2
Attacker Techniques and Motivations

2.1 How Hackers Cover Their Tracks (Antiforensics)

2.1.1 How and Why Attackers Use Proxies

Masking one's IP address is a standard practice when conducting illicit activities. A well-configured proxy provides robust anonymity and does not log activity, thereby frustrating law enforcement efforts to identify the original location of the person(s) involved.

A proxy allows actors to send network traffic through another computer, which satisfies requests and returns the result. Students or employees can use proxies to communicate with blocked services such as Internet Relay Chat (IRC) and instant messaging, or to browse websites that administrators block. Attackers also use proxies because Internet Protocol (IP) addresses are traceable, and they do not want to reveal their true locations. As one example, iDefense wrote about the fast-flux architecture (ID# 484463), which uses a proxy infrastructure to satisfy requests. Proxies are also a common source of spam e-mail messages, which use open relays (a simple mail transfer protocol [SMTP] proxy).

Proxies are useful to attackers in many ways. Most attackers use proxies to hide their IP address and, therefore, their true physical location. In this way, attackers can conduct fraudulent financial transactions, launch attacks, or perform other actions with little risk. While law enforcement can visit a physical location identified by an IP address, attackers that use one (or multiple) proxies across country boundaries are more difficult to locate (see Exhibit 2-1). The endpoint can only view the last proxy with which it is directly communicating and not any of the intermediary proxies or the original location.

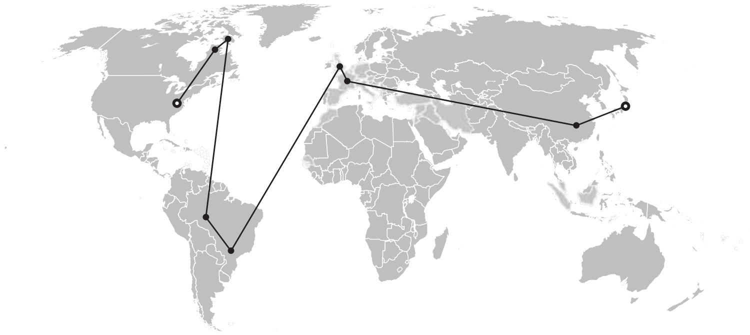


Exhibit 2-1 Multiple proxies make identifying the true source of an attack difficult.

Proxies provide attackers with a way to lower their risks of investigator identification of their true IP address. In the hypothetical attack displayed in Exhibit 2-1, the victim's log file contains only one of the many IP addresses that investigators need to locate the attacker.

Attackers operate free proxies or alter a victim's proxy settings because proxies can serve as a monitoring tool. AnonProxy is one example of a malicious proxy that its authors designed to monitor users and steal information such as social-networking passwords.[1] Since a proxy relays traffic, it also has the ability to log and alter sensitive pages or information. Attackers must either convince users or install malicious code to modify proxy settings themselves.

Malicious code authors also install local proxies. By altering the host's file or browser configuration to use the proxy, the attacker redirects requests and captures confidential information. Some banking Trojans give attackers the ability to proxy requests through the victim's browser because conducting fraud from a legitimate user's IP address is less suspicious. Local proxies are more difficult to identify because the local proxy does not open any network ports and scanning the system will reveal no changes.

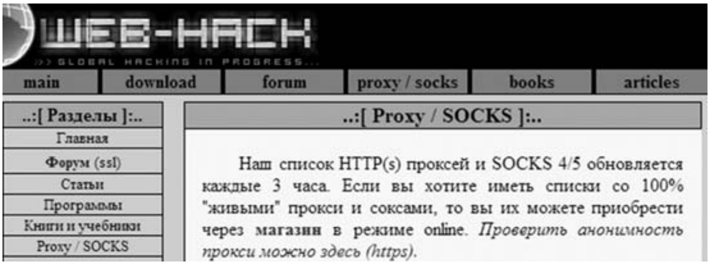
2.1.1.1 Types of Proxies Proxies are so common that many attackers scan the Internet for common listening proxy ports. The most common proxies listen on TCP port 80 (HTTP proxies), 8000, 8081, 443, 1080 (SOCKS Proxy), and 3128 (Squid Proxy), and some also handle User Datagram Protocol (UDP). Attackers who install custom proxies often do not use standard ports but instead use random high ports. Some lightweight proxies are written in scripting languages, which run with an HTTP server and are easier for attackers to modify. Application proxies require configuration. Some applications either do not operate correctly through proxy services because the proxy server removes necessary information or cannot satisfy the request. Some services like The Onion Router (Tor)[2] also give users the ability to proxy traffic and hide their original location from victims.

