National Cyber Security Centre
a part of GCHQ

Advisory: APT29 targets COVID-19 vaccine development

Version 1.0

16 July 2020

About this document

This report details recent Tactics, Techniques and Procedures (TTPs) of the group commonly known as ‘APT29’, also known as ‘the Dukes’ or ‘Cozy Bear’.

This report provides indicators of compromise as well as detection and mitigation advice.

Disclaimer

Introduction

The United Kingdom's National Cyber Security Centre (NCSC) and Canada's Communications Security Establishment (CSE) assess that APT29 (also known as 'the Dukes' or 'Cozy Bear') is a cyber espionage group, almost certainly part of the Russian intelligence services. The United States' National Security Agency (NSA) agrees with this attribution and the details provided in this report.

The United States' Department of Homeland Security's Cybersecurity and Infrastructure Security Agency (DHS CISA) endorses the technical detail and mitigation advice provided in this advisory.

The group uses a variety of tools and techniques to predominantly target governmental, diplomatic, think-tank, healthcare and energy targets for intelligence gain.

Throughout 2020, APT29 has targeted various organisations involved in COVID-19 vaccine development in Canada, the United States and the United Kingdom, highly likely with the intention of stealing information and intellectual property relating to the development and testing of COVID-19 vaccines.

APT29 is using custom malware known as 'WellMess' and 'WellMail' to target a number of organisations globally. This includes those organisations involved with COVID-19 vaccine development. WellMess and WellMail have not previously been publicly associated to APT29.

Details of techniques

Initial infection vectors

The group frequently uses publicly available exploits to conduct widespread scanning and exploitation against vulnerable systems, likely in an effort to obtain authentication credentials to allow further access. This broad targeting potentially gives the group access to a large number of systems globally, many of which are unlikely to be of immediate intelligence value. The group may maintain a store of stolen credentials in order to access these systems in the event that they become more relevant to their requirements in the future.

In recent attacks targeting COVID-19 vaccine research and development, the group conducted basic vulnerability scanning against specific external IP addresses owned by the organisations. The group then deployed public exploits against the vulnerable services identified.

The group has been successful using recently published exploits to gain initial footholds. Examples include, but are not limited to:

- CVE-2019-19781 Citrix [1]
- CVE-2019-11510 Pulse Secure [2]
- CVE-2018-13379 FortiGate [2]
- CVE-2019-9670 Zimbra [3]

The group likely seeks to take full advantage of a variety of new exploits when publicised. More information about these exploits can be found in previous NCSC advisories on Citrix and VPN vulnerabilities [1,2].

The group also uses spear-phishing to obtain authentication credentials to internet-accessible login pages for target organisations.

Persistent access

Upon gaining access to a system, the group likely drops further tooling and/or seeks to obtain legitimate credentials to the compromised systems in order to maintain persistent access. The actor is likely to use anonymising services when using the stolen credentials.

WellMess malware

In some cases, APT29 also deploys custom malware known as WellMess or WellMail to conduct further operations on the victim's system.

WellMess is malware written in either Golang or .NET and has been in use since at least 2018. WellMess was first reported on by JPCERT and LAC researchers in July 2018[4][5]. It is named after one of the function names in the malware - 'wellmess'. WellMess is a lightweight malware designed to execute arbitrary shell commands, upload and download files. The malware supports HTTP, TLS and DNS communications methods.

Indicators of compromise (IOCs) for WellMess are available in the appendix.

WellMail malware

WellMail is a lightweight tool designed to run commands or scripts with the results being sent to a hardcoded Command and Control (C2) server.

The NCSC has named this malware 'WellMail' due to file paths containing the word 'mail' and the use of server port 25 present in the sample analysed. Similar to WellMess, WellMail uses hard-coded client and certificate authority TLS certificates to communicate with C2 servers.

The binary is an ELF utility written in Golang which receives a command or script to be run through the Linux shell. To our knowledge, WellMail has not been previously named in the public domain.

IOCs for WellMail are available in the appendix.

