ANOMALI®

Rise of Legitimate Services for Backdoor Command and Control

Steve Miller

Peyton Smith

Table of Contents

Abstract

There is a rise in the number of malware ecosystems that use legitimate internet services as part of a command-and-control (C2) schema, adding yet another layer of abstraction between attackers and network defenders. There is currently little public research on use of legitimate services for C2. The purpose of this paper is to detail the technique, note its rise in prevalence, and suggest experimental methods to detect it.

Backdoor Command-and-Control via "Legitimate Services"

Defenders rely on their ability to identify attacks early, providing enough time for a thoughtful and thorough incident response. Attackers, on the other hand, are constantly evolving their toolsets to subvert existing detection techniques and technologies. One of the most notable trends in the evolution of malware is the rise of command-and-control (C2) channels using so-called "legitimate services," or simply "legit services." In this context, and for the purposes of this report, "legit services C2" refers to malware abusing common internet services such as Twitter and GitHub, employing fake users and accounts, and otherwise utilizing such service APIs as part of a C2 schema. In this report we expand on this concept, identify challenges and potential detection techniques, and provide an appendix of examples of what this network traffic may look like.

We would like to state at the beginning of this paper that in describing legitimate services C2 we will be specifying internet service providers that are being abused by malicious actors. We by no means wish to imply that these providers are negligent in policing abuse, nor do we wish to suggest that these services are unsafe for use by the layperson. Each service provider listed is working unremittingly to identify and stop abuse whenever possible.

Before we detail the variety of legitimate services and malware using them, let's first address how backdoors typically connect and communicate.

Backdoor-Controller Relationship and Nomenclature

For the purposes of this discussion, we will bypass the common lingo for client-server relationships and instead deviate into specific malware vernacular.

In the greater information technology space, the "client" is usually a program/user on a host/workstation (inside your network), and, in that relationship, the client sends numerous requests to the server (outside your network). The server then provides resources back to the client system. This is the standard client-server relationship. Imagine, if you will, the size and the directionality of the network data in the standard client-server relationship. For internet browsing, a client system may send 10 packets (in this case implying data packets payloads, not just TCP packets) to a server for every 100 packets it receives, making the internal:external sent packets (or data size) a 1:10 ratio. This relationship (and ratio) is reversed in the world of malware.

A malware backdoor is implanted on a compromised host, yet instead of being a "client" and connecting to a "server" to ask for resources, the backdoor itself is the server. Let that sink in for a second. What is typically seen as a "server" and what is usually (inaccurately) referred to as a "C2 server" is often just an attacker-controlled system with a console or "controller" program. The attacker uses the command prompt or controller to connect to the backdoor "implant", to query and command the implanted/compromised system. Imagine the volume of output if the attacker runs each of the commands "systeminfo," "ipconfig /all," and "tasklist." For this internal reconnaissance effort, the compromised host may send 10 packets for every 1 packet it receives from the attacker controller. What should normally be a 1:10 ratio is reversed, and is now 10:1.

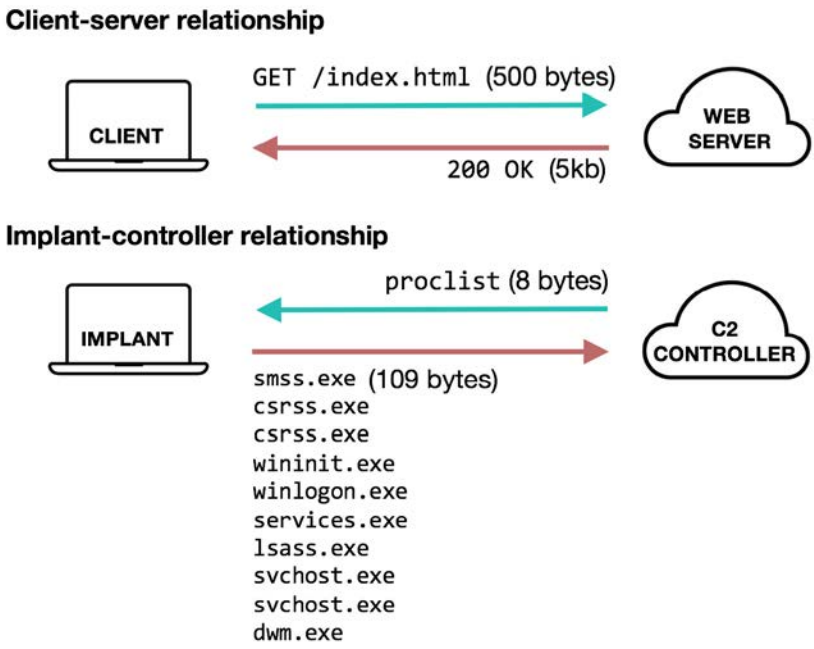


Figure 1. Illustration depicting the difference between a client-server and malware-controller relationship.

Thinking of the compromised system as a host and thinking of the C2 system as a server is a gross mischaracterization of the relationship. When you consider the directionality, the size, and the ratio of data sent from the remote (attacker-controlled) system to internal (implanted) system, you realize that the client-server relationship has indeed been inverted. Sure, we're mincing words, but the reversal of this standard relationship is important. The distinction in how you think about the flow of network traffic data is helpful as you invent creative ways to detect otherwise invisible backdoors.

Accordingly, rather than saying client/host and C2/server in this article, we will refer to the malware backdoor as an "implant" and the remote attacker system as the "C2 system" or "C2 controller." Generally we use the terms "backdoor" and "implant" synonymously, however "backdoor" is used more broadly to describe the entire class of malware with backdoor functionality, and Implant is used more to specify an instance of a backdoor when installed on a compromised system.

Furthermore, we will refer to the "C2 schema" as the totality of IP addresses, domains, legitimate services, and all the remote systems that are part of the implant's communications architecture.

When we say simply "C2" we mean command and control.

Two Types of Backdoors

Backdoors are differentiated from other types of malware because they can be interactively controlled from a remote location by a human operator. We break backdoors into two broad categories based on how they receive initial communique: active backdoors and passive backdoors.

Active Backdoors

Most backdoors are classified as active, meaning that the onus is on the implant to call out to its designated C2 system and tell the world that it is online and ready for instruction. That typically works like this:

1. Upon execution, the implant sends data to preconfigured C2 address (domain,IP, or URL) on some regular interval (such as every 60 seconds or every 5 minutes). We have different definitions for what this is and how it works based on the malware family, but this functionality is sometimes part of a "check-in," a "heartbeat," a "keep alive," or a "beacon".
2. Controller, if up, may conduct a "handshake" to verify that the implant is authorized to check-in. The C2 controller interprets some basic metadata sent in the initial traffic.
3. Implant and controller maintain a connection. When attacker/controller is

ready, it will send commands, instructions, or other information to the implant.

4. Backdoor implant parses the controller information, executes as necessary, and responds as necessary with results.
5. This usually requires both the implant and server to be "online" and successfully connected in some way.

The SOGU backdoor, for example, is considered an "active backdoor" because, upon execution and successful resolution/connection to its C2 address, the backdoor will send a "beacon" or "hello" packet.

Passive Backdoors

1. Upon execution, implant sets up a network listener on a pre-configured port.
2. Controller sends "magic packet" or password to implant on defined port when it has a command to be run. The controller does not talk to the implant unless necessary.
3. The implant parses the command, follows instructions, and responds if necessary with response data.
4. This requires the implant to be online and accessible from the Internet (externally addressable). The implant does not need a preconfigured C2 address. The C2 controller system can be anywhere and does not need to be online. C2 traffic to a passive backdoor implies an active human operator.

If webshells are considered backdoors, then they are passive backdoors. For example, the ASPXSPY webshell (sample on Github [1]) makes no outbound communications from the compromised system unless it first receives instructions from an external source.

It is worth noting that passive backdoors often:

- Require implantation on publicly addressable compromised systems (IP or domain)
- Require passwords or "magic" values for access
- Use only low-level APIs and custom binary protocols
- Do not use DDR or legit services in any way

Though it is outside of the scope of this discussion, we would like to note that searching for passive backdoors is an onerous task that begins with stacking network listeners and ends with sorting through a mountain of unsolicited inbound communications to the defended network. Other than hunting for webshells, few organizations have the resources or the tenacity to search for passive backdoors because the juice is not always worth the squeeze.

The distinction between passive and active systems is important in terms of how we think about what we're looking for and the types of network traffic we would expect for these systems. Active backdoors almost always beacon, and they typically employ a multitude of communications protocols.They also utilize different APIs and different C2 schemas. Passive backdoors do not typically use legit services for C2, often use low-level custom binary protocols, and usually simply lie in wait for someone to come knocking on the correct door with the correct password.

High-level vs Low-level APIs

It is also important to consider the types of APIs that backdoors use when communicating. Windows has dozens of network APIs across all layers of the OSI model. For the purposes of this discussion we will classify APIs into two groups, high-level and low-level.

High-level APIs

Most types of backdoors use high-level APIs. When doing static analysis of a backdoor sample you may see imports of wininet.dll or urlmon.dll and imported functions such as HttpOpenRequest, URLDownloadToFile, and FtpCommand.

While there is little flexibility to modify or add to the communications protocols involved, these libraries make networking simple and efficient because there are lots of built-in functionalities. For example, if an HTTP connection is established, certain HTTP functions will apply system defaults in areas where some headers aren't specified. These libraries may also allow for automatic proxy checking and easy proxy authentication.

Low-Level APIs

When malware authors are feeling frisky and require more flexibility, they may use low-level APIs such as Winsock, with libraries such as ws2_32.dll and functions such as Socket, Recv and Bind.

Low-level APIs have their benefits and drawbacks. Once a socket is created, every bit of the protocol is manually created. One might implement binary TCP or HTTP over a socket, however it requires additional laborious programming to make that happen. The flexibility in designing one's own C2 protocol introduces opportunities for errors, and makes things like proxy identification a pain.

Introduction to Legitimate Services C2

Organizations and vendors across the world tend to coin their own names for the same things. With respect to the topic at hand, some organizations will call this "alternative C2," "bravo channel C2," or any number of other terms. The dominant nomenclature for this technique is "legitimate services C2." Legit services C2 comes in many forms, but can be broken into two categories: Legit Services C2 "Dead Drop Resolving" (DDR) and Legit Services "Full C2."

Legit Services C2 DDR

Dead drop resolving is a technique where a backdoor is configured to look at a web resource to extract its true C2 address. This typically works in the following manner:

- Implant executes and reaches out to a web URL and scrapes HTML text
- Implant looks for specific tags/markers/delimiters in the text and extracts an encoded value
- Implant decodes encoded value, which contains commands and further config information such as a true C2 domain or IP address
- Implant initiates connection to true C2 address awaits further instructions

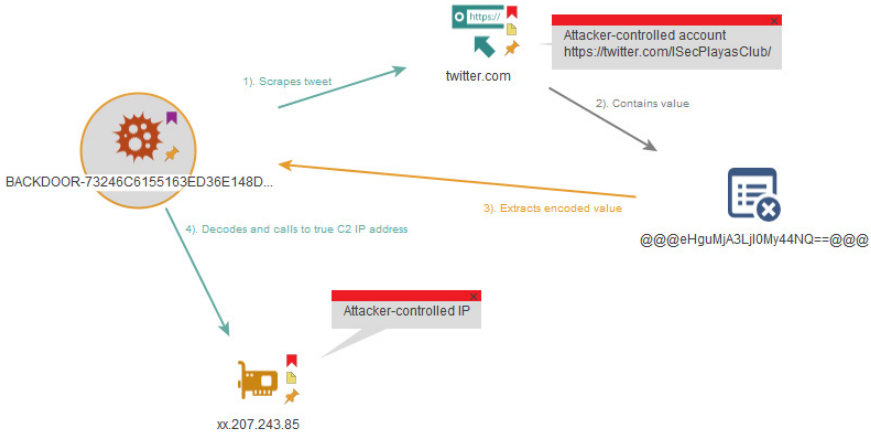


Figure 2. Legit Services C2 DDR

The name “dead drop resolving” is an allusion to traditional intelligence tradecraft, where one agent may secretly cache valuable information for another in a seemingly public place, allowing a security buffer between the dropping party and the receiving party.

Legit Services Full C2

Full C2 using legitimate services means that the backdoor and the controller system never talk directly, but rather they communicate through a middle party, using a legitimate service such as GitHub or Twitter to pass communications back and forth. This typically works in the following manner:

- Implant executes and uses hard-coded credentials to connect to a legit service.
- Implant scrapes account page or uses API to search for recent “comments” or “posts” or “updates” and looks for special encoded text within these portions.
- Implant decodes encoded value, which contains commands and other information. If the attacker is not ready for interactive access, the information will often contain configuration updates, sleep commands, or instructions on when the implant should call back to the legit service for updates.
- Controller connects to legit service and uses hard-coded credentials to apply updates and enter encoded values into account page or other data storage areas (attacker may also do this manually in some small capacity).
- Controller Attacker monitors sites/changes data to add new commands.

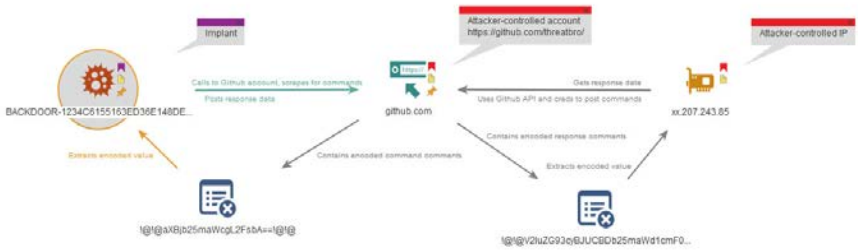


Figure 3. Legit Services Full C2

Using legit services for full C2 requires more effort on part of the malware authors, who must create backdoors and controllers with credentials necessary to use the service APIs.

