180014 00000000 033503c0 00000000
00000000`025e080 00180016 00000000 03350b40 00000000
00000000`025e090 00260024 00000000 025bfe00 00000000

Next entry
Previous entry
This address
LUID address
Username address
Netbios domain name address
Encrypted Password address
Domain name address
Username@domain address
MaxLength
MinLength

@giMini - PA Braeken
Key is 0x18 bytes:
bd00c989 2a089930 919bc481 722179b2 016a665d
424f0046

Key transformed little-endian with db command
89 c9 00 bd 30 99 08 2a 81 c4 9b 91-b2 79 21 72 5d 66 6a 01 46 00 4f 42
And finally

- 0:000> db Isasrv!InitializationVector
- 000007fe`fcf9e7f0  f0 dd 9a c5 1d c3 ed 92-d9 3e cc fa d0 c5 b7 c1 ..........>......
- 000007fe`fcf9e800  10 31 3e 00 00 00 00 00-ff ff ff ff 00 00 00 00  .1>................
- 000007fe`fcf9e810  00 00 00 00 00 00 00 00-0c 10 00 00 00 00 00 00 .................
- 000007fe`fcf9e820  00 00 00 00 00 00 00 00-d0 b0 ea fc fe 07 00 00 ................
- 000007fe`fcf9e830  f0 a1 f0 fc fe 07 00 00-d0 b0 ea fc fe 07 00 00 ................
- 000007fe`fcf9e840  c0 08 e7 fc fe 07 00 00-d0 b0 ea fc fe 07 00 00 ................
- 000007fe`fcf9e850  f0 a1 f0 fc fe 07 00 00-d0 b0 ea fc fe 07 00 00 ................
- 000007fe`fcf9e860  c0 03 e7 fc fe 07 00 00-80 04 e7 fc fe 07 00 00 ................
Demo!
Inject a shellcode in a remote process and execute it

User land
We need information

• A memory **executable** zone
• A **null padding zone** in the memory executable zone to inject our shellcode in
• The **address of the null padding zone** where we injected our shellcode
How to get the information?

We need to parse the PE executable loaded in memory

• The **address** of the **module loaded** to inject

• From the module address, the **PE Header address** (we found in the MS-DOS header) which is at [(module loaded address)+3C] address

• From the PE Header address which is 24 bytes, the **size** of the **optional header**, in bytes

• From the Optional Header, the **Section Table structure** which follows immediately the Optional Header

• From the section table,
  • The **virtual size**
  • The **virtual address**
  • The **raw data pointer**

Then "r @rip=0x$moduleAddress"
Demo!
Kernel stuff

Kernel land
Hide a process by manipulating the bytes (not API) with PowerShell and a Microsoft debugger

```
"f $thisProcessLinks ('$BLINK.Substring(17, 2)') $thisProcessLinks ('$BLINK.Substring(15, 2)') $thisProcessLinks ('$BLINK.Substring(13, 2)') $thisProcessLinks ('$BLINK.Substring(11, 2))"
```

```
"f $thisProcessLinks ('$BLINK.Substring(17, 2)') $thisProcessLinks ('$BLINK.Substring(15, 2)') $thisProcessLinks ('$BLINK.Substring(13, 2)') $thisProcessLinks ('$BLINK.Substring(11, 2))"
```
Demo!
1. What is PowerMemory?
2. Debug all the things
3. Let’s get technical
4. Weaponization: integrated to Empire
5. Mitigations
Weaponization: integrated to Empire

In real world
“Empire is a pure built on cryptologically-secure communications and a flexible architecture. Empire implements the ability to run PowerShell agents without needing powershell.exe, rapidly deployable post-exploitation modules ranging from key loggers to Mimikatz, and adaptable communications to evade network detection, all wrapped up in a usability-focused framework.”
Load PowerMemory into memory

1. Target a machine
2. Force the machine to load the Empire agent
3. Through the Empire agent, load PowerMemory into the target machine memory
4. Make fun and profits
Demo!
1. What is PowerMemory?
2. Debug all the things
3. Let’s get technical
4. Weaponization: integrated to Empire
5. Mitigation
Mitigation
Mitigate attacks

1. Don’t trust trusted tools. Look at their behavior.
2. Look for dumping activities.
3. Look for `bcdedit.exe` uses (someone successfully launched it with `/debug` on, you should know, control and prevent when suspicious).
4. Don’t trust the endpoint.
5. Look for behavior.
Summary

1. PowerMemory leverages PowerShell and a signed Microsoft debugger to hack Microsoft operating system
2. It’s difficult to detect because it just sends and receives text messages and bytes
3. PowerMemory helps to recon and attack Active Directory domain
4. PowerMemory can help you to elevate your rights with Power-Escalate
Thank you!

Questions?