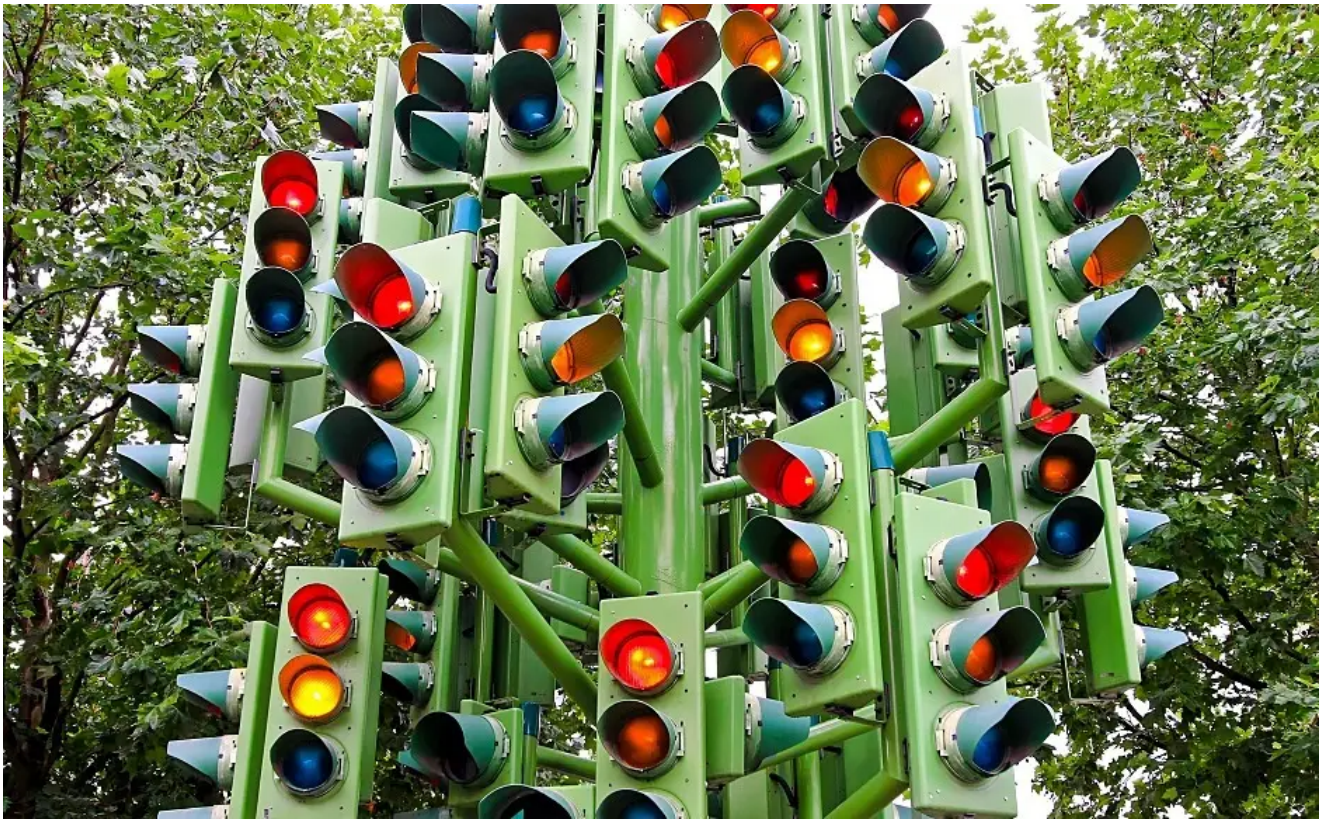
GootKit Developers Dress it Up With Web Traffic Proxy

securityintelligence.com/gootkit-developers-dress-it-up-with-web-traffic-proxy/

March 1, 2017



Banking & Finance March 1, 2017

By Gadi Ostrovsky co-authored by Limor Kessem 6 min read

Discovered in summer of 2014, GootKit is widely considered one of the most sophisticated banking Trojans active in the wild. The malware is being used in online banking fraud attacks on consumer and business accounts, mostly in the U.K. and other parts of Europe.

In this blog post, I will describe my analysis of a recent GootKit sample (MD5: 60e079ec28d47ef85e93039c21afd19c) discovered by IBM X-Force research in January 2017. The sample caught our attention when we realized that GootKit's developers had modified its architecture and changed the way it operates on the infected endpoints.

My research into GootKit's inner workings unveiled its new network interception method, which now proxies internet traffic through the malware instead of placing hooks on the browser. GootKit also bypasses certificate validation by hooking other relevant APIs to continue its malicious operation unhindered.

GootKit Renovating the Architecture

The first and most significant change I noticed in recent GootKit samples is an architectural expansion. In the past, GootKit operated on two principal modules:

1. The Loader module, which was responsible for persistence, malware updates and the injection of malicious code into the web browser and Windows OS processes; and
2. The Main module, which was responsible for general malware functionality. This module is based on a node.js engine, bound with the malware's code.

The Main module can be injected into a svchost process, at which point it acts as a master. It can also be injected into a browser process and act as a slave, intercepting all network communications via the browser by hooking the NtDeviceIoControlFile API. The NtDeviceIoControlFile service is a device-dependent interface that extends the control that applications have over various devices within the system. This API provides a consistent view of the input and output data to the system while still providing the application and the driver a device-dependent method of specifying a communications interface.

That two-module setup has changed. Recent samples we collected in 2017 feature three modules instead of two:

1. The Loader module, which has the same purpose;
2. The Main module, named node32.dll by GootKit's developers, which handles the malware's functionality, is based on the same node.js engine and implements the overall functionality in JavaScript code; and
3. The Browser injection module, a recent addition to GootKit's modular architecture that is responsible for network interception and access to the browser.