Further Considerations for Use of Legit Services C2 DDR

Malware code families that use legit services for C2 DDR will typically use high-level APIs and HTTP compliant requests for initial communications, and then switch to low-level APIs and more non-traditional (custom) C2 protocols once the backdoor has acquired its true C2 address.

The implications of this typicality are twofold. Foremost, the initial C2 DDR communications in plaintext HTTP may represent a better (and potentially the only) network detection opportunity for these backdoors, especially if they switch to custom encrypted traffic post DDR. If you're wondering "...but I thought the legit services would be encrypted?" you'd be correct. However, some of the backdoors using legit services for C2 DDR are relying on the encryption of the service provider, and using regular HTTP APIs without making an HTTPS mandatory for communication. Believe it or not, sometimes the SSL handshake will fail and in that case, the connection would revert to HTTP and thus make HTTP-based methodology detections possible based on the initial GET requests.

In addition to the network detection aspects, there may be ways to identify legit services backdoor binaries based on the variety and volume of runtime imports.

Furthermore, backdoors that use legit services for C2 DDR are more likely to:

- Be classified as "active" and perform "beaconing" to pre-configured web resources
- Use high-level APIs such as HTTP to perform DDR
- Use low-level APIs and custom binary protocols for post-DDR C2

Backdoors that use legit services for Full C2 are more likely to:

- Use high-level Windows APIs and HTTP protocols to scrape legit service accounts and pages
- Contain hard-coded account credentials for legit services

Notable Legit Services Being Abused

The number of legitimate services used for C2 is nigh unlimited. The following list is a sample of high profile and otherwise prolific services that have been observed in backdoor C2 schemas in the last few years.

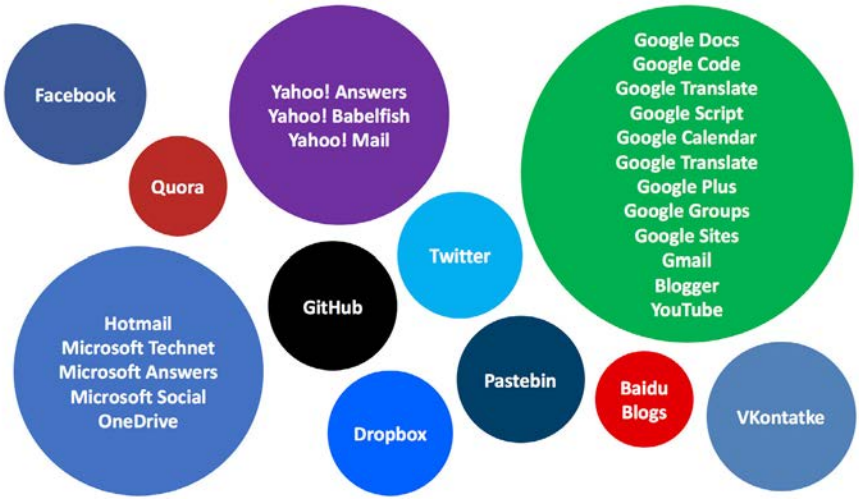


Figure 4. A variety of legitimate services seen abused for C2

Notable Malware Families Using Legit Services C2

There are likely hundreds of distinct malware code families that have used (or are currently using) legitimate services for C2.

The table below describes a few notable malware code families that have been observed using legitimate services for C2. This is but a fraction of the total code families and legit services used for C2. In terms of the observed legit services used, these examples may indicate attacker preferences (in configuring samples) rather than capabilities of the backdoors or functionalities of the legit services themselves.

Please note that for aspects of C2, many malware families are fully customizable and possess the ability to communicate in multiple ways. This is especially true for code families that are actively developed. Accordingly, the following details are not expected to be permanently comprehensive.

[See table on the next page]

Table 1. Notable malware code families observed using legit services C2

Code Family	AKAs	Use Type	Services Used	Notes
SOGU	Kaba, Gulpix, PlugX, Thoper, Destory	DDR Only	Microsoft Answers Microsoft Technet Google Code Pastebin GitHub	Used by a multitude of China-based APT actors
BLACKCOFFEE		DDR Only	Microsoft Technet	APT17
WHISTLETIP		DDR Only	Microsoft Social	
BARLAIY	POISONPLUG	DDR Only	Microsoft Answers Microsoft Technet Pastebin GitHub	
BELLHOP	ggldr	Full C2	Google Docs	Associated with CARBANAK (and possibly FIN7)
RAINYDROP		Full C2	GitHub	APT3
HAMMERTOSS	HammerDuke, NetDuke	Full C2	Twitter GitHub	Twitter -> URL -> Git Stego Image w Cmds and Creds
LOWBALL		Full C2	Dropbox	Purportedly associated with threat actor **admin@338**

Motivations and Technology Drivers for Abusing Legit Services

Use of legitimate services for some form of C2 dates back to at least 2009 [2]. While there have been a few incidents of botnets and worms using legit services for C2, at the time of this writing, the technique is usually employed only by so-called Advanced Persistent Threat (APT) actors and state-sponsored (enabled or tolerated) threat groups. This technique may have started with traditional state-sponsored groups such as APT1, but has since branched out into many other groups and backing nations, as well as gaining traction in the international criminal underground.

There are several driving forces for the adoption of this C2 technique. First, when using legit services for C2 the malware network traffic becomes nearly impossible to identify because it mimics the behavior of legitimate network traffic. This is in part driven by the "open workplace," bring-your-own-device and telecommuting movements.

Modern enterprises that give their employees latitude in internet usage (allowing social media and unfettered web access) are ultimately providing an auspicious cover (and opportunity) for emerging threat actor tactics. Social media and encrypted cloud services are everywhere. Many users rely on Google Docs, OneDrive, and Dropbox to get their work done, regardless of whether these services are offered by or systemically endorsed by their employer. With legit services C2, the threat actor activity is blending in with the noise — and defenders have few ways to differentiate the good from the evil.

While use of legit services C2 is on the rise, common malware such as ransomware and botnet implants rarely use this technique. Some publicly available penetration testing frameworks such as Empire and Powersploit have been implemented to use legitimate services like Pastebin to redirect or download malicious text code, however, this is a trivial use of the service and it does not come with the benefits of other, more comprehensive legit services.

There is a barrier for entry in using legit services C2 because it may require more complicated programming for malware tools. It may also induce overhead in managing legit services accounts. This requires thoughtful organization and nuanced management of the infrastructure, which is more diverse in nature than traditional C2 schemas.

If there is a rise in the use of legitimate services C2, it must be happening for a significant reason. Attackers, like any goal-oriented people, are driven by convenience, cost, and operational security.

Advantages of Using Legit Services for C2

- It is easy to hide your C2 inside communications that are believed to be good.

    - Nobody questions network traffic to Google, Microsoft, Twitter or Github.
- It is easy to register new accounts on these services.
    - There is minimal vetting to create new accounts and personas for most public cloud services and social media services.
    - Authorities such as Twitter, Microsoft and Github claim to be cracking down on account abuse, but it is to sign up for a new account and remain undetected. Service providers may be able to programmatically reduce abuse by "bot" accounts, however those controlled by human attackers are much more difficult to tease out of the haystack.
- It is easy to get a "web page" up somewhere on the publicly accessible internet.
    - Never in our history has it been so easy to get public data up somewhere.
    - Image and text "paste" and "dump" sites make this simple.
- It is easy to usurp encryption for your C2 protocols
    - Why set up C2 servers with encryption and build encryption into your malware if all you need to do is use a legit service and adopt its SSL certificate?
    - This is worth it for the convenience alone, but it provides the added benefit of a publicly endorsed SSL stream that makes the C2 traffic nearly undetectable.
- It is easy to adapt and transform when the situation becomes complicated.
    - You can reconfigure implants in the moment without waiting for DNS updates.
    - You can reuse implants across attacks without reusing DNS or IP addresses.
- You can reduce likelihood of burning your C2 infrastructure (better OPSEC).
    - Putting a major service provider in the middle of your C2 schema and make it difficult to detect and block your malware communications.
    - No more hard-coding malware with your IP addresses and domains. When your operation is done, you simply take down your legit services pages and nobody will ever know your IP addresses.
    - Never register a domain or SSL certificate again! Attribution for cyber threats is primarily based on registrants, domains, and IP addresses. Legit services places an immense layer of anonymity between attackers and their victims.
- You can reduce overhead and increase ROI and other business metrics.
    - If switching to legit services C2 means you succeed in your attack mission more often, and also spend less time and money retooling, then it is a smart investment.

Challenges Presented to Defenders

As mentioned in the previous paragraphs, using legitimate services for C2 has many benefits that are in turn challenges for defenders.

Legit services are difficult to block — If you are a large international enterprise, you know how difficult it can be to remediate compromised systems around the world. Sometimes you may simply implement IP or DNS blocking at an egress point, or potentially just sinkhole the malware C2 address. However, with legitimate services this may become impossible. Do you have the technical capability to block a full Uniform Resource Identifier (URI)? And if so, does the service actually utilize a unique URI for the C2 landing page or rely on being "logged in" and dynamically delivering the content? Can you risk blocking parts of Google or Twitter?

Legit services are often encrypted and innately difficult to inspect (difficult to monitor/enforce for misuse) — SSL decrypting is expensive and not always possible at enterprise scale, so the malware hides its communications inside of the encrypted traffic, making it difficult, if not impossible, to identify the evil traffic at all (unless you locate the malware on the endpoint). Even if you do identify evil via the endpoint, if you do not know the profile pages or exact location in the legit service that is being used, you may never be able to extract the encoded information or identify further C2 addresses, commands, and responses that are stored on these services pages — making the effectiveness of your incident response negligible.

Use of legit services subverts domain and certificate intelligence — Many companies buy indicator feeds for reputation filtering and indicator blacklisting, yet many of these feeds are based on newly generated and newly registered domains, certificates, and IP addresses connected thereto. Using legit services for C2 will circumvent all of this for obvious reasons.

Use of legit services complicates clustering and attribution — A huge amount of threat intelligence is based on clustering IPs, domains, email addresses, and other registrant information in order to form groups and serve as the basis for attribution. Anyone that tells you different is full of it. With a shift to legit services C2, it is possible to move away from domain registration because you can consider the legit service account as the initial C2 address. No longer will truly sophisticated attackers continue to register SSL certificates or use self-signed SSL certificates for C2 schemas, which we all know has historically played a big part in tracking and clustering threat activity. Furthermore, backdoor binaries no longer require hard-coding real C2 addresses, so even if you find a sample on an endpoint, you may never be able to trace that to an attacker IP address if it has been removed from the legit service.

Experimental Detection Methodologies

Detecting legit services C2 is fundamentally difficult, thus the methodologies we are discussing are experimental and will likely yield a number of false positives when first tested. We recommend evaluating these methodologies on a small subset of your network, then spending some time whitelisting and tuning the logic before deploying to an entire enterprise.

For detecting evil traffic to legit services, we can apply the simple thesis that browsers are smart and malware is stupid. Browsers have undergone decades of development to optimize network usage with things like caching, cookies, and session memory. Even though some network traffic is encrypted, there will be differences in how a piece of malware communicates with a legitimate services. We suggest exploring the following four experimental methodologies to detect malware using legitimate services for C2.

1. Non-browser non-app process network connections to legitimate services

Endpoint products from vendors such as Vector8, Tanium, Crowdstrike, Carbon Black and Mandiant (ask about "HIP") may be able to inspect system data and trigger on non-browser process network connections. Using these triggers, you may be able to tease out network connections to IP ranges for things like Microsoft Technet or GitHub or Twitter, thereby identifying source processes of interest for further investigation.

For this method, we recommend creating specific rules for each legitimate service. If you're looking for legit services C2 to GitHub, create a rule for non-browser process connections to GitHub IP space that are also non-app processes such as Git.exe and GitHub Desktop. Purely as an example, the abstract detection logic might look something like this:

```
Source Process NOT (firefox.exe OR chrome.exe OR iexplore.
exe) AND TCP Connection To (netblock is 192.30.252.0/22 OR
AS is AS36459) AND Source Process NOT (git.exe OR github.
exe OR OR git-bash.exe OR git-cmd.exe OR git*.exe OR git-
gui.exe OR githubdesktop.exe)
```

You could also tune this logic to look for traffic sourced from designated parts of your enterprise, such as source subnets or lists of computers that should never be talking to legitimate services. At first, this will probably generate a fair number of false positives for apps and social media programs, but after tuning this is a viable way to find lots of weird and potentially unwanted things communicating with these services.

2. Unique or low DDR page response traffic sizes from legitimate services

When pieces of malware make HTTP GET requests to profile pages or other legit services pages, they're typically doing a dirty download of HTML to a temp file, which is then later processed for the encoded DDR text to find the real C2 address. This is fundamentally different from how a browser views a page because a browser will download and render additional linked content from the page. This opens up an opportunity to "fingerprint" near-default page sizes for legit services profiles.

As a small experiment, we simulated malware GETs of new profile pages on a few legit services to get an idea of the network flow differences.

Table 2. Experiment data depicting the differences between browsing and raw page downloads

Profile/Page	Base Page (Raw)	Get Page HTTPS	Browsing Page
MS Technet 1	41kb	50kb	72kb
MS Technet 2	41kb	49kb	72kb
MS Social 1	27kb	35kb	69kb
MS Social 2	27kb	25kb	69kb
GitHub 1	33kb	42kb	100kb+

Even with the TLS/SSL encryption overhead, the raw HTTPS GETs of the profile pages were significantly smaller than browsing the pages in Chrome. This suggests the possibility of identifying the abuse of legit services using fingerprint sizes to identify abnormally small network flows.

In the opening paragraphs we discussed the reversal of the standard client-server relationship in terms of directionality and ratios of data sent and received. We can expand further on this concept by thinking about this for legit services C2 as well. For example, if a backdoor is using GitHub for full C2, it might be searching a page or a project for encoded instructions and then providing responses to those commands in the form of comments. In this example, the backdoor would be sending a high volume of data to GitHub over a long period of time and receiving very little from the page or project itself, likely less than 1kb per command received. This might characterize the natural flow of committing or uploading to Github, but a time graph of active malware C2 versus a legitimate upload will show entirely different patterns visualized. There are opportunities here for profiling or "fingerprinting" standard traffic and looking for things that deviate from the norm.