Certificate usage

WellMess and WellMail samples contained TLS certificates with the hard-coded subjectKeyIdentifier (SKI) '0102030406', and used the subjects 'C=Tunis, O=IT' and 'O=GMO GlobalSign, Inc' respectively. These certificates can be used to identify further malware samples and infrastructure. Servers with this GlobalSign certificate subject may be used for other functions in addition to WellMail malware communications.

SoreFang malware

Malware, dubbed ‘SoreFang’ by the NCSC, is a first stage downloader that uses HTTP to exfiltrate victim information and download second stage malware. The sample analysed by the NCSC contains the same infrastructure as a WellMess sample (103.216.221[.]19).

It is likely that SoreFang targets SangFor devices. Industry reporting indicates that other actors, reportedly including ‘DarkHotel’, have also targeted SangFor devices. Therefore, not all SangFor exploitation activity relates to targeting by APT29.

Conclusion

APT29 is likely to continue to target organisations involved in COVID-19 vaccine research and development, as they seek to answer additional intelligence questions relating to the pandemic.

It is strongly recommended that organisations use the rules and IOCs in the appendix in order to detect the activity detailed in this advisory.

Appendix

Indicators of compromise and detection rules

WellMess IOCs

Hashes
00654dd07721e7551641f90cba832e98c0acb030e2848e5efc0e1752c067ec07
0322c4c2d511f73ab55bf3f43b1b0f152188d7146cc67ff497ad275d9dd1c20f
03e9adae529155961f1f18212ff70181bde0e3da3d7f22961a6e2b1c9da2dd2e
0b8e6a11adaa3df120ec15846bb966d674724b6b92eae34d63b665e0698e0193
14e9b5e214572cb13ff87727d680633f5ee238259043357c94302654c546cad2
1fed2e1b077af08e73fb5ecffd2e5169d5289a825dcaf2d8742bb8030e487641
21129ad17800b11cdb36906ba7f6105e3bd1cf44575f77df58ba91640ba0cab9
2285a264ffab59ab5a1eb4e2b9bcab9baf26750b6c551ee3094af56a4442ac41
2daba469f50cd1b77481e605aeae0f28bf14cedfcd8e4369193e5e04c523bc38
49bfff6b91ee71bbf8fd94829391a36b844ffba104c145e01c92732ada52c8ba
4c8671411da91eb5967f408c2a6ff6baf25ff7c40c65ff45ee33b352a711bf9c
5ca4a9f6553fea64ad2c724bf71d0fac2b372f9e7ce2200814c98aac647172fb
797159c202ca41356bee18c5303d37e9d2a43ca43d0ce02e1fd9e7045b925d11
7c39841ba409bce4c2c35437ecf043f22910984325c70b9530edf15d826147ee
84b846a42d94431520d3d2d14262f3d3a5d96762e56b0ae471b853d1603ca403
8749c1495af4fd73ccfc84b32f56f5e78549d81feefb0c1d1c3475a74345f6a8
92a856a2216e107496ee086e1c8cfe14e15145e7a247539815fd37e5a18b84d9
93e9383ae8ad2371d457fc4c1035157d887a84bbfe66fbbb3769c5637de59c75
953b5fc9977e2d50f3f72c6ce85e89428937117830c0ed67d468e2d93aa7ec9a
a03a71765b1b0ea7de4fbcb557dcfa995ff9068e92db9b2dada9dd0841203145
a117b2a904c24df62581500176183fbc282a740e4f11976cdfc01fe664a02292
a3ca47e1083b93ea90ace1ca30d9ef71163e8a95ee00500cbd3fd021da0c18af
b75a5be703d9ba3721d046db80f62886e10009b455fa5cdfd73ce78f9f53ec5a
bec1981e422c1e01c14511d384a33c9bcc66456c1274bbbac073da825a3f537d
c1a0b73bad4ca30a5c18db56c1cba4f5db75f3d53daf62ddc598aae2933345f3
d7e7182f498440945fc8351f0e82ad2d5844530ebdba39051d2205b730400381
dd3da0c596fd699900cdd103f097fe6614ac69787edfa6fa84a8f471ecb836bb
e329607379a01483fc914a47c0062d5a3a8d8d65f777fbad2c5a841a90a0af09
e3d6057b4c2a7d8fa7250f0781ea6dab4a977551c13fe2f0a86f3519b2aaee7a
f3af394d9c3f68dff50b467340ca59a11a14a3d56361e6cffd1cf2312a7028ad
f622d031207d22c633ccec187a24c50980243cb4717d21fad6588dacbf9c29e9
fd3969d32398bbe3709e9da5f8326935dde664bbc36753bd41a0b111712c0950