GootKit now uses this module to redirect internet traffic through its own Main module instead of letting it flow directly from the browser.

A Modified Deployment Flow

GootKit changes the way it deploys on infected endpoints and the inner workings of its operations. In the sample I analyzed for this post, GootKit is fetched by a dropper. As the droppee, it is run from an already-established persistence mechanism that was set up by its loader.

GootKit's recently updated persistence is achieved through a Group Policy Object (GPO) entry in the Windows Registry. Group Policies control the user's working environment, and GootKit abuses this feature to launch whenever the computer boots up. Most financial malware families set a RunKey in the Registry, but GootKit has changed that scheme, likely to avoid automated detection rules.

Next, GootKit creates a mstsc.exe process in suspended state and injects the Loader module into it. This process, also known as Remote Desktop Connection (RDC), is the client application for remote desktop services. It allows a user to remotely log into a networked computer running the terminal services server.

The mstsc process, in turn, runs a svchost.exe process and injects the Loader module into svchost. Next, the Main module is loaded from the svchost process.

The malware's binaries are stored in the following Windows registry keys: HKCU\Software\AppDataLow\binary**Image32_0** to HKCU\Software\AppDataLow\binary**Image32_7**.

GootKit as Its Own Internet Traffic Proxy

Upon launching the malware, the Main module begins to set up a proxy server on the infected machines, utilizing GootKit's new browser injection module.

Two functions implemented by GootKit within its node.js engine are being used here: SpInitializeStandaloneMitm and RandomListenPortBase. Using a malicious JavaScript function, SpInitializeStandaloneMitm, the malware creates a proxy for both HTTP and HTTPS traffic on the infected endpoint.

GootKit uses the RandomListenPortBase function to specify the standard port choice for its traffic. HTTP traffic will route via port 80 and HTTPS via port 443. As for the listening port, which is typically randomly selected in the 1024 to 5000 range, GootKit's hardcoded choice is port 6000.

Figure 1: JavaScript code used for GootKit's proxy server setup

Note that the next function called is Initialize Certificate Faker.

Overcoming the Browser's Certificate Verification Flags

Upon the creation of a new browser process on the infected endpoint, the Loader module injects the Browser module into that process. The Browser module will, in turn, hook two types of APIs.

1. Network APIs responsible for connection creations:
    1. mswsock.dllMSAFD_ConnectEx API (the mswsock.dll is a module providing extensions for Winsock; services provided by this file are not part of Winsock); and
    2. mswsock.dll WSPConnect API.
2. Certificate validation APIs:
    1. crypt32.dll CertVerifyCertificateChainPolicy API; and
    2. crypt32.dll CertGetCertificateChain API.

These hooks enable GootKit to redirect the internet traffic through its own module and switch up the OS flag for an invalid certificate with a valid one, thus disabling any error alerts from the browser.

Redirecting Through the GootKit Proxy

When a new connection is initiated in the browser, GootKit's hooks change the endpoint's IP address and the listening port number to those predefined by GootKit earlier on. GootKit uses its hooks on MSAFD_ConnectEx and WSPConnect to check that the traffic type is relevant (TCP/IP). It looks at the header to ensure that sockaddr struct shows the number 16, which is the maximum length for nonhybrid structures in IPv4 addresses:

 *Figure 2*

 *Figure 3*

The next verification has to do with the modification of the endpoint's IP address from an external IP to a local one. GootKit checks that the sa_family is indeed AF_INET, which represents the internet protocol address format.


Figure 4: Sockaddr and SA_Family setup

GootKit modifies the endpoint's remote IP address to the local home IP (127.0.0.1) and switches the Transmission Control Protocol (TCP) port to its own selection based on a static value, to which it adds the original TCP port number.

 *Figure 5: The HEX value 0x100007F translates into the home IP address 127.0.0.1*

The TCP listening port is also changed:


Figure 6

The value 1770h translates to port number 6000.

Faking the Certificate Flag

Since traffic from the user to the bank is routed through the GootKit proxy at this point, GootKit's developers ensured that Secure Sockets Layer (SSL) error messages don't pop up and raise the victim's suspicion. To fake the validity flag, GootKit hooks a couple of APIs.

The first is called CertGetCertificateChain. This hook will ensure that every result returned from PCERT_CHAIN_CONTEXT with a pPolicyStatus parameter will change the flag to reflect a valid certificate (set to 0):

TrustStatus.dwErrorStatus to CERT_TRUST_NO_ERROR and TrustStatus.dwInfoStatus set to 0.

Figure 7

The second API hooked for this purpose is CertVerifyCertificateChainPolicy. Here, too, GootKit updates the PCERT_CHAIN_POLICY_STATUS error result to 0 for a valid certificate.

Figure 8

Conclusion

GootKit's developers are evidently still working on this malware project, enhancing its evasion techniques and its ability to control what victims see when they browse to their bank's website. This recent development in GootKit's operations on infected machines is consistent with the advanced, technical nature of this malware project.

Of attacks detected by IBM X-Force in the past year, GootKit was the most active in Europe in 2016, with configurations targeting the U.K., France and Italy. Per X-Force data, GootKit ranks eighth on the global list of the most active financial malware families per attack volume. Although it is very sophisticated, it keeps its infection and attack campaigns small and focused, which helps it fly under the radar and avoid detection.

Figure 9: Most prevalent global financial malware families per attack volume (Source: IBM Security, 2017 YTD).

Read the white paper: How to outsmart Fraudsters with Cognitive Fraud Detection

Gadi Ostrovsky
Malware Researcher, IBM

Gadi Ostrovsky is one of the top security researchers at IBM Security's Trusteer group. He joined the company in 2014, coming from a deep technical backgro...


IBM Security