Consider the following anecdote. In one intrusion where the threat actor deployed malware using legit services C2, a keen defender noticed suspicious network flows to

GitHub IP space where the average response size from GitHub was between 11000 and 15000 bytes. These sizes, mind you, are significantly smaller than the average web page size for a profile or project page, leading the incident responder to suspect non-browser traffic, and potentially something malicious. With this behavior in mind, he crafted a netflow query to search for the top flows with bytes in a designated size range. Using this netflow query, the responders identified additional machines worth investigating, ultimately locating the backdoors using the GitHub API for C2.

```
nfdump -R %NFDUMPFILES% -t YYYY/MM/dd.00:00:00-YYYY/
MM/dd.23:59:59 "(src net 192.30.252.0/24 or src net
192.30.253.0/24) and (bytes > 9000 and bytes < 17000)" -s
dstip/bytes -n 25
```

3. High certificate exchange frequencies to legitimate services

Once again, with the assumptions that backdoors are stupid, we can consider that every time a backdoor calls out to its DDR page there will be an SSL handshake and certificate exchange. To illustrate the difference in cert exchange frequency, we compare on the left 20 minutes of malware GETs and on the right 20 minutes of browsing to a Microsoft Social profile. The malware does indeed prove to be stupid.

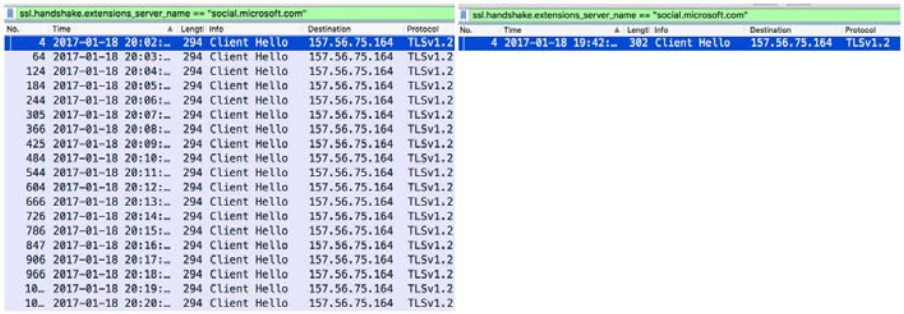


Figure 5. (left) 20 minutes of malware GETs to a Microsoft Social Profile; (right) 20 minutes of browsing to a Microsoft Social profile

One might implement detection logic for this behavior by creating Snort or Suricata rules looking for "social.microsoft.com" in an SSL certificate where the certificate is presented more than, say, 50 times per day to the same internal IP address (implying that the internal system was making an unusually high number of GETs to Microsoft Social). This is just an example, of course. This methodology of measuring certificate exchange numbers can be applied to any of the legit services as long as they are encrypted.

The following Snort rules demonstrate the logic that may be used to identify

systems emanating unusually voluminous requests for SSL certificates:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 443 (msg: "TLS
Client Hello - Microsoft Answers"; content"|00 00 00 19 00
17 00 00 14|social|2e|microsoft|2e|com"; threshold: type
both, track by_src, count 30, seconds 86400; sid:13370001;
revision:1;)

alert tcp $HOME_NET any ->  $EXTERNAL_NET 443 (msg:"TLS
Client Hello - Google Docs"; content:"docs.google.com";
ssl_state:client_hello; threshold: type both, track by_
src, count 30, seconds 86400; sid:13370002; revision:1;)
```

This same logic can be deployed on both client-side and server-side certificate exchanges. We have specified client-side here because they represent traffic that is "closer to the malware". Naturally, the specific logic and format of these rules would depend on the implementation of Snort, which may include certain traffic preprocessors and special configurations for IP ranges and ports. Moreover, the thresholds for alerting would need to be fine tuned and tailored to the resident network. Still, we hope that this helps illustrate the potential for methodology detections based only on network traffic.

4. Bulk processing samples for suspicious DNS calls to legitimate services

There is an argument to be made for malware "hunting" in bulk by processing large amounts of samples in a sandbox. For example, if you purchased some type of "sandboxing" appliance that "detonates" inbound binaries for malware analysis, you could implement special rules for those evil-looking binaries that make DNS requests to legitimate services and legit services APIs. For example, if you see a sample make a call to "api-dropboxusercontent.dropbox.com," even if the sample is not deemed "malicious," it may merit more scrutiny before you allow such a binary into your enterprise environment. When you do find samples like this that are malicious, you can create tactical detection rules or blacklists on a per-sample, per-family or per-functionality basis. (We maintain a philosophical objection to detection by file hash, which we view as trite and ineffective, but we do acknowledge that in this case there is a small use case for hash blacklisting).

5. Bulk processing samples for morphology matching

Morphology means the study of the form and structure of things and the relationships between things. In our little slice of the cyber universe, "morphology matching" means identifying similarities between files with the purpose of identifying unknown

malicious files based on relationships to known-good and known-bad files. Pretty simple, right? Though this is not specific to legit services C2, we see a lot of promise in the method of bulk processing large amounts of samples using "binary similarity" or "morphology" malware technologies. This detection methodology would focus on file attributes rather than network behavior.

There are several names and buzzwords for describing this concept and how it might work in practice. The basic premise of this idea is that you take a piece of malware and perform static attribute analysis, disassembly, and dynamic analysis to extract a list of tool marks and features such as: opcodes, byte patterns, symbolic functions, system calls, code blocks, processor instructions, debug symbols, unique strings, and run-time behaviors. You combine all of these features into a special pattern (a/k/a "morphology") that describes the attributes that are unique to this single sample and the entire family of malware. Then you compare the "morphology pattern" to a database of other patterns to help you identify or classify your unknown file as malicious or non-malicious. At the highest level it does not seem like rocket science, but engineers and data scientists have been working on this problem for decades and still haven't nailed down a great solution.

We believe that by bulk processing samples for morphology matching, you may be able to identify samples using legit services C2 that would otherwise not be detected by standard endpoint or network detection methodologies. At this time, there are few commercial or open source technologies that offer morphology matching with a solid reference database. We can, however, highlight two notable examples. VxClass by Zynamics (acquired by Google in 2011 and unfortunately VxClass is no longer for sale), performed disassembly of binaries and used "bioinformatics algorithms to classify malware info "family trees based on a matrix of similarity values". More recently we can note products from Intezer, a Tel Aviv startup that describes their technology as "DNA mapping for software". The Intezer technology, "dissects any given file or binary into thousands of small fragments, and then compares them to Intezer's Genome Database, which contains billions of code pieces ('genes') from legitimate and malicious software offering an unparalleled level of understanding of any potential threat." We've used only the community edition, but in our opinion, Intezer has the most promising commercial offerings in the malware morphology space. We have yet to determine if deploying such technology in large scale security operations would indeed find malicious files based only on code similarity or "gene" matching.

Data Points on Rise of Legit Services C2

In March 2017, I co-presented "Middle-out Network Analysis: Finding Evil with a Low Signal-to-Noise Ratio" at Bsides Canberra, Australia. In this presentation we discussed

the rise of legit services C2 and detailed a longitudinal study of malware capabilities that we called "Evolution of Malware," all of which was publicly presented under the auspice of Mandiant and FireEye.

In short, we studied ten years of malware capabilities to get a better understanding of how things like C2 protocols were changing over time. The data set was comprised of all the malware samples that underwent reverse engineering by Mandiant's incident response and intelligence teams from 2006 to 2016. Analysis of these samples presented several trends, including the rise of "marketshare" for samples that used legit services C2.

Though these trends do not necessarily reflect what's happening in the world at large, after looking at the stats from this data set, we can indeed see that Mandiant/FireEye is seeing an increase in the amount of malware using legit services C2 each year. This research was presented publicly at Bsides Canberra in early 2017. (See Appendix D for further details.)

The biggest takeaway from all of this is that, for malware involved in some of the most notable cyber attacks, the biggest data breaches, and the most world-shaking intrusions… the use of legit services for C2 is on the rise. This has several implications.

Chart A. This graph depicts the percentage of samples analyzed annually that are capable of legit services C2. From 2008 to 2011, legit services C2 samples represented approximately 3% of the annual submissions, later rising to 6% in 2014 and 9% in 2016. According to this data set, the percentage of annual samples using legit services C2 tripled in the last decade.

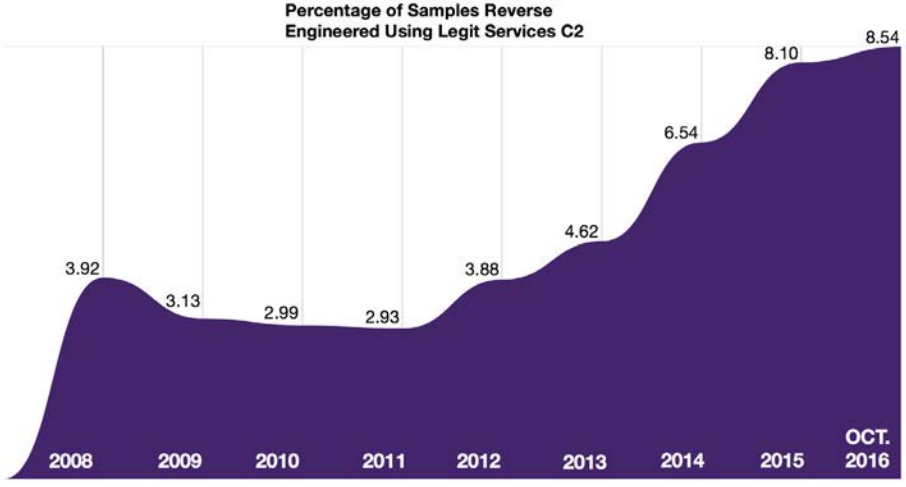
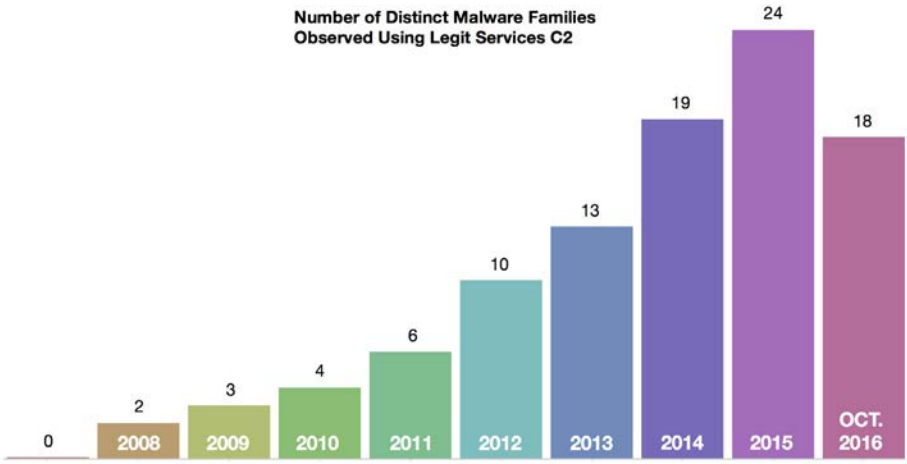


Chart B. This graph depicts the raw number of unique families seen "in the wild" using legit services C2. In 2006 there were 4 distinct malware families using legit services for C2, rising to 26 unique families active in 2016.



Key Points and Conclusions

- Backdoors that use legit services C2 are almost always active, and perform beaconing to legit services websites and APIs.
- Backdoors that use legit services C2 for DDR often switch from high-level Windows APIs and HTTP protocols to low-level Windows APIs and custom protocols that are very difficult, if not impossible, to detect with traditional network detection rules.
- We have observed a rise in abuse of legitimate services for attack operations, and we expect this trend to continue from both the actor group perspective and the malware code family perspective.
- We advise pentesters and red teamers to look into this technology as well, both in terms of maturing the attack infrastructure and becoming undetectable/unattributable.
- We advise defenders to explore experimental methodologies for detecting these C2 techniques.
- We recommend security researchers everywhere begin to measure, study, and develop methods to mitigate these C2 techniques.

Threat actors are in the business of conducting network intrusion operations,

which are costly in resources. For long term mission success, they must use their resources wisely, adapt when necessary, and invest in strategies that allow them to scale and operate undetected for long periods of time.

If the threat actors (or if a development "quartermaster" in the supply chain) create attack tooling that is easily found by detection systems, not only would it ruin the success of any current intrusion operations, but it would also cost time and money before the mission could resume. This is why security researchers hasten to publicly unveil attack technologies, effectively forcing the attackers to retool, and ideally driving up the cost of conducting an attack.

Threat actors need to deploy malware and maintain access to their implants in victim networks. It is easier to hide and maintain access to these implants if they can be controlled through the conduit of legitimate services. We are detailing use of legitimate services C2 because threat actors are investing in this technique and we expect the use of this technique to increase in the coming years.

We hope that this paper has been interesting, useful, or at least fun to read. There are few resources on legit services C2, and, to date, longitudinal research on the general topic of malware C2 capabilities has been shaky at best. This paper is admittedly no better, but we hope it serves to illustrate the type of study necessary for defenders to do what they do best: find evil and solve crime.

Appendix A: Example Malware Families and Network Traffic

SOGU DDR to Microsoft TechNet (profile)

SOGU is a versatile, full-featured backdoor ecosystem that has been around at least 10 years. While this tool is not exactly publicly accessible, it is believed to be controlled and shared by multiple threat actor groups originating in China. The SOGU ecosystem has been branched and modified several times. In the example below, this SOGU builder/controller program enables easy DDR address designation and supplies the encoded value to be pasted into the legit service or webpage used. This functionality has been documented extensively [3] by several [4] security researchers [5].

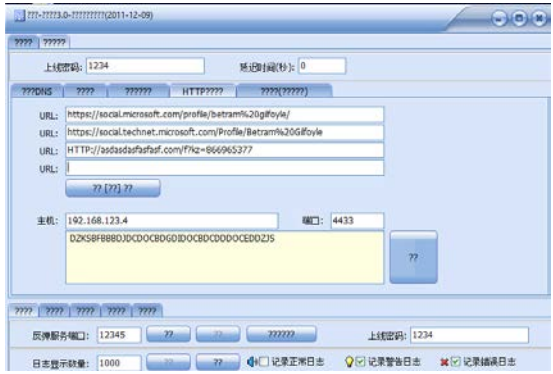


Figure 6. SOGU builder configuration options for DDR

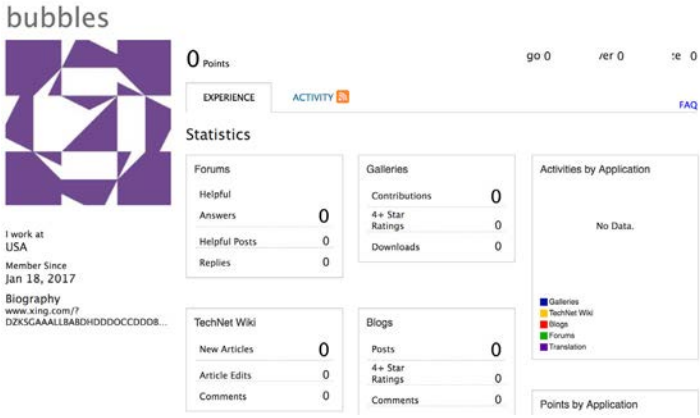


Figure 7. Microsoft Technet profile displaying encoded data for a SOGU follow-on C2 address.

BLACKCOFFEE DDR to Microsoft TechNet (forum post

BLACKCOFFEE is a backdoor once seen used by the threat group APT17 aka DeputyDog. This backdoor has been observed using Microsoft Technet profiles and forum posts for DDR C2. Similar to other backdoors, BLACKCOFFEE makes an HTTP GET request to the designated page and looks for specific delimiters. In this example, the attackers store a custom encoded IP address between the tags @MICRoSoFT [6] and CoRPoRATIoN. The backdoor decodes the text between, which provides the true C2 IP address.