IP Addresses
103.103.128[.]221
103.13.240[.]46
103.205.8[.]72
103.216.221[.]19
103.253.41[.]102
103.253.41[.]68
103.253.41[.]82
103.253.41[.]90
103.73.188[.]101
111.90.146[.]143
111.90.150[.]176
119.160.234[.]163
119.160.234[.]194
119.81.173[.]130
119.81.178[.]105
120.53.12[.]132

122.114.197[.]185
122.114.226[.]172
141.255.164[.]29
141.98.212[.]55
145.249.107[.]73
146.0.76[.]37
149.202.12[.]210
169.239.128[.]110
176.119.29[.]37
178.211.39[.]6
185.120.77[.]166
185.145.128[.]35
185.99.133[.]112
191.101.180[.]78
192.48.88[.]107
193.182.144[.]105
202.59.9[.]59
209.58.186[.]196
209.58.186[.]197
209.58.186[.]240
220.158.216[.]130
27.102.130[.]115
31.170.107[.]186
31.7.63[.]141
45.120.156[.]69
45.123.190[.]167
45.123.190[.]168
45.152.84[.]57
46.19.143[.]69
5.199.174[.]164
66.70.247[.]215
79.141.168[.]109
81.17.17[.]213
85.93.2[.]116

Yara Rules

```
rule wellmess_dotnet_unique_strings {
  meta:
    description = "Rule to detect WellMess .NET samples based on unique
strings and function/variable names"
    author = "NCSC"
    hash =
"2285a264ffab59ab5a1eb4e2b9bcab9baf26750b6c551ee3094af56a4442ac41"
  strings:
    $s1 = "MaxPostSize" wide
    $s2 = "HealthInterval" wide
    $s3 = "Hello from Proxy" wide
    $s4 = "Start bot:" wide
    $s5 = "Choise" ascii wide
    $s7 = "FromNormalToBase64" ascii
    $s8 = "FromBase64ToNormal" ascii
    $s9 = "ConvBytesToWords" ascii
    $s10 = "WellMess" ascii
    $s11 = "chunksM" ascii
  condition:
    uint16(0) == 0x5a4d and uint16(uint16(0x3c)) == 0x4550 and 3 of them
}
```

```
rule wellmess_botlib_function_names {
  meta:
    description = "Rule to detect WellMess Golang samples based on the
function names used by the actor"
    author = "NCSC"
    hash =
"8749c1495af4fd73ccfc84b32f56f5e78549d81feefb0c1d1c3475a74345f6a8"
  strings:
    $s1  = "botlib.wellMess" ascii wide
    $s2  = "botlib.saveFile" ascii wide
    $s3  = "botlib.reply" ascii wide
    $s4  = "botlib.init" ascii wide
    $s5  = "botlib.generateRandomString" ascii wide
    $s6  = "botlib.encrypt" ascii wide
    $s7  = "botlib.deleteFile" ascii wide
    $s8  = "botlib.convertFromString" ascii wide
    $s9  = "botlib.chunksM" ascii wide
    $s10 = "botlib.Work" ascii wide
    $s11 = "botlib.UnpackB" ascii wide
    $s12 = "botlib.Unpack" ascii wide
    $s13 = "botlib.UDFile" ascii wide
    $s14 = "botlib.Split" ascii wide
    $s15 = "botlib.Service" ascii wide
    $s16 = "botlib.SendMessage" ascii wide
    $s17 = "botlib.Send.func1" ascii wide
    $s18 = "botlib.Send" ascii wide
    $s19 = "botlib.ReceiveMessage" ascii wide
    $s20 = "botlib.RandStringBytes" ascii wide
    $s21 = "botlib.RandInt" ascii wide
    $s22 = "botlib.Post" ascii wide
    $s23 = "botlib.Parse" ascii wide
    $s24 = "botlib.Pad" ascii wide
    $s25 = "botlib.Pack" ascii wide
    $s26 = "botlib.New" ascii wide
    $s27 = "botlib.KeySizeError.Error" ascii wide
    $s28 = "botlib.Key" ascii wide
    $s29 = "botlib.Join" ascii wide
    $s30 = "botlib.GetRandomBytes" ascii wide
    $s31 = "botlib.GenerateSymmKey" ascii wide
    $s32 = "botlib.FromNormalToBase64" ascii wide
    $s33 = "botlib.EncryptText" ascii wide
    $s34 = "botlib.Download" ascii wide
    $s35 = "botlib.Decipher" ascii wide
    $s36 = "botlib.Command" ascii wide
    $s37 = "botlib.Cipher" ascii wide
    $s38 = "botlib.CalculateMD5Hash" ascii wide
    $s39 = "botlib.Base64ToNormal" ascii wide
    $s40 = "botlib.AES_Encrypt" ascii wide
    $s41 = "botlib.AES_Decrypt" ascii wide
    $s42 = "botlib.(*rc6cipher).Encrypt" ascii wide
    $s43 = "botlib.(*rc6cipher).Decrypt" ascii wide
    $s44 = "botlib.(*rc6cipher).BlockSize" ascii wide
    $s45 = "botlib.(*KeySizeError).Error" ascii wide
    $s46 = "botlib.DownloadDNS" ascii wide
    $s47 = "botlib.JoinDnsChunks" ascii wide
    $s48 = "botlib.SendDNS" ascii wide
    $s49 = "botlib.CreateDNSName" ascii wide
  condition:
    ((uint16(0) == 0x5a4d and uint16(uint16(0x3c)) == 0x4550) or
uint32(0) == 0x464c457f) and any of them
}
```

```
rule wellmess_certificate_base64_snippets {
  meta:
    description = "Rule for detection of WellMess based on base64
snippets of certificates used"
    author = "NCSC"
    hash =
"8749c1495af4fd73ccfc84b32f56f5e78549d81feefb0c1d1c3475a74345f6a8"
  strings:
    $a1 = "BgNVHQ4EBwQFAQIDBA"
    $a2 = "YDVR0OBAcEBQECAwQG"
    $a3 = "GA1UdDgQHBAUBAgMEB"
    $b1 = "BgNVBAYTBVR1bmlzMQswCQYDVQQKEwJJVD"
    $b2 = "YDVQQGEwVUdW5pczELMAkGA1UEChMCSVQx"
    $b3 = "GA1UEBhMFVHVuaXMxCzAJBgNVBAoTAklUM"
  condition:
    ((uint16(0) == 0x5a4d and uint16(uint16(0x3c)) == 0x4550) or
uint32(0) == 0x464c457f) and any of ($a*) and any of ($b*)
}
```

```
rule wellmess_regex_used_for_parsing_beacons {
  meta:
    description = "Detects WellMess Golang and .NET samples based on the
regex they used to parse commands and beacon information"
    author = "NCSC"
    hash =
"8749c1495af4fd73ccfc84b32f56f5e78549d81feefb0c1d1c3475a74345f6a8"
  strings:
    $a = "fileName:(?<fn>.*?)\\sargs:(?<arg>.*)\\snotwait:(?<nw>.*)"
ascii wide
    $b = "<;(?<key>[^;]*?);>(?<value>[^<]*?)<;[^;]*?;>" ascii wide
  condition:
    ((uint16(0) == 0x5a4d and uint16(uint16(0x3c)) == 0x4550) or
uint32(0) == 0x464c457f) and any of them
}
```

WellMail IOCs

Hashes
83014ab5b3f63b0253cdab6d715f5988ac9014570fa4ab2b267c7cf9ba237d18 (UPX)
0c5ad1e8fe43583e279201cdb1046aea742bae59685e6da24e963a41df987494 (Unpacked)

IP Addresses (Malware)
103.216.221[.]19

IP Addresses (‘GlobalSign’ certificate, operated by APT29 but not necessarily used for WellMail malware communications)
119.81.184[.]11
185.225.226[.]16
188.241.68[.]137
45.129.229[.]48