A virtual private network (VPN) acts as a more versatile proxy and supports more security features. Instead of configuring the application to use a proxy, users can tunnel all traffic through the VPN. VPN services usually support strong authentication and are less likely to leak information that could identify the user of a proxy.

Attackers commonly use free or commercial proxies (e.g., SOCKS and VPN) that operators advertise on hacking forums. Attackers may prefer these services to public proxies because they advertise anonymity and claim they do not keep logs, unlike Tor, where community operators can monitor traffic going through an exit node that it controls. Proxy services that keep logs are a danger to attackers who use these services for conducting fraud and can lead to their arrests. Some commercial VPN and SOCKS proxy services include

- hxxp://secretsline.net
- hxxp://vpn-secure.net
- hxxp://thesafety.us
- hxxp://5socks.net
- hxxp://vpn-service.us
- hxxp://vip72.com
- hxxps://www.cryptovpn.com
- hxxp://www.vipvpn.com
- hxxp://openvpn.ru

Another example of such a service from web-hack.ru shows free and commercial proxies that are available (see Exhibit 2-2). Translated from Russian, these free Proxy and SOCKS services are updated every three hours; users can also purchase proxy access through the store. Attackers may prefer proxy services advertised on hacking forums because they are less responsive to abuse requests. For example, commercial proxy services like FindNot keep logs of their



Наш список HTTP(s) проксей и SOCKS 4/5 обновляется каждые 3 часа. Если вы хотите иметь списки со 100% "живыми" прокси и соксами, то вы их можете приобрести через магазин в режиме online. *Проверить анонимность прокси можно здесь (https).*

Exhibit 2-2 Free and commercial proxies available from web-hack.ru.

users for a maximum of five days to protect the system from being used for abusive purposes, while many of those services advertised on hacking forums do not keep any logs. Operating proxy services is not illegal because it has legitimate purposes related to anonymity for users; however, some commercial proxy services are more willing to respond to abuse than others.

2.1.1.2 Detecting the Use of Proxies Detecting proxies is difficult and not always reliable. Since many malicious code authors install custom proxies and use encrypted or custom protocols, it is very difficult to detect all proxies. There are techniques to detect common proxies, but such techniques are unlikely to be effective against attackers who use proxies aggressively.

Port scanning on corporate networks can identify proxies that listen on default ports. Organizations should also monitor changes to proxy configuration because such changes could indicate that an attacker compromised a host. The registry key at HKCU\Software\Microsoft\Windows\CurrentVersion\InternetSettings, ProxyServer, controls the proxy settings for Internet Explorer. To detect proxies on the network with intrusion detection systems (IDSs), organizations may use proxy rules available from emergingthreats.net.[3] The domain name system blacklist (DNSBL) is one example of a blacklist that allows administrators to block certain proxies.[4]

Certain proxies do not proxy all traffic. For instance, a Web application can force users to perform unique DNS requests with subdomains (see Exhibit 2-3). The application links the DNS request to the user's IP address and verifies that the HTTP request originates from the same IP address. If they are not the same, indicating the use of a proxy, the application can determine that the proxy IP address

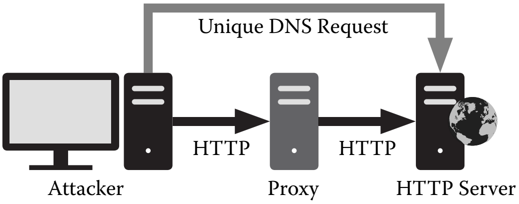


Exhibit 2-3 Certain proxy protocols may provide a way to identify the user of a proxy.

made the HTTP request and that the user's actual IP address made the DNS request. Similarly, some Web plug-ins may query the local information rather than using the proxy address. As an example, decloak.net is a Metasploit project that uses the following application plug-ins to determine the true IP address of a proxy user:

- Word
- Java
- Flash
- QuickTime
- iTunes

Metasploit has even provided an application programming interface (API) for website owners to determine the true IP addresses of their visitors. iDefense configured a browser to use a proxy and showed that the Flash test correctly identified the real IP address because Flash does not use Internet Explorer proxy settings.