```
GET /Forums/en-US/22b7efb1-bcf2-491c-9170-c24608f2fa98/
system-startup-gpo-fails-to-run?forum=winserverGP HTTP/1.0
Accept-Language: en-US
Pragma: no-cache
Connection: close
Host: social.technet.microsoft.com

HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/7.5
Set-Cookie:
```

Figure 8. HTTP GET request to a forum post on Microsoft Technet.

```
outline:0px; padding:0px; color:#333333; font-size:13px; line-height:
14.266666412353516px"></strong><strong style="border:0px; font-
family:Verdana,sans-serif; margin:0px; outline:0px; padding:0px; color:#333333;
font-size:13px; line-height:14.2666664123535px"><span style="font-
family:Verdana,Geneva,sans-serif; font-size:0.75em; line-height:
1.5">@MICR0S0FT[redacted]E0RP0RATI0N</span></strong></p></div>
                    <input type="hidden" id="9d5563ba-c1f3-4da1-b6f3-
eb71cdd0f0f8_attachments" value="" />

                </div>
                    <div>
```

Figure 9. HTTP response containing encoded information.

WHISTLETIP DDR to Microsoft Social (profile)

WHISTLETIP is a first-stage backdoor that has been observed using Microsoft Social profiles for DDR C2. In 2016, Mandiant researchers reported that a sample of this backdoor made an HTTP GET request to a Microsoft Social profile page and read the HTTP response, parsing the strings in the Biography section for designated HTML markers. The backdoor would then base64 decode and RC4 decrypt these special strings, the result of which was a URL for the next stage download. Finally, the backdoor would download and decrypt the resource at the URL and inject this code into explorer.exe memory, effectively launching a second-stage memory-resident backdoor called JOYRIDE that uses the same DDR technique for its C2.

```
GET /Profile/[redacted] HTTP/1.1
Accept: */*
Accept-Language: en-US
Accept-Encoding: none
User-Agent: Mozilla/4.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0; .NET5.0C; .NET5.0E)
Host: social.microsoft.com
Connection: Keep-Alive

HTTP/1.1 200 OK
Content-Length: 25183
Content-Type: text/html
```

Figure 10. WHISTLETIP HTTPS GET request to Microsoft Social profile.

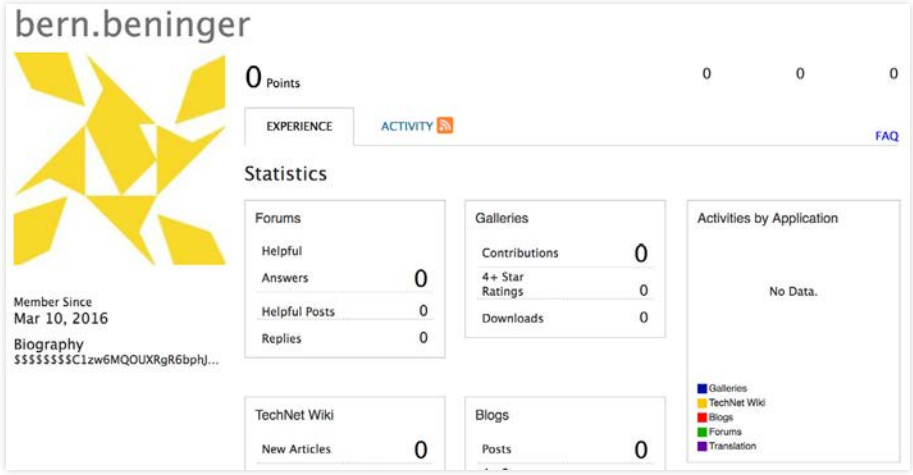


Figure 11. WHISTLETIP-specific Microsoft Social Profile with encoded information in the Biography section.

BARLAIY DDR C2 to GitHub and Microsoft Social

BARLAIY is a rare, advanced, modular plugin framework with full backdoor functionality. With respect to its advanced nature and malleability, it appears as though significant investment went into the design and development of the framework, suggesting that its authors intend to be use it for a long time. This malware has several built-in anti-analysis checks that prevent it from being easily sandboxed. Anomali Labs triaged this malware to help illustrate the type of communications it makes.

Figure 12. BARLAIY sample A initial communications to a GitHub project

Function Name	Parameters
GetTempFileNameA	filename = C:\Users\DSSDPM~1\AppData\Local\Temp\BoDo.tmp, path = C:\Users\DSSDPM~1\AppData\Local\Temp\

URLDownloadToFileA	url = https://github.com/alexsteven33/mobile-phone-project/blob/master/.classpath, filename = C:\Users\DSSDPM~1\AppData\Local\Temp\BoDo.tmp

Figure 13. BARLAIY sample B initial communications to a Microsoft Social profile

Function Name	Parameters
GetTempFileNameA	filename = C:\Users\HJRD1K~1\AppData\Local\Temp\4E4E.tmp, path = C:\Users\HJRD1K~1\AppData\Local\Temp\
URLDownloadToFileA	url = https://social.microsoft.com/profile/chrisweaver049, filename = C:\Users\HJRD1K~1\AppData\Local\Temp\4E4E.tmp

Figure 14. Microsoft Social profile page downloaded in response to the requests in Figure 13.

BELLHOP Full C2 to Google Docs

FireEye iSIGHT Intelligence reported in 2017 on a new malware family called BELLHOP, a Javascript backdoor that uses Google Docs for C2. Forcepoint researchers publicly blogged [7] about this functionality, explaining that BELLHOP was dropped by malicious RTF documents with embedded VBScript. Once decoded, the backdoor payload uses Google Apps Script, Google Sheets and Google Forms services for C2. Anomali Labs performed runtime analysis on a sample BELLHOP dropper document (4b783bd0bd7fcf880ca75359d9fc4da6), the results of which are provided below as an example of how this backdoor may use legit services C2.

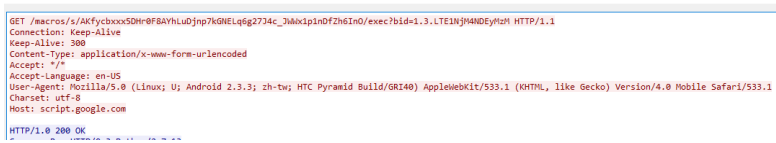


Figure 15. BELLHOP HTTPS GET request to Google Script macro.

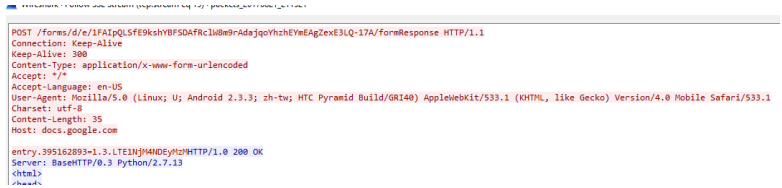


Figure 16. BELLHOP HTTPS POST request to a Google Docs Form document

RAINYDROP Full C2 to GitHub

In 2016, FireEye reported on a new APT3 backdoor designated RAINYDROP that used GitHub user accounts for C2. Anomali Labs scoured the internet for samples that are likely to be RAINYDROP to help illustrate this form of legit services C2.

```
GET /login HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/xaml+xml, application/x-
ms-xbap, application/x-ms-application, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR
3.0.04506.648; .NET CLR 3.5.21022; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
Host: github.com
Connection: Keep-Alive