Yara Rules

```
rule wellmail_unique_strings {
  meta:
    description = "Rule for detection of WellMail based on unique strings
contained in the binary"
    author = "NCSC"
    hash =
"0c5ad1e8fe43583e279201cdb1046aea742bae59685e6da24e963a41df987494"
  strings:
    $a = "C:\\Server\\Mail\\App_Data\\Temp\\agent.sh\\src"
    $b = "C:/Server/Mail/App_Data/Temp/agent.sh/src/main.go"
    $c = "HgQdbx4qRNv"
    $d = "042a51567eea19d5aca71050b4535d33d2ed43ba"
    $e = "main.zipit"
    $f = "@[^\\s]+?\\s(?P<tar>.*?)\\s"
  condition:
    uint32(0) == 0x464C457F and 3 of them
}
```

```
rule wellmail_certificate_base64_snippets {
  meta:
    description = "Rule for detection of WellMail based on base64
snippets of certificates used"
    author = "NCSC"
    hash =
"0c5ad1e8fe43583e279201cdb1046aea742bae59685e6da24e963a41df987494"
  strings:
    $a1 = "BgNVHQ4EBwQFAQIDBA"
    $a2 = "YDVR0OBAcEBQECAwQG"
    $a3 = "GA1UdDgQHBAUBAgMEB"
    $b1 = "BgNVBAoTE0dNTyBHbG9iYWxTaWduLCBJbm"
    $b2 = "YDVQQKExNHTU8gR2xvYmFsU2lnbiwgSW5j"
    $b3 = "GA1UEChMTR01PIEdsb2JhbFNpZ24sIEluY"
  condition:
    uint32(0) == 0x464C457F and any of ($a*) and any of ($b*)
}
```

SoreFang IOCs

Hashes
58d8e65976b53b77645c248bfa18c3b87a6ecfb02f306fe6ba4944db96a5ede2
65495d173e305625696051944a36a031ea94bb3a4f13034d8be740982bc4ab75
a4b790ddffb3d2e6691dcacae08fb0bfa1ae56b6c73d70688b097ffa831af064