More aggressive techniques, such as operating proxies, allow law enforcement to determine the source and target of attacks that utilize proxies. While such measures are useful, they are generally very difficult to operate because of abuse. Analysts must carefully monitor activity because attacks now originate from proxy nodes and may result in illegal or otherwise unwanted activity.

2.1.1.3 Conclusion Free and commercial proxies are very numerous on the Internet and can use standard protocols and ports. Other proxies are more difficult to identify, and administrators can detect the use of proxies through configuration changes, IDSs, or tools like decloak.net. Attackers who want to hide their locations have resources available to them. Since it is difficult to detect all proxy users accurately, proxy tools and services will continue to be useful for attackers.

2.1.2 Tunneling Techniques

Most enterprise security controls include strong firewalls, intrusion detection systems (IDSs), and user policies, such as proxies and time-of-day rules that limit the amount and type of traffic generated on user networks. Tunneling data through other protocols often bypasses these controls and may allow sensitive data to exit the network and unwanted data to enter. It is even possible to extend all networks through these means without ever triggering an alert or log entry.

Most researchers cannot help but think of secure shell (SSH) when hearing the word *tunneling*. The authors of SSH, the encrypted version of Telnet "on steroids," designed it to be able to tunnel data over the connections it makes so that other applications and protocols could potentially be more secure. Data, after all, is no different when it is composed of keystrokes and terminal printouts than when it is simply files sent over FTP, Web requests sent over HTTP, or entire IP packets. That is all tunneling really is—a way to transfer arbitrary data in the payload of a protocol and then potentially interpret them differently or in some other extended way than originally intended.

A common, simple form of traffic tunneling in SSH is the tunneling of a Transmission Control Protocol (TCP) port. When a user configures such tunneling over an SSH session, the protocol simply proxies a TCP connection over the SSH connection, and the content of the TCP connection does not flow directly from source to destination, but rather through the SSH connection. One side of the SSH connection (either server or client) listens on a specified TCP port as the source of the data and transfers all the data to the other side of the SSH connection. This other side then forwards the data to the specified TCP destination. An SSH tunneling configuration can become more complicated, because users can configure it to provide a reverse tunnel or arbitrary application proxying through protocols such as SOCKS, but the underlying concept remains the same. Exhibit 2-4 shows how an SSH connection can tunnel a Telnet connection securely between trusted environments. The example tunnels traffic between two unrelated hosts that have no SSH capability to illustrate the flexibility of the solution.

Researchers designed SSH to provide this capability, but it is simple to block. The Internet Engineering Task Force Request for

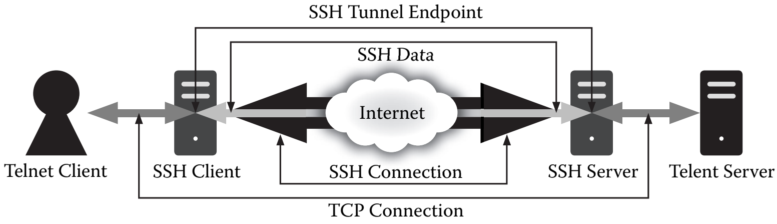


Exhibit 2-4 Telnet tunneled over a secure shell (SSH) connection.

Comments (IETF RFC), which has published documents describing protocol standards, defines SSH's port cleanly so administrators can filter it. The traffic signature, especially the initial handshake, is obvious, and some deep packet inspection tools may block it regardless of what port a user chooses. Still, tunneling is not limited to SSH. A deft attacker can coax any protocol into tunneling traffic; however, for the tunnel to be valuable for hoarding data in and out of a network, protocols with substantial areas of payload work best. Many types of open source software already exist, all of which allow tunneling through well-known protocols, which attackers can use out of the box or with some simple tweaking to defeat most firewalls' rules, proxies, and other administrative access controls quickly. By writing custom applications that act as the client and server for other protocols in a given environment, malicious code can hide its activities and gain unfettered access to and from any network. To illustrate this point, this section examines some of the most common unrestricted protocols in the enterprise—HTTP, the domain name system (DNS), and Internet Control Message Protocol (ICMP)—to show how open and flexible they are.

2.1.2.1 HTTP HTTP has become the de facto high-level protocol on the Internet. As the protocol used for accessing content on the World Wide Web, developers adapted it to carry much more than just the static text and images of Web pages. It now carries audio and video streams, can transfer large files, and can even carry application-to-application remote procedure calls (RPCs). Its ubiquity and indispensability make it a prime candidate for tunneling operations.