HTTP/1.1 200 OK
Content-Length: 59745
Content-Type: text/html

<html xmlns:v="urn:schemas-microsoft-com:vml"
xmlns:o="urn:schemas-microsoft-com:office:office"
xmlns:w="urn:schemas-microsoft-com:office:word"
```

Figure 17. RAINYDROP HTTPS GET request to GitHub

Figure 17a. Sample RAINYDROP initialization routines and communications to GitHub.Note that some of the values and strings here are specific to our virtual execution and are not necessarily tool marks of the malware.

Function Name	Parameters
CoCreateInstance	(in: rclsid=0x75e4a0*(Data1=0x8856f961, Data2=0x340a, Data3=0x11d0, Data4=([0]=0xa9, [1]=0x6b, [2]=0x0, [3]=0xc0, [4]=0x4f, [5]=0xd7, [6]=0x5, [7]=0xa2)), pUnkOuter=0x0, dwClsContext=0x1, riid=0x75e490*(Data1=0x0, Data2=0x0, Data3=0x0, Data4=([0]=0xc0, [1]=0x0, [2]=0x0, [3]=0x0, [4]=0x0, [5]=0x0, [6]=0x0, [7]=0x46)), ppv=0x75e488 | out: ppv=0x75e488*=0x90a560) returned 0x0

WebBrowser:IUnknown:AddRef	(This=0x90a560) returned 0x10
WebBrowser:IUnknown:QueryInterface	(in: This=0x90a560, riid=0x75e2b0*(Data1=0xd30c1661, Data2=0xcdaf, Data3=0x11d0, Data4=([0]=0x8a, [1]=0x3e, [2]=0x0, [3]=0xc0, [4]=0x4f, [5]=0xc9, [6]=0xe2, [7]=0x6e)), ppvObject=0x75e290 | out: ppvObject=0x75e290*=0x90a740) returned 0x0
WebBrowser:IUnknown:Release	(This=0x90a560) returned 0x10
IUnknown:AddRef	(This=0x90a740) returned 0x11
IWebBrowser2:Navigate2	(This=0x90a740, URL=0x75d9d8*(varType=0x8, wReserved1=0x0, wReserved2=0x0, wReserved3=0x0, varVal1="https://github.com/login", varVal2=0x0), Flags=0x75d9c0*(varType=0x3, wReserved1=0x0, wReserved2=0x0, wReserved3=0x0, varVal1=0x0, varVal2=0x0), TargetFrameName=0x75d9a8*(varType=0x0, wReserved1=0x0, wReserved2=0x0, wReserved3=0x0, varVal1=0x0, varVal2=0x0), PostData=0x75d990*(varType=0x0, wReserved1=0x0, wReserved2=0x0, wReserved3=0x0, varVal1=0x0, varVal2=0x0), Headers=0x75d978*(varType=0x0, wReserved1=0x0, wReserved2=0x0, wReserved3=0x0, varVal1=0x0, varVal2=0x0)) returned 0x0
SysStringLen	[0033.038] (param_1="{D0FCA420-D3F5-11CF-B211-00AA004AE837}") returned 0x26

CreateWindowExW	(dwExStyle=0x90000, lpClassName="WindowsForms10. Window.8.app.0.378734a", lpWindowName=0x0, dwStyle=0x2010000, X=-300, Y=-300, nWidth=10, nHeight=11, hWndParent=0x0, hMenu=0x0, hInstance=0x610000, lpParam=0x0) returned 0x30032
SysStringLen	(param_1="Waiting for https://github. com/getlook23/project1/issues/1...") returned 0x3d
IOleCommandTarget:Exec	(in: This=0x1d43c678, pguidCmdGroup=0x75d140, nCmdID=0x0, nCmdexecopt=0x0, pvaIn=0x0, pvaOut=0x75d128*(varType=0x0, wReserved1=0x0, wReserved2=0x0, wReserved3=0x0, varVal1=0x0, varVal2=0x0) | out: pvaOut=0x75d128*(varType=0x8, wReserved1=0x0, wReserved2=0x0, wReserved3=0x0, varVal1="https:// github.com/getlook23/project1/issues/1", varVal2=0x0)) returned 0x0

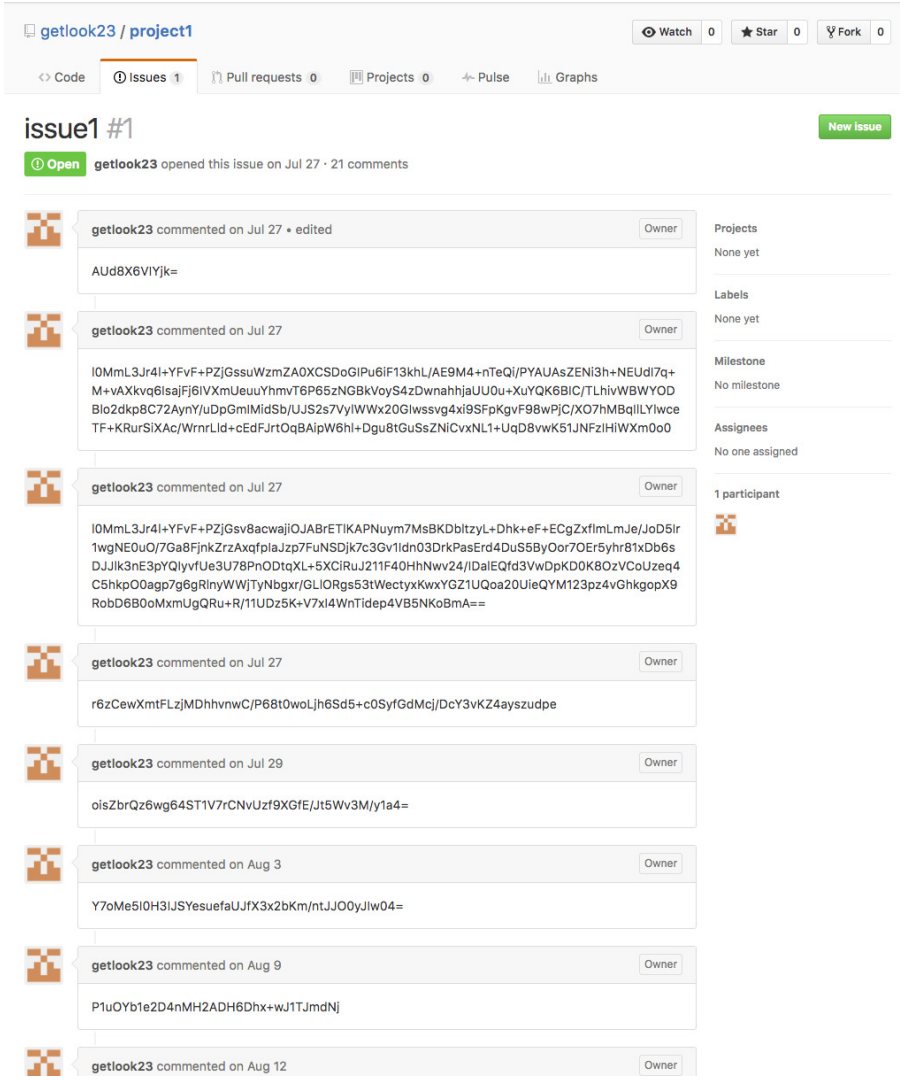


Figure 18. GitHub page with encoded RAINYDROP C2 traffic. This malware family and traffic was courteously decoded and documented by Sean Wilson of Open Analysis [8]. Big ups to OA.

HAMMERTOSS Full C2 to Twitter

HAMMERTOSS, one of many sophisticated malware ecosystems used by Russian threat actor APT29, has been extensively documented by F-Secure [9] and FireEye [10]. HAMMERTOSS has been observed in the wild using complicated, multi-layered C2 schemas involving Twitter, Github and steganography. In one usage of this malware family, a sample of HAMMERTOSS would scrape a Tweet for a link to a Github URL, use a hashtag to identify a value, download the image from Github and use the hashtagged value to decrypt/decode the image starting from a particular offset. Have fun detecting that mess.

Figure 19. Now defunct Twitter profile hosting a tweet used by HAMMERTOSS for C2 (image courtesy of FireEye)

LOWBALL Full C2 to Dropbox

First detailed by FireEye in 2015 [11], LOWBALL is a backdoor family used by that uses a hard-coded access token and the Dropbox API for C2. The malware has the ability to upload, download and execute files.

Table 3. Initial LOWBALL communications via the Dropbox API.

Function Name	Parameters
InternetOpenA	user_agent = Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E; InfoPath.3), access_type = INTERNET_OPEN_TYPE_PRECONFIG

InternetConnectA	protocol = HTTP, server_name = api-content.dropbox.com, server_port = 443
HttpOpenRequestA	http_verb = GET, http_version = HTTP/1.1, target_resource = /1/files/auto/WmiApCom, accept_types = 0, flags = INTERNET_FLAG_PRAGMA_NOCACHE, INTERNET_FLAG_HYPERLINK, INTERNET_FLAG_SECURE, INTERNET_FLAG_NO_CACHE_WRITE, INTERNET_FLAG_RELOAD
HttpSentRequestA	headers = Authorization: Bearer sgKddaX_ntAAAAAAAAAADVYeex9PcoNuhGII10uLUhy-Kte7gEehQSxjYgRB2yWT, url = api-content.dropbox.com/1/files/auto/WmiApCom

Appendix B: Research Methodology

We think opaque research is nonsense...and frankly, anyone can do this analysis with the right tools, the right data, and the right approach. This section is intended not only to explain how we gathered and interpreted our own data, but also help aspiring researchers and analysts identify tools and methods that will help in conducting similar studies.

Gathering Malware Samples

Much of our research was initially sourced from public reports on malware and intrusion operations. Sometimes vendors purposefully provide reference hashes that are nowhere to be found on the internet, knowing you will be unable to get a sample for validation or research. When we're lucky and they do provide references to publicly accessible data, we often find that they provide as few references as possible in order to protect customer privacy and shelter their own intellectual property. Needless to say, it is often difficult for folks to substantiate vendor claims based only on public data. This is especially true for longitudinal and wide scope study of malware across regions, groups, and specific intrusion campaigns.

To fill in gaps of vendor reports, and to help validate public information, we used our data subscriptions to search for malware samples that use legitimate services as a means of C2.

In the section above detailing "Experimental Detection Methodologies" we suggested that you might be able to find backdoors using legit services C2 based purely on DNS calls in a sandboxing system. Following this assumption, we took to VirusTotal Intelligence (VT Intel) searches to enumerate samples that communicated with a few legit services.

Starting with Dropbox, we sought to identify the API endpoints by which samples may communicate. A note on the Dropbox website revealed the following information:

> "The current list of API hostnames is: api.dropboxapi.com, content.dropboxapi.com, and notify.dropboxapi.com. Previously, these hostnames were used: api.dropbox.com, api-content.dropbox.com, and api-notify.dropbox.com. These will continue to be supported for v1, but the dropboxapi.com hostnames are preferred."

Using the VT Intel search function, we searched for each of the above domains in the results in the VT sandbox run of any samples. Currently, the VT Intel search this only searches samples submitted in the last two months, so your results may vary.

- behavior:"api.dropboxapi.com" - 0 files found
- behavior:"content.dropboxapi.com" - 0 files found

- behavior:"notify.dropboxapi.com" - 0 files found
- behavior:"api.dropbox.com" - 21 files found
- behavior:"api-content.dropbox.com" - 3 files found
- behavior:"api-notify.dropbox.com" - 1 file found

(At the time of this writing (late 2017) some experimental features of the VirusTotal Graph allow users to search sample behavior and relationship data across all time, without being limited by the recency of submissions, for example: https://www.virustotal.com/graph/gace31ba244a724639ac1b125e909048236a5d2c52d930a6cac5f6c118d450a8a)

Then on further inspection of the files that connected to api-content.dropbox.com we see the context in which they communicate to this domain name.

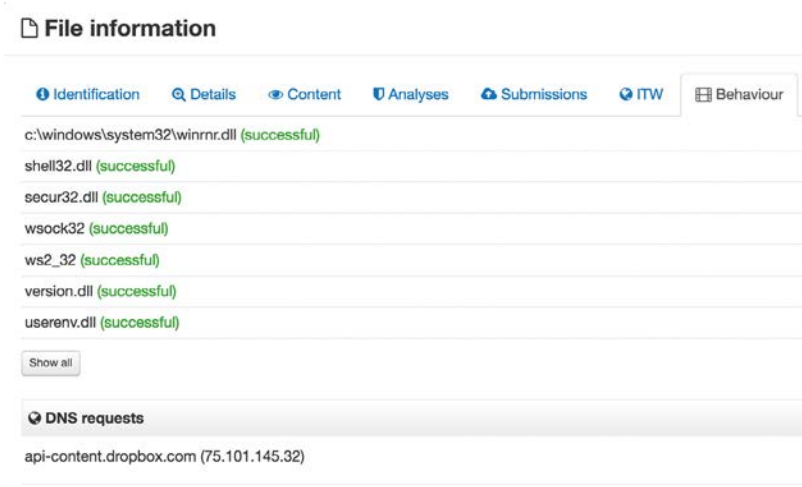


Figure 20. Segment of the VirusTotal Intelligence file "behavior" tab, showing part of the sandbox output.

We arrive at a list of files that each make DNS requests api-content.dropbox.com for further analysis:

- e1bb763cdee50091b39e26c9ef4252a7
- 13e59631c7cde1cff83e38161fd7e17f
- aa4f9ab5bd0249063b65c2955ad5d480

Similarly, we may be able to use VT Intel searches to enumerate samples that call out to other legitimate services simply by querying for the domains in the output of the sandbox behavior. Below, we query a few legit services to illustrate the scope of files that call out to these services during execution.

Haters, hold your horses. Look, we get it. Just because a sample calls out to a website during execution doesn't mean it uses that service for anything evil, or even if they are

evil it doesn't mean they're using legit services for C2. Obviously, not all of these samples are going to be malicious — though some of them are definitely super evil — but at the very least we believe that you can use simple queries like this to identify sample sets that serve as leads for additional processing, deeper analysis or rote blacklisting.

Example VT Intel Query	**Files Found (within 2 months of Nov 8, 2017)**
behavior:"docs.google.com"	155
behavior:"script.google.com"	11
behavior:"translate.google.com"	49
behavior:"storage.googleapis.com"	94
behavior:"smtp.gmail.com"	431
behavior:"blogger.com"	71
behavior:"blogspot.com"	503
behavior:"onedrive.com"	34
behavior:"onedrive.live.com"	10
behavior:"livefilestore.com"	3
behavior:"dropbox.com"	387
behavior:"twitter.com"	1000+
behavior:"api.twitter.com"	33
behavior:"github.com"	1000+
behavior:"raw.githubusercontent.com"	176
behavior:"qzone.qq.com"	963

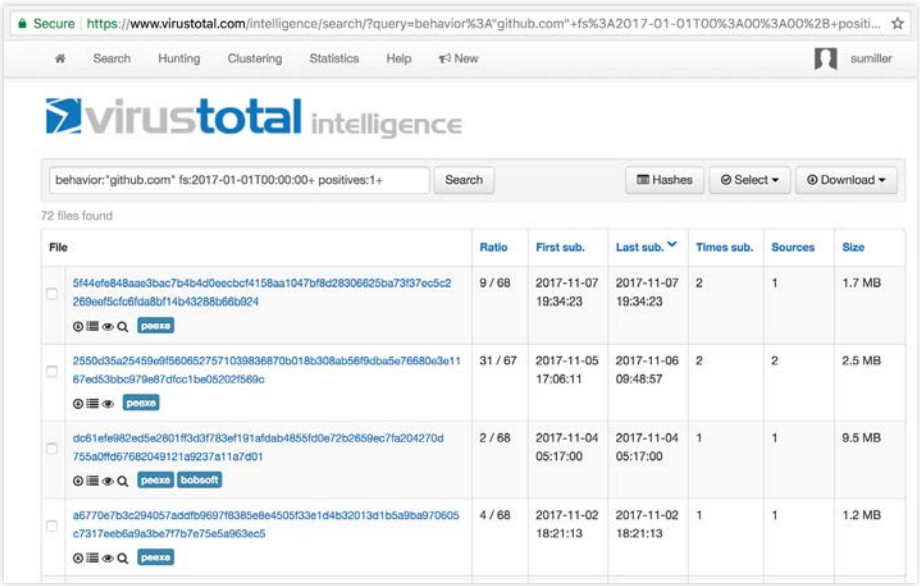


Figure 21. Sample VT Intel Query and Results

Again, we're just using these service providers as examples because they are amongst the most prolific on the internet and we have definitive examples of malware use to each of them. These companies and services are not inherently evil, though they are prime targets for abuse by malicious actors. Our friends at Google, Dropbox, Microsoft, GitHub, Twitter et al take abuse seriously and work hard to identify and ban malicious users with extreme prejudice. The abuse of these services, while representing challenges to providers and defenders, does not represent a risk to customers or average users of these services.

Performing Malware Function Analysis