IP Addresses (Malware)
103.216.221[.]19

Yara Rules

```
rule sorefang_directory_enumeration_output_strings {
  meta:
    description = "Rule to detect SoreFang based on formatted string
output for directory enumeration"
    author = "NCSC"
    hash =
"58d8e65976b53b77645c248bfa18c3b87a6ecfb02f306fe6ba4944db96a5ede2"
```

```
  strings:
    $ = "----------All usres directory----------"
    $ = "----------Desktop directory----------"
    $ = "----------Documents directory----------"
  condition:
    (uint16(0) == 0x5A4D and uint16(uint32(0x3c)) == 0x4550) and any of
them
}
```

```
rule sorefang_encryption_key_2b62 {
  meta:
    description = "Rule to detect SoreFang based on hardcoded encryption
key"
    author = "NCSC"
    hash =
"58d8e65976b53b77645c248bfa18c3b87a6ecfb02f306fe6ba4944db96a5ede2"
  strings:
    $ = "2b6233eb3e872ff78988f4a8f3f6a3ba"
  condition:
    (uint16(0) == 0x5A4D and uint16(uint32(0x3c)) == 0x4550) and any of
them
}
```

```
rule sorefang_encryption_key_schedule {
  meta:
    description = "Rule to detect SoreFang based on the key schedule used
for encryption"
    author = "NCSC"
    hash =
"58d8e65976b53b77645c248bfa18c3b87a6ecfb02f306fe6ba4944db96a5ede2"
  strings:
      $ = {C7 05 ?? ?? ?? ?? 63 51 E1 B7 B8 ?? ?? ?? ?? 8B 48 FC 81 E9 47
86 C8 61 89 08 83 C0 04 3D ?? ?? ?? ?? 7E EB 33 D2 33 C9 B8 2C 00 00 00
89 55 D4 33 F6 89 4D D8 33 DB 3B F8 0F 4F C7 8D 04 40 89 45 D0 83 F8 01
7C 4F 0F 1F 80 00 00 00 00}
  condition:
     (uint16(0) == 0x5A4D and uint16(uint32(0x3c)) == 0x4550) and any of
them
}
```

```
rule sorefang_command_elem_cookie_ga_boundary_string {
  meta:
    description = "Rule to detect SoreFang based on scheduled task
element and Cookie header/boundary strings"
    author = "NCSC"
    hash =
"58d8e65976b53b77645c248bfa18c3b87a6ecfb02f306fe6ba4944db96a5ede2"
  strings:
    $ = "<Command>" wide
    $ = "Cookie:_ga="
    $ = "------974767299852498929531610575"
  condition:
    (uint16(0) == 0x5A4D and uint16(uint32(0x3c)) == 0x4550) and 2 of
them
}
```

```
rule sorefang_encryption_round_function {
  meta:
    description = "Rule to detect SoreFang based on the encryption round
function"
    author = "NCSC"
    hash =
"58d8e65976b53b77645c248bfa18c3b87a6ecfb02f306fe6ba4944db96a5ede2"
  strings:
    $ = {8A E9 8A FB 8A 5D 0F 02 C9 88 45 0F FE C1 0F BE C5 88 6D F3 8D
14 45 01 00 00 00 0F AF D0 0F BE C5 0F BE C9 0F AF C8 C1 FA 1B C0 E1 05
0A D1 8B 4D EC 0F BE C1 89 55 E4 8D 14 45 01 00 00 00 0F AF D0 8B C1}
  condition:
    (uint16(0) == 0x5A4D and uint16(uint32(0x3c)) == 0x4550) and any of
them
}
```

```
rule sorefang_add_random_commas_spaces {
  meta:
    description = "Rule to detect SoreFang based on function that adds
commas and spaces"
    author = "NCSC"
    hash =
"58d8e65976b53b77645c248bfa18c3b87a6ecfb02f306fe6ba4944db96a5ede2"
  strings:
    $ = {E8 ?? ?? ?? ?? B9 06 00 00 00 99 F7 F9 8B CE 83 FA 04 7E 09 6A
02 68 ?? ?? ?? ?? EB 07 6A 01 68 ?? ?? ?? ??}
  condition:
    (uint16(0) == 0x5A4D and uint16(uint32(0x3c)) == 0x4550) and any of
them
}
```

```
rule sorefang_modify_alphabet_custom_encode {
  meta:
    description = "Rule to detect SoreFang based on arguments passed into
custom encoding algorithm function"
    author = "NCSC"
    hash =
"58d8e65976b53b77645c248bfa18c3b87a6ecfb02f306fe6ba4944db96a5ede2"
  strings:
    $ = {33 C0 8B CE 6A 36 6A 71 66 89 46 60 88 46 62 89 46 68 66 89 46
64}
  condition:
     (uint16(0) == 0x5A4D and uint16(uint32(0x3c)) == 0x4550) and any of
them
}
```

```
rule sorefang_custom_encode_decode {
  meta:
    description = "Rule to detect SoreFang based on the custom
encoding/decoding algorithm function"
    author = "NCSC"
    hash =
"58d8e65976b53b77645c248bfa18c3b87a6ecfb02f306fe6ba4944db96a5ede2"
  strings:
    $ = {55 8B EC 8B D1 53 56 8B 75 08 8B DE 80 42 62 FA 8A 4A 62 66 D3
EB 57 3A 5A 5C 74 0F}
```

```
    $ = {3A 5A 5D 74 0A 3A 5A 58 74 05 3A 5A 59 75 05 FE C1 88 4A 62 8A
4A 62 B8 01 00 00 00}
    $ = {8A 46 62 84 C0 74 3E 3C 06 73 12 0F B6 C0 B9 06 00 00 00 2B C8
C6 46 62 06 66 D3 66 60 0F B7 4E 60}
    $ = {80 3C 38 0D 0F 84 93 01 00 00 C6 42 62 06 8B 56 14 83 FA 10 72
04 8B 06}
    $ = {0F BE 0C 38 8B 45 EC 0F B6 40 5B 3B C8 75 07 8B 55 EC B3 3E}
    $ = {0F BE 0C 38 8B 45 EC 0F B6 40 5E 3B C8 75 0B 8B 55 EC D0 EB C6
42 62 05}
    $ = {8B 55 EC 0F BE 04 38 0F B6 DB 0F B6 4A 5F 3B C1 B8 3F 00 00 00
0F 44 D8}
    $ = {8A 4A 62 66 8B 52 60 66 D3 E2 0F B6 C3 66 0B D0 8B 45 EC 66 89
50 60 8A 45 F3 02 C1 88 45 F3 3C 08 72 2E 04 F8 8A C8 88 45 F3 66 D3 EA
8B 4D 08 0F B6 C2 50}
    $ = {3A 5A 5C 74 0F 3A 5A 5D 74 0A 3A 5A 58 74 05 3A 5A 59 75 05 FE
C1 88 4A 62}

  condition:
     (uint16(0) == 0x5A4D and uint16(uint32(0x3c)) == 0x4550) and any of
them
}
```

```
rule sorefang_remove_chars_comma_space_dot {
  meta:
    description = "Rule to detect SoreFang based on function that removes
commas, spaces and dots"
    author = "NCSC"
    hash =
"58d8e65976b53b77645c248bfa18c3b87a6ecfb02f306fe6ba4944db96a5ede2"
  strings:
    $ = {8A 18 80 FB 2C 74 03 88 19 41 42 40 3B D6 75 F0 8B 5D 08}
    $ = {8A 18 80 FB 2E 74 03 88 19 41 42 40 3B D6 75 F0 8B 5D 08}
    $ = {8A 18 80 FB 20 74 03 88 19 41 42 40 3B D6 75 F0 8B 5D 08}
  condition:
        (uint16(0) == 0x5A4D and uint16(uint32(0x3c)) == 0x4550) and all
of them
}
```

```
rule sorefang_disk_enumeration_strings {
  meta:
    description = "Rule to detect SoreFang based on disk enumeration
strings"
    author = "NCSC"
    hash =
"a4b790ddffb3d2e6691dcacae08fb0bfa1ae56b6c73d70688b097ffa831af064"
strings:
    $ = "\x0D\x0AFree on disk: "
    $ = "Total disk: "
    $ = "Error in GetDiskFreeSpaceEx\x0D\x0A"
    $ = "\x0D\x0AVolume label: "
    $ = "Serial number: "
    $ = "File system: "
    $ = "Error in GetVolumeInformation\x0D\x0A"
    $ = "I can not het information about this disk\x0D\x0A"
  condition:
     (uint16(0) == 0x5A4D and uint16(uint32(0x3c)) == 0x4550) and all of
them
}
```