Referring to Exhibit 2-4, in the case of HTTP and most other tunnels, other appropriate software that communicates via the protocol

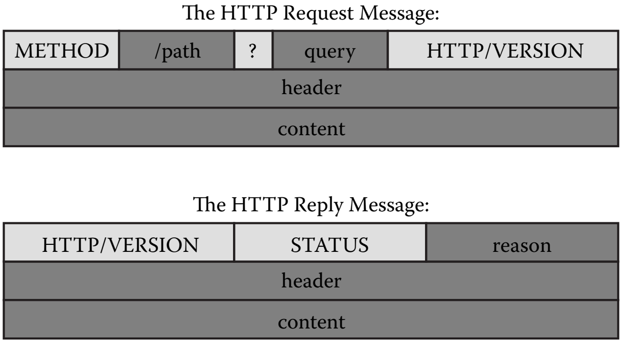


Exhibit 2-5 HTTP messages.

of choice simply replaces the client, server, connection, and data (payload). Exhibit 2-5 shows the syntaxes of an HTTP request and reply that illustrate areas of the protocol that can contain discretionary information for data transfer.

As one can see, the protocol allows, in essence, unlimited space for content (or payload) in the request or reply message in addition to other open areas, such as the headers, whether this content includes arbitrary custom headers or inappropriate data in valid headers. This makes it convenient to transfer arbitrary data to and from an HTTP server. All one needs to tunnel the traffic is software that can pretend to talk to the protocol but in reality can transfer data for some other (perhaps nefarious) purpose. A tunneling Web server or a tunneling Web application running on a legitimate Web server will work. Both types of solutions are readily available as open source software. Since tunneled traffic looks and acts like HTTP, application proxies are not a viable defense, as malicious users or software will simply use the proxies as their administrators intended: to transmit HTTP message through a control point.

Most malicious code already acts as a simple HTTP tunnel; in practice, it posts sensitive data to malicious Web servers for purposes other than to retrieve a Web-based resource. It also sometimes communicates over HTTP; however, it may not act as fully functional tunnels that attackers could use to infiltrate or exfiltrate the network on which they sit. To do that, attackers need a complete tunnel client and server. Some common software for the task is GNU httptunnel,[5] JHttpTunnel,[6] and Webtunnel.[7] There are also paid services dedicated to the task so that a malicious user needs only a client, such as PingFu[8] on Art of Ping. Commercial solutions also exist that almost make the practice appear as legitimate and valid solutions for defeating "restrictive" firewalls, such as HTTP Tunnel by netVigilance.[9]

HTTPS, which is HTTP secured over a secure socket layer (SSL) against eavesdropping and tampering, is no different from HTTP except that it makes detection harder. If a malicious actor cannot eavesdrop, he or she does not have a chance to detect known signatures of tunnels. Proxies that support HTTPS through a CONNECT method may actually make matters far worse, as the CONNECT method simply establishes an arbitrary TCP connection that can send or receive any data.

2.1.2.2 DNS The DNS is the core directory service of the Internet. Without it, translations between names, such as www.verisign.com and IP addresses, could not happen, and it would be difficult, if not impossible, to manage the daily operations of the Internet. Although the DNS architecture is fundamentally different from that of HTTP, the community designed it to carry different data over a different protocol; however, they share one important aspect: required availability. Since DNS is a service that an administrator cannot block and must always make available, it is also a good choice for data exfiltration and tunneling. The basic construct of DNS is different from HTTP and other content delivery protocols. The most common delivery mechanism for DNS is the UDP, not TCP, so the specifications do not guarantee communication reliability. It is hierarchically decentralized so clients may not send transmissions directly to a specific end server, but other servers may relay it, and the size of the information contained in each burst of communication is relatively small. These features make the deployment of functional tunnels more difficult but not impossible. Exhibit 2-6 shows the layout of DNS message packets; the darker areas indicate where software can hide payloads.