After gathering a collection of samples that call out for legit services C2, we used additional malware analysis technologies and software tools to help us better understand how these malware ecosystems communicate with the legitimate services.

Though there are many malware analysis systems, for this study we used our trial subscription to VMRay [12] Cloud, and executed our malware samples in their agentless, hypervisor-based sandboxing system. This allowed us to observe and fully execute samples that are "sandbox aware," meaning they have anti-analysis logic that

would normally stop execution if the malware is run in a standard malware analysis virtual machine. Furthermore, VMRay's incredible function logging ability gives us an unrivaled look at API calls and program behavior while the malware is executing — this granular detail is key to understanding how the backdoors operate and extracting even the most covert indicators and tool marks.

Most of our samples possessed anti-analysis checks, which meant that they did not execute properly in our other sandbox technologies. Furthermore, many of our samples connected to legitimate services accounts were no longer active, which meant that the backdoors would either sleep or stop execution altogether. However, when submitting our samples to VMRay, we were able to not only execute "sandbox aware" samples, but also to use the function logs to pull out the network indicators of the exact legit services accounts and URIs that the samples were using for C2.

Please note that we are neither a VMRay customer nor partner. We merely think their product is incredibly impressive, and we're highlighting it here because it was integral to our research.

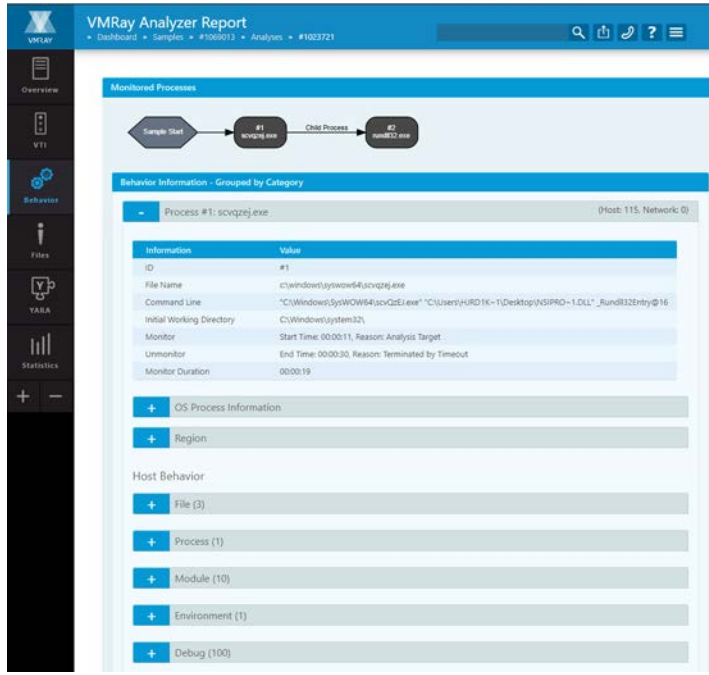
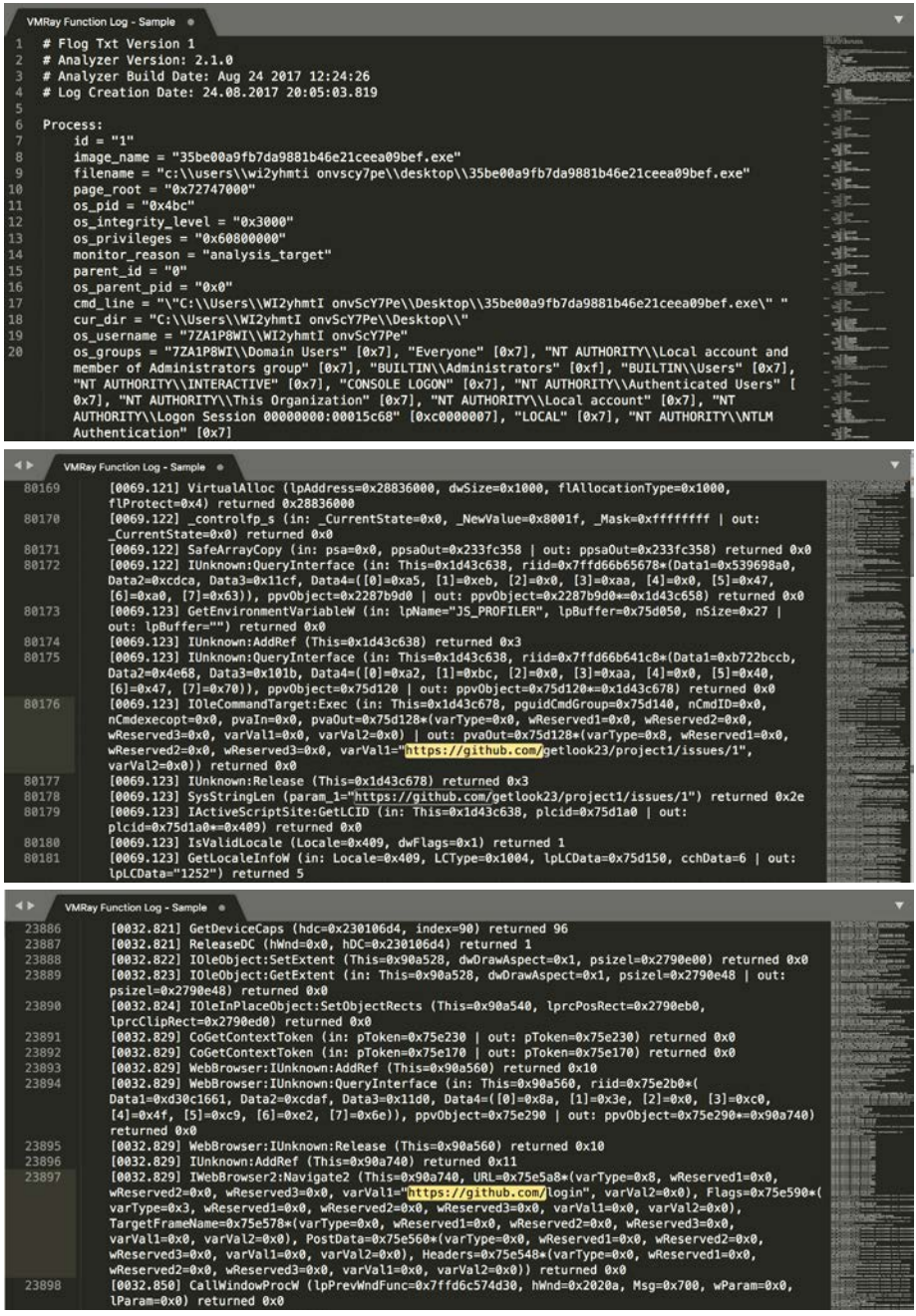


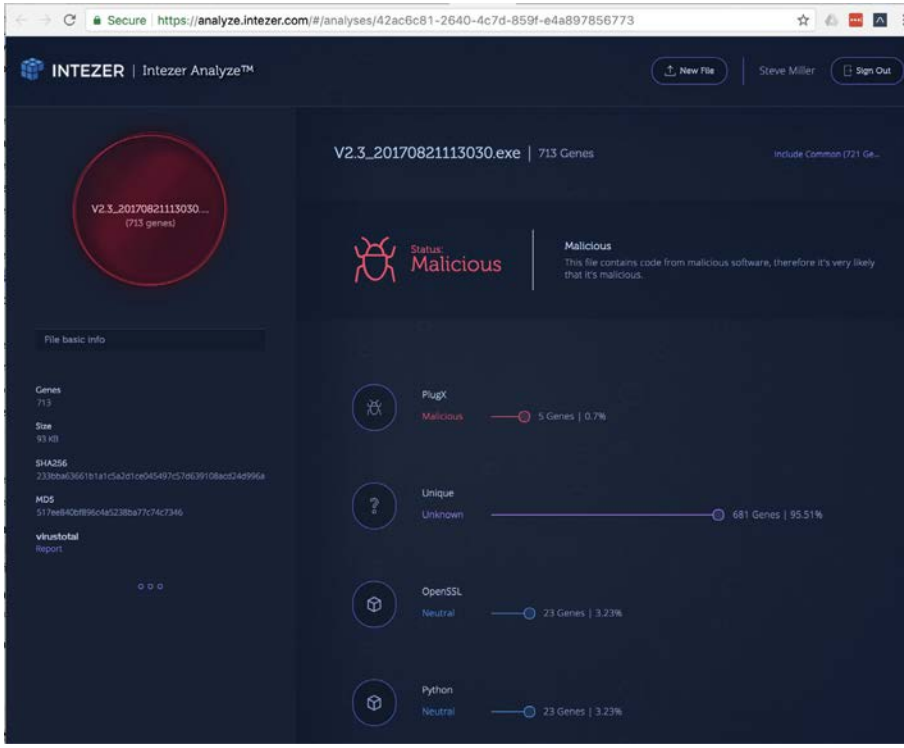
Figure 22. Sample VMRay Analyzer Report detailing malware behavior. We used these summaries to determine the nature of our submitted files and to identify network callouts.



Figures 23-25. Sample VMRay process function logs for RAINYDROP. We used the function logs to identify exactly which legitimate services accounts (or repositories) were used and how they were invoked.

We also used Intezer for code similarity and morphology matching analysis. This did not yield any significant results as many of our malware samples were using packers and file types that were not currently supported by Intezer. Still, we believe this is a crucial facet of studying malware, as identifying relationships and similarities between code families is important for understanding how families and groups evolve in capabilities.

If you have access to Intezer Analyze community edition, you can see gene matching on a previously unknown SOGU sample here. This SOGU sample was built by our team and as seen in this example, the Intezer analysis shows 5 gene matches on "PlugX".



Performing Network Traffic Analysis

After understanding the Windows API functions and the C2 configurations, we wanted to take a closer look at what the underlying HTTP traffic would be inside of the HTTPS connections to the legitimate services. To aid in this task we look to our own analysis virtual machine and FireEye's FakeNet NG [13].

FakeNet allows us to "intercept and redirect all or specific network traffic while

simulating legitimate network services." In this case, it simulates an HTTPS server and tricks the malware into believing it has an encrypted connection to the desired network service. This encouraged the malware to communicate and allowed us to gather a packet capture of C2 network traffic. The process is simple: set up your malware virtual machine [14]; run FakeNet; execute malware; pull pcap; sit back and revel in the miracle of modern malware technology.

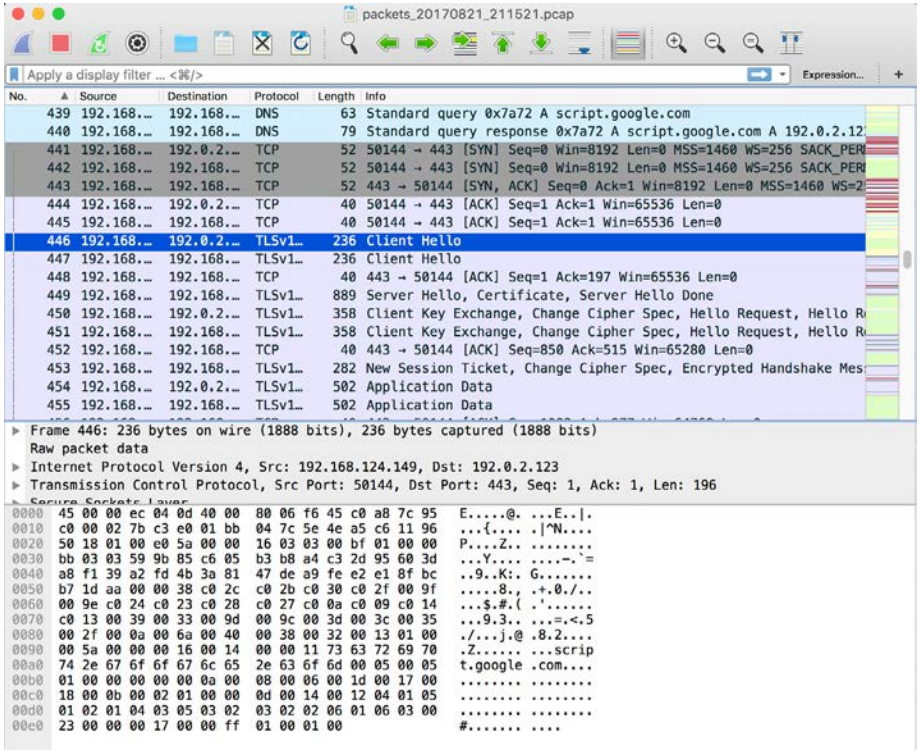


Figure 26. Packet capture showing encrypted C2 to Google Script.

Once we had our encrypted malware traffic pcap we used FakeNet's private key and Wireshark's SSL decryption setup to help us investigate the underlying traffic.

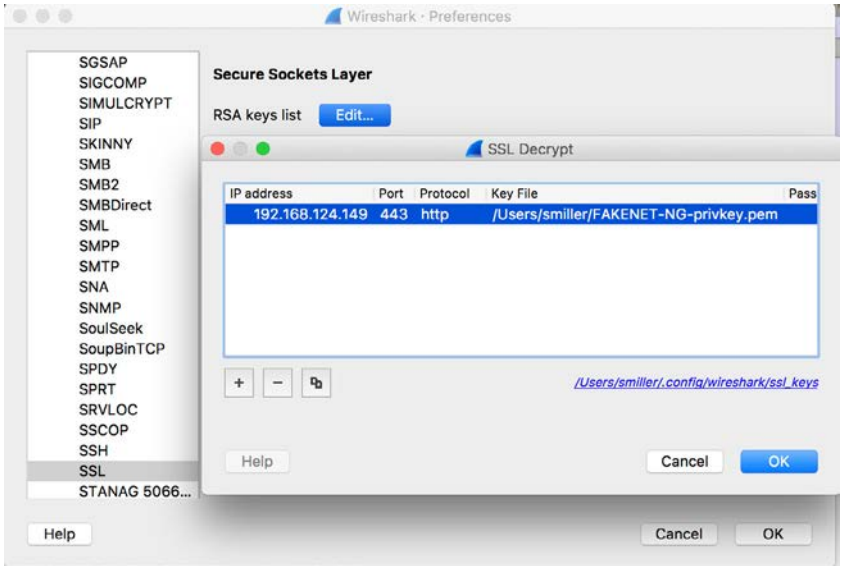


Figure 27. Wireshark preferences window showing the SSL Decrypt setup.

Figure 28. Following the decrypted SSL stream showing C2 to Google Docs.

Follow-on Analysis

There are several reasons to perform malware behavior and traffic analysis. Analysis can help with code classification, indicator extraction, attribution and more. Once you fully understand how these pieces of malware work, you can use what you've learned to assess your detection posture.

- Will the malware process behavior trip any methodology triggers on my endpoint agent?
- Will the malware be detected by my network appliance or network detection rules?
- Does it circumvent all detection tech and do I need to simply blacklist this MD5?
- Are there any special analytics that I can run to ensure that I will catch this family of malware or this C2 technique in the future?

These are the questions that must be asked, and these are the questions that can only be answered through careful, comprehensive analysis of malware samples.

Appendix C: Reference Data and Links

Welcome to the 21st century internet, where we paste links instead of MLA citations. The internet is a growing, changing place, so at some point these links might break, in which case you'll have to get googling. Sorry not sorry.

Referenced Works

1. https://github.com/tennc/webshell/blob/master/net-friend/aspx/aspxspy.aspx
2. https://www.arbornetworks.com/blog/asert/twitter-based-botnet-command-channel/
3. https://blog.trendmicro.com/trendlabs-security-intelligence/winnti-abuses-github/
4. https://www.fireeye.com/blog/threat-research/2014/08/operation-poisoned-hurricane.html
5. https://www.blackhat.com/docs/asia-14/materials/Haruyama/Asia-14-Haruyama-I-Know-You-Want-Me-Unplugging-PlugX.pdf
6. https://community.rsa.com/community/products/netwitness/blog/2015/05/19/wolves-among-us-abusing-trusted-providers-for-malware-operations
7. https://blogs.forcepoint.com/security-labs/carbanak-group-uses-google-malware-command-and-control
8. http://oalabs.openanalysis.net/2016/09/18/the-case-of-getlook23-using-github-issues-as-a-c2/
9. https://www.f-secure.com/documents/996508/1030745/dukes_whitepaper.pdf
10. https://www2.fireeye.com/rs/848-DID-242/images/rpt-apt29-hammertoss.pdf
11. https://www.fireeye.com/blog/threat-research/2015/11/china-based-threat.html
12. https://www.vmray.com/
13. https://github.com/fireeye/flare-fakenet-ng
14. https://zeltser.com/build-malware-analysis-toolkit/

Other Readings and Projects

1. BELLHOP and FIN7 - The Magnificent FIN7: Revealing a Cybercriminal Threat Group - https://www.infosecurityeurope.com/__novadocuments/367989?v=636338290033030000
2. BLACKCOFFEE and APT17 - Revealing Attack Operations Targeting Japan - https://www.jpcert.or.jp/present/2015/20151028_codeblue_apt-en.pdf
3. DropSmack and Dropbox C2 - https://media.blackhat.com/eu-13/briefings/Williams/bh-eu-13-dropsmack-jwilliams-wp.pdf
4. Dropbox C2 for Empire - https://bneg.io/2017/05/13/dropbox-for-the-empire/
5. Dropbox C2 for Powershell and DBC2 - https://vimeo.com/195596062 and https://truneski.github.io/blog/2017/03/03/dropbox-command-and-control-over-

powershell-with-invoke-dbc2/

6. Social Media C2 - https://zeltser.com/bots-command-and-control-via-social-media/

Exemplar Malware Samples

We're definitely not here to give you "IOCs" for purposes of detection. IOCs are dead. That said, here are some reference hashes if you want to take a look at some malware and examine the related network traffic. Don't have access to VirusTotal Intelligence? Don't worry, we got you. All of these samples are zipped up for your convenience here ***(pw: infected)***.

Family Relation	MD5 (All of which are in VT)	Note
BLACKCOFFEE	4c21336dad66ebed2f7ee45d41e6cada	https://community.rsa.com/community/products/netwitness/blog/2015/05/19/wolves-among-us-abusing-trusted-providers-for-malware-operations https://www.fireeye.com/blog/threat-research/2015/05/hiding_in_plain_sigh.html
SOGU	029c8f56dd89ceeaf928c3148d13eba7	https://www.fireeye.com/blog/threat-research/2014/08/operation-poisoned-hurricane.html https://blog.trendmicro.com/trendlabs-security-intelligence/winnti-abuses-github/ https://www.blackhat.com/docs/asia-14/materials/Haruyama/Asia-14-Haruyama-I-Know-You-Want-Me-Unplugging-PlugX.pdf
WHISTLETIP	;(	

BARLAIY	2904c1876f5727c256a07e9bf0140086 8ab6166526048c935765ecdc714febdd 830a09ff05eac9a5f42897ba5176a36a dc297a03e39f0479666b37662ad274d3	https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Trojan:Win32/Barlaiy.A!dha
BELLHOP	4b783bd0bd7fcf880ca75359d9fc4da6 ae8404ad422e92b1be7561c418c35fb7 af53db730732aa7db5fdd45ebba34b94 280d328f24fb611234b7d2d15f85b970	https://www.infosecurityeurope.com/__novadocuments/367989?v=636338290033030000 https://blogs.forcepoint.com/security-labs/carbanak-group-uses-google-malware-command-and-control
RAINYDROP	35be00a9fb7da9881b46e21ceea09bef 4d1ff8d0c377470b558b13eb60b56a89 ad10be0ecd27d6ff734c541a79168ef4 c6fdf7134d7ef8a4f38e40938ad21f7d	http://oalabs.openanalysis.net/2016/09/18/the-case-of-getlook23-using-github-issues-as-a-c2/
HAMMERTOSS	d3109c83e07dd5d7fe032dc80c581d08	https://www2.fireeye.com/rs/848-DID-242/images/rpt-apt29-hammertoss.pdf https://www.f-secure.com/documents/996508/1030745/dukes_whitepaper.pdf
LOWBALL	d76261ba3b624933a6ebb5dd73758db4	https://www.fireeye.com/blog/threat-research/2015/11/china-based-threat.html

Services to Malware Family Mapping

There are just too many services, too many malware families, and too much diffuse public information to do a thorough literature review of legit services here and now. And it might be pointless to even try because this information is changing almost daily. For the time being, we can talk about the low hanging fruit and do a quick mapping of legit services to some of the more notable malware families abusing them for C2, each of which have public reference data. By no means is this intended to be complete, but rather to briefly substantiate our claims.

Service	Mal Fams or Associations	Notes
Twitter	HAMMERTOSS FLASHBACK	https://www2.fireeye.com/rs/848-DID-242/images/rpt-apt29-hammertoss.pdf https://www.welivesecurity.com/media_files/white-papers/osx_flashback.pdf https://www.arbornetworks.com/blog/asert/twitter-based-botnet-command-channel/
Dropbox	LOWBALL	https://www.fireeye.com/blog/threat-research/2015/11/china-based-threat.html
Github	BARLAIY, RAINYDROP, HAMMERTOSS	https://www2.fireeye.com/rs/848-DID-242/images/rpt-apt29-hammertoss.pdf