External Reading

[1] https://www.ncsc.gov.uk/news/citrix-alert
[2] https://www.ncsc.gov.uk/news/alert-vpn-vulnerabilities
[3] https://nvd.nist.gov/vuln/detail/CVE-2019-9670
[4] https://blogs.jpcert.or.jp/en/2018/07/malware-wellmes-9b78.html
[5] https://www.botconf.eu/wp-content/uploads/2018/12/2018-Y-Ishikawa-S-Nagano-Lets-go-with-a-Go-RAT-_final.pdf

Mitigation

A variety of mitigations will be of use in defending against the campaigns detailed in this report:

- **Protect your devices and networks by keeping them up to date**: use the latest supported versions, apply security patches promptly, use anti-virus and scan regularly to guard against known malware threats. See NCSC Guidance: https://www.ncsc.gov.uk/guidance/mitigating-malware.
- **Use multi-factor authentication (/2-factor authentication/two-step authentication) to reduce the impact of password compromises**. See NCSC Guidance: https://www.ncsc.gov.uk/guidance/multi-factor-authentication-online-services and https://www.ncsc.gov.uk/guidance/setting-two-factor-authentication-2fa
- **Treat people as your first line of defence**. Tell staff how to report suspected phishing emails, and ensure they feel confident to do so. Investigate their reports promptly and thoroughly. Never punish users for clicking phishing links or opening attachments. See NCSC Guidance: https://www.ncsc.gov.uk/phishing
- **Set up a security monitoring capability** so you are collecting the data that will be needed to analyse network intrusions. See NCSC Guidance: https://www.ncsc.gov.uk/guidance/introduction-logging-security-purposes.
- **Prevent and detect lateral movement in your organisation's networks**. See NCSC Guidance: https://www.ncsc.gov.uk/guidance/preventing-lateral-movement