Since there are few small areas where data can be stored, DNS tunnels need many packets to transfer large amounts of data and tend to be chatty. In most cases, the tunnel client is simple end user software that makes many requests for nonexistent hosts (where the host names are the encoded payload, such as 0123456789abcdef.badguy.goodguy.com) on some malicious domain or subdomain (badguy.goodguy.com) and expects a payload as the response from the tunnel server. The tunnel

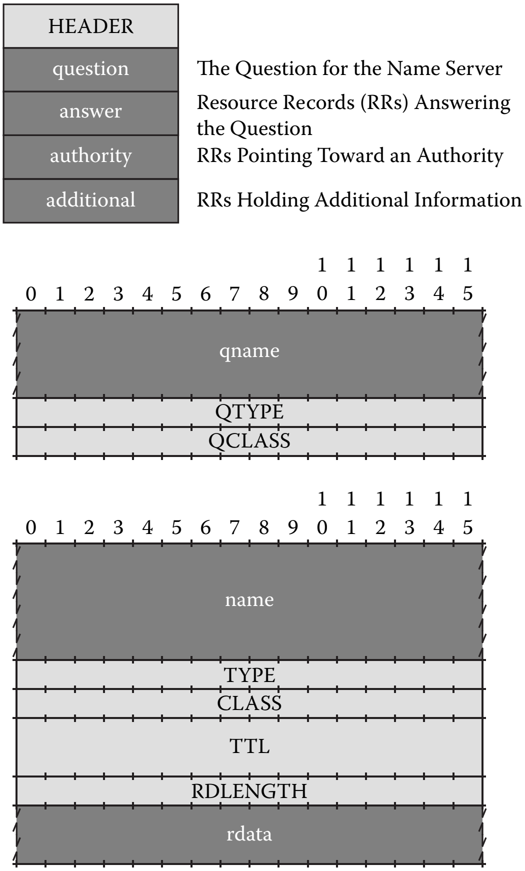


Exhibit 2-6 DNS messages. All communications inside of the domain protocol are carried in a single format called a message (top); the question section is used to carry the "question" in most queries (middle), and the answer, authority and additional sections all share the same format (bottom).

server is generally a rogue (fake) DNS server where intermediary resolvers eventually route the questions. Thus, this setup may require more infrastructure access than just a simple Web server does because an attacker needs to delegate DNS authority to a rogue system. Since the communication requires that the payload be small and since latency may be high, DNS tunnels create many packets and tend to be slow and unresponsive. Additionally, complexity may be high because retransmission and other delivery options, which are readily available in a TCP stream, may not be available. Thus, authors must implement custom solutions.

As with HTTP tunnels, open source solutions and open services are readily available on the Internet. Some of the most common software,

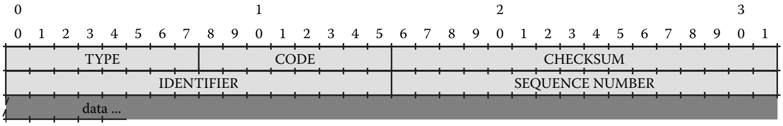


Exhibit 2-7 Internet Control Message Protocol (ICMP) echo message.

such as NSTX,[10] OzymanDNS,[11] iodine,[12] and heyoka,[13] is well established and has been around for years. Services, such as DNStunnel.de,[14] provide tunneling capabilities to those that cannot delegate authority to an "evil" subdomain and make barriers to entry low.

2.1.2.3 ICMP ICMP is a signaling protocol for IP. It is used mostly to deliver status and error messages when IP-based communication errors occur or to troubleshoot and test connectivity status. Although most enterprise policies already block outbound ICMP packets, some Internet service provider (ISP) solutions may not, and its use as a tunnel is mostly to bypass ISP authentication requirements or as a simple covert channel. Most ICMP messages offer little in the way of embedding payload, and implementation details may make it difficult to get the messages delivered; however, ICMP echo messages, which users and administrators alike use to test the accessibility of a host, are well suited for tunneling. Ping, as the most common software that implements this ICMP mechanism, sends data to a host and expects a reply. Exhibit 2-7 is the layout of an ICMP echo message, again showing payload areas.

As shown in the protocol illustration in Exhibit 2-4, there are plenty of data in which to place payload for a tunnel; therefore, ICMP offers good throughput in both directions. ICMP tunneling was one of the earliest[15] methods publicly available to transmit traffic over a protocol in a covert way that essentially abused the protocol. The open source community actively maintains and makes available several software packages to provide this functionality, including Ping Tunnel,[16] ICMPTX,[17] Simple ICMP Tunnel,[18] and Skeeve.[19]

2.1.2.4 Intermediaries, Steganography, and Other Concepts Aside from