Amazon	Pentesters have used EC2 exhaustively for red teaming operations. Threat actors (using many of the same tools, such as Empire, Metasploit, and Cobalt Strike) have also shifted to using AWS, merely to host malicious services and binaries for attack operations.	https://www.theregister.co.uk/2014/01/16/amazon_cloud_security_nightmare/ https://blog.cobaltstrike.com/2014/09/09/infrastructure-for-ongoing-red-team-operations/ https://www.theregister.co.uk/2011/07/29/amazon_hosts_spyeye/ http://www.zdnet.com/article/zeus-crimeware-using-amazons-ec2-as-command-and-control-server/
Pastebin	SOGU, EMPIRE	https://www.sc0tfree.com/sc0tfree-blog/optimizing-rubber-ducky-attacks-with-empire-stagers
Microsoft TechNet	BLACKCOFFEE	https://www.fireeye.com/blog/threat-research/2015/05/hiding_in_plain_sigh.html
Microsoft Social	BARLAIY, BLACKCOFFEE	https://www.fireeye.com/blog/threat-research/2015/05/hiding_in_plain_sigh.html

Hotmail	MACROMAIL	https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/mandiant-apt1-report.pdf
OneDrive/ SkyDrive	CLOUDDUKE	https://labsblog.f-secure.com/2015/07/22/duke-apt-groups-latest-tools/ https://blog.malwarebytes.com/threat-analysis/2014/01/neutrino-delivers-fake-flash-malware-hosted-on-skydrive/ https://www.virustotal.com/intelligence/search/?query=952276eaabda56bdfc631a34f17bdd63
Quora	CONFUCIUS	https://researchcenter.paloaltonetworks.com/2016/09/unit42-confucius-says-malware-families-get-further-by-abusing-legitimate-websites/
Yahoo! Answers	CONFUCIUS	https://researchcenter.paloaltonetworks.com/2016/09/unit42-confucius-says-malware-families-get-further-by-abusing-legitimate-websites/
Yahoo! Mail	ICOSCRIPT	https://www.virusbulletin.com/virusbulletin/2014/08/icoscript-using-webmail-control-malware
Yahoo! Babelfish	XSLCMD	https://www.fireeye.com/blog/threat-research/2014/09/forced-to-adapt-xslcmd-backdoor-now-on-os-x.html
Google Docs	BELLHOP, MAKADOCS	https://blogs.forcepoint.com/security-labs/carbanak-group-uses-google-malware-command-and-control https://www.symantec.com/connect/blogs/malware-targeting-windows-8-uses-google-docs https://contagiodump.blogspot.com/2012/12/nov-2012-backdoorw32makadocs-sample.html#more https://www.symantec.com/security_response/writeup.jsp?docid=2012-111609-4148-99&tabid=2

Google Drive	GOOGLESOCKS	https://www.blackhillsinfosec.com/google-docs-becomes-google-socks-c2-over-google-drive/ https://github.com/lukebaggett/google_socks
Google Translate	XSLCMD	https://www.fireeye.com/blog/threat-research/2014/09/forced-to-adapt-xslcmd-backdoor-now-on-os-x.html
Google Code	SOGU, XSLCMD	https://www.fireeye.com/blog/threat-research/2014/08/operation-poisoned-hurricane.html https://www.fireeye.com/blog/threat-research/2014/09/forced-to-adapt-xslcmd-backdoor-now-on-os-x.html
Google Scripts	BELLHOP	https://blogs.forcepoint.com/security-labs/carbanak-group-uses-google-malware-command-and-control http://blog.talosintelligence.com/2017/09/fin7-stealer.html
Google Calendar	GCAL	https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/mandiant-apt1-report.pdf
Google Plus	BLACKENERGY2	https://securelist.com/be2-custom-plugins-router-abuse-and-target-profiles/67353/
Google Talk/XMPP	GLOOXMAIL	https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/mandiant-apt1-report.pdf
Google Groups	"Shadows in the Cloud"	http://www.nartv.org/mirror/shadows-in-the-cloud.pdf
Google Sites	"Shadows in the Cloud"	http://www.nartv.org/mirror/shadows-in-the-cloud.pdf
Google AppEngine/Cloud	STALEMATE	https://www.proofpoint.com/us/exploring-bergard-old-malware-new-tricks
Gmail	GCAT	https://github.com/byt3bl33d3r/gcat
YouTube	JANICAB	https://www.f-secure.com/weblog/archives/00002576.html

Blogger/ Blogspot	XSLCMD	https://www.fireeye.com/blog/threat-research/2014/09/forced-to-adapt-xslcmd-backdoor-now-on-os-x.html
Facebook	There are many examples of Facebook abuse in malware schemas. One of the notable examples, which is from a leaked U.S. Government presentation, discusses Facebook C2 by an unnamed malware family associated with the "BYZANTINE" actor.	Click warning. The following resource has a TS // COMINT // FIVE EYES classification: https://www.eff.org/files/2015/02/03/20150117-spiegel-byzantine_hades_-_nsa_research_on_targets_of_chinese_network_exploitation_tools.pdf
Baidu Blogs	"Shadows in the Cloud"	http://www.nartv.org/mirror/shadows-in-the-cloud.pdf
VKontake	FOXY	http://www.securityweek.com/new-%E2%80%9Cfoxy%E2%80%9D-malware-uses-clever-techniques-stay-hidden

Appendix D: FireEye's "Evolution of Malware"

We covered the gist of the "data points on the rise of legit services" in the main body of this report — that sums up what you really need to know at the highest level. This appendix exists to document the nitty gritty details of this study, so you can get a better understanding of the previous work on the topic.

Note that here we are summarizing rhetoric and data is originally sourced from Mandiant/FireEye all of which was presented publicly in 2017. All credit for the following information goes to the founding members of FireEye's Advanced Practices Team: Steve Miller, Ben Withnell, Matt Berninger, Steve Stone, and Nicole Oppenheim.

Background of the Study

Of the thousands of transport- and application-layer protocols, which ones will provide the most visibility into malware network traffic? Of the mechanisms that malware might use to survive a reboot, which ones will give me the most bang for buck when the data is collected and analyzed at enterprise scale?

From security analyst to CISO, we need to ask ourselves questions about how malware affects our defensive posture. What network protocols are being used by malware in what amounts? And where are my blind spots in endpoint and network data with respect to finding malware? And most importantly, what are the most important investments I can make to maximize detection of malware and help reduce my risk of an undetected intrusion? These are difficult questions to answer, so you take to the news to help identify trends.

However, if you look at the news of recent exploits, C2 protocols and other "new hotness" of cyber attacks, you may realize that it is hard to differentiate between events of fleeting importance and real trends that affect defenders on the ground in the long term. Unfortunately, folks end up using anecdotal stories to make security decisions, because "breaking" news and topical vendor reports are often focused on the specific incidents and "attention grabbing" events (read: fearmongering) without showing how these events fit into the bigger picture. Where are the scientific, longitudinal studies that show how specific incidents fit into overall technical trends in how malware operates?

The "Evolution of Malware" study emerged out of the need to make educated security decisions for how we collect data and how we detect malware across different data types. At the time we put this study together, nobody was able to say a certain "percent of malware uses HTTP-like protocols for C2", and say if that usage was rising or falling over the last decade. We wanted to see if, when, and how malware developers (and attack groups) make significant changes in how their backdoors operate. We wanted to know the year-over-year prevalence of C2 protocols for malware, and then use that knowledge to assess our defensive posture (and vendors), and make better decisions on how to invest our time and money to detect these things.

In late 2016, FireEye's Advanced Practices Team (featuring yours truly) attempted to measure the prevalence of legit services C2 by studying the Mandiant malware corpus. This rich sample set of roughly ten thousand samples spanned a decade of global incident response and intelligence collection, each sample having undergone some level of manual reverse engineering for code classification and capability. Using this sample set, we attempted to measure year-over-year statistics to tease out trends for samples using legit services C2.

Sample Set Description and Biases

Foremost, it is important to remind everyone that this is not a bona fide scientific

study. We did not conduct a true experiment and apply the scientific process, so you will not be reading about our sampling technique, confidence intervals, or statistical significance for any of our results. Again, because we wanted to get a basic "wag" at malware trends, and because we didn't want to spend all damn year doing it, we simply took the best data set we could get and calculated some quick and dirty percentages of malware capabilities.

What was the initial data set?

The initial data set is from Mandiant's malware analysis corpus. While the Mandiant intelligence database holds bajillions of samples, there is a subset of these samples that through the years has undergone full reverse engineering, which helps classify samples into code families and provides a full description of how each sample operates with details on both the endpoint and network capabilities.

The data set represents samples collected from almost all thinkable sources, including open source repositories, paid feeds, "special access sensors," partners, friends, community tips, and global incident response engagements from 2007 to 2016. The data set for our purposes was roughly ten thousand samples in total, representing about seven thousand pieces of malware and accompanying files such as legitimate DLLs, shellcode, and scripts that would somehow aid in the malware execution.

How did you reduce, adjust, and interpret the data set?

For purposes of this study, we selected only the standalone, fully functional (non-corrupted) samples that were successfully identified as Portable Executable files by the Mandiant Threat Analyzer (MTA), which is Mandiant's (hella dope) internal hypervisor-based malware analysis technology. Our final sample set was comprised of approximately five thousand Windows PEs and DLLs.

Because we wanted to get a sense of malware development and use over time, we wanted to take a best guess of when our samples were developed, created, or deployed for malicious use in the wild. So we started with compile time, and corrected for default or obviously falsified compile times with Mandiant or VirusTotal's "first submitted" or "first seen in the wild" timestamps. For example, if a compile year read 1899, 1970, 1992 or 2078 we would adjust to the earliest of Mandiant or VirusTotal observed years. This obviously adds yet another layer of abstraction to the data, and this introduces chance for error, but this was the best way to estimate when malware with a particular capability was "created", "deployed" or otherwise "present" somewhere in the world.
Next, we reviewed each of the approximately five thousand samples by reading the reverse engineering reports and manually documenting each sample's communications capabilities. We then documented the samples by year into a spreadsheet to calculate the annual percentage presence, to finally arrive at a graph of "marketshare" for each communications capability.

What are the biases of the data set?

We would like to be objective as possible when understanding if, where, when

and how, malware authors are developing and changing malware capabilities, but it is difficult to do with the layers of artifice that lead to biases in the data set. We don't believe these biases are a deal breaker for the study, but they must be kept in mind when thinking about if and how trends in the data set apply to you.

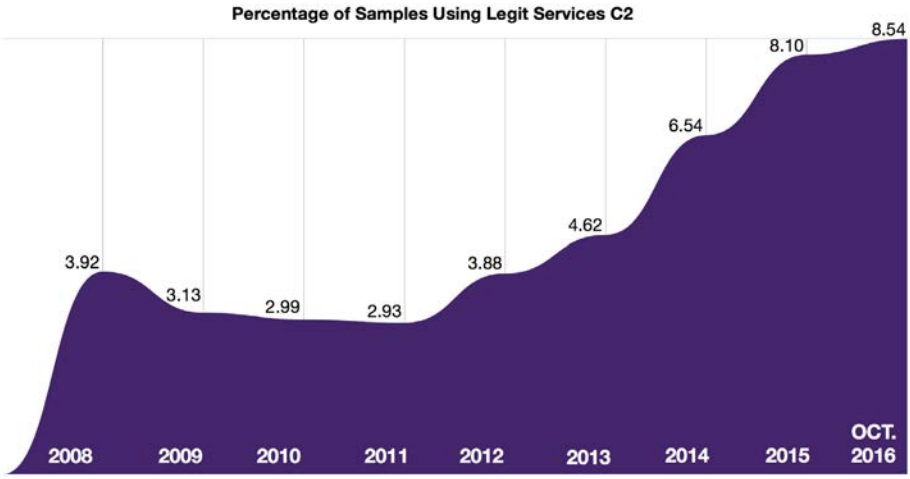
1. The data is biased towards Windows malware.
2. The data is biased towards malware that inherently possesses automatic communications capabilities.
3. The data is biased towards malware positively sourced from espionage- and financially-motivated intrusions, which represent the bulk of Mandiant's incident response engagements. Many of these samples are attributed to known APT and FIN groups, and not samples that are simply "found" without reference to how they are used. The data is biased against malware used for "proof of concept" or any other demonstratively academic purpose.
4. The data is biased towards malware that was discovered in Global 2000 companies, which covers much of the Mandiant consulting customer base.
5. The data is biased towards malware that was new (previously unknown) or most interesting to Mandiant's consultants and intelligence analysts at the time of discovery.
6. The data is biased towards malware that was difficult to analyze at the time of discovery, thus requiring the assistance of our friendly neighborhood malware analysts. For example, a sample that was obfuscated using custom string tables required thorough reverse engineering to identify malware family and functionality, whereas a sample of common AutoIt malware compiled with Aut2Exe required little analysis whatsoever because it is easily decompiled to plaintext with software tools.
7. The data is biased towards malware that was discovered using Mandiant's compromise assessment, incident response, and hunting methodologies. Using these specific methodologies over the years may create a positive feedback loop towards particular malware types and associated threat groups.
8. The data is biased against second-stage malware which must be manually installed or configured by attackers at the command line. Second-stage malware may include a myriad of communications methods that while possible in the malware are not part of its automatic execution instructions.
9. The data is biased against malware that has unreferenced capabilities which may represent vestigial routines, experimental functions, or other artifacts that while inside of the malware are not part of the sample's automatic behavior. These malware families might have more capabilities in the future, or require additional files to access any unlinked functions. For the purpose of this study we excluded samples with unreferenced capabilities.

So with the selection bias, expectation bias, frequency illusion and all that — take all our numbers with a grain of salt, and keep in mind that we're using these measurements

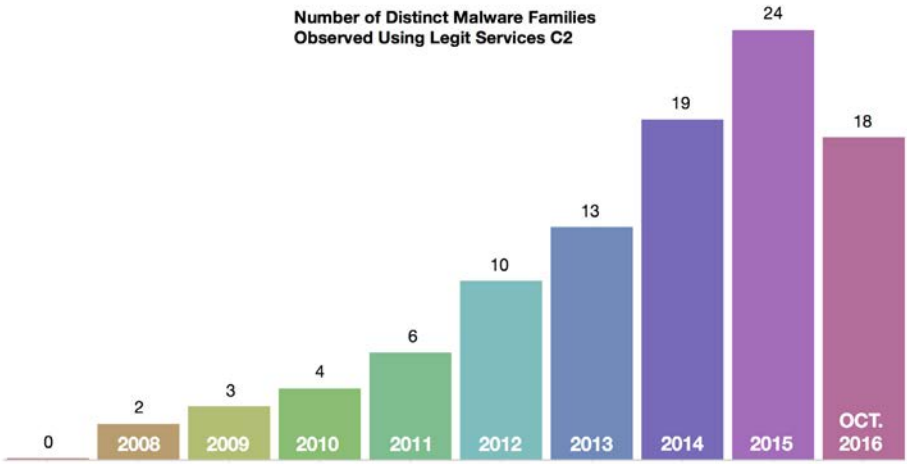
as a "wag" to help us understand the long term changes in malware communications methods. Furthermore, remember that we're using this study to show that some real science needs to be done in this area. We're hoping that other vendors and security researchers will see the potential for this kind of study and contribute to collecting a much larger data set that can be randomly sampled and scientifically assessed.

Where are the pretty pictures?

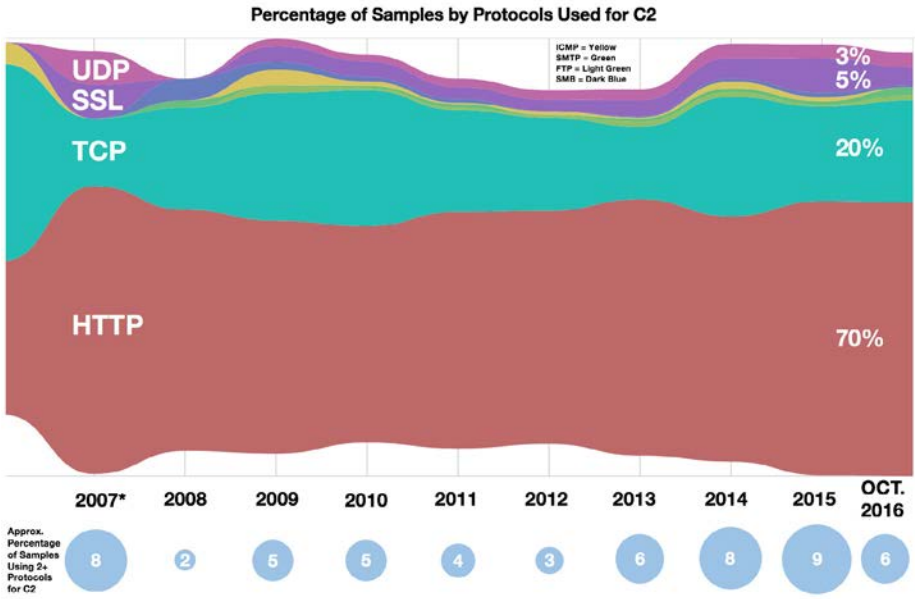
The following images show the year-over-year trends that we identified for usage of legit services C2, use of various network protocols for C2 communications.



Marketshare of Malware Samples Using Legit Services C2



Raw Number of Distinct/Unique Malware Families Using Legit Services C2



Marketshare of Malware Samples by Various Internet Protocols (including legit services C2)

What do all the words mean?

Malware developers don't play nice with our beloved internet. Malware C2 does not really adhere to the the TCP/IP or OSI/ISO models of network communications. Accordingly, we generally refrain from discussing things like RFCs and "layers" when discussing what types of protocols malware use for C2. Instead, we will focus on how the C2 appears practically when looking at network packet traffic. Think of how it will appear in "Wireshark" and other tools.

Almost all malware C2 protocols are "custom" network protocols. Although they attempt to mimic legitimate protocols, rarely do they ever completely conform to the standards by which legitimate (non-malicious) applications abide.

When we say TCP protocol, we mean to indicate that the malware uses a completely custom binary protocol that is carried over TCP, and without special decoders, at the highest recognizable level the traffic appears to be nothing more than abstract TCP payload data. If you open up a pcap of custom binary TCP traffic in Wireshark, it will appear to be raw TCP data. Wireshark will not have dissectors for custom malware protocols. Similarly, when we say UDP or ICMP, we are saying that the malware is using a custom binary protocol carried within these protocols.

When we say a sample uses HTTP, we are indicating that the malware is most likely using high-level APIs for HTTP functions, and the subsequent C2 protocol appears very similar to legitimate HTTP traffic. Of course, we all know that malware authors do not design C2 protocols by the RFC for any given protocol. Still, the network traffic for these samples would appear as HTTP, and would likely get parsed by log devices and other security appliances and software tools as HTTP traffic.

When we say a sample uses SSL, we mean two things. First, we include those malware samples that are using an HTTP protocol for C2, but uses special API functions to enforce security in the connection. Second, we include those malware samples using TCP protocols in conjunction with built-in SSL functionality, often using borrowed DLLs from legitimate or open source software to host their own SSL server and encapsulate TCP C2 data.

When we say FTP, SMB, SMTP, et cetera, we are saying that the malware uses APIs or hard-coded functions and is creating network traffic that appears to be any of these protocols, but is in fact being used for the malware C2. Some malware may use nothing but legitimate functions and API calls, allowing the malware to communicate through intermediary devices without ever being noticed. For example, if the malware is using SMTP to communicate with the Gmail API, even though it is a custom protocol per se, it might be indistinguishable from a normal mail client. For what it's worth, all of these protocols were used by <1% of annual samples, producing lines so small they can barely be seen on the chart.

If you could do the math percentages, you might be a bit perplexed at why some of the years add up to more than one hundred percent. This is because there were a significant number of samples each year that were capable of using two or more protocols to communicate. Interestingly, this is especially applicable to those malware families that use legit services for C2 DDR, as many of these families switch from HTTP and/or HTTPS to custom TCP protocols. This reflects not the capabilities of malware overall, but for those families that are immensely configurable, the samples would be created with these functionalities on a case-by-case basis.

Also, 2007 was not a great year for the data. We kept the scope of the study for a decade for posterity (and also it sounds cooler when you say "decade"). But, frankly, the 2007 data should just be wiped from the study because there were only a handful of samples from a few malware families and threat groups, dramatically skewing any capabilities that were counted for the year.

What am I supposed to take away from all of this?

Studying malware capabilities at a large scale is difficult because it is hard to get unbiased data and it is even harder to automatically extract detailed capabilities from malware samples. This study was a costly exercise without particularly scientific results. Still, we believe that the research serves as a prototype for how we want to examine and make sense of trends in malware evolution. We can use things like the "marketshares"

of communications protocols to educate ourselves and make conscious decisions on what we're missing and what we're not even trying to detect or "hunt" for.

For example, if only 5+% of malware network traffic is encrypted, is it worth your time to pursue SSL decryption or other methods of finding malicious traffic in those network streams? Take a look at your vendors, your security technologies, your IT infrastructure, and use your knowledge of protocol marketshare figure out your blind spots. Will your "next-generation firewall" detect it? If it is not caught on the network, will your "continuous monitoring" endpoint agent detect it? Will your "intelligence-powered" security products detect it? Maybe, maybe not. You better find out.

The study was too biased and these stats are too squishy for us to start shouting confident conclusions from the rooftops. But we think the following assumptions are safe:

1. Use of legit services C2 is on the rise and that poses significant problems for detection systems and defenders worldwide.
2. The majority of Windows malware uses plaintext HTTP protocols for C2, but there is a significant amount malware using custom binary and encrypted protocols producing data that is harder to collect and "hunt" through if you wish to find malicious things that your antivirus and vintage security systems miss.
3. You should try to use a broader, data-driven understanding of malware capabilities to test your vendors. Think about what they can and cannot do, and scrutinize their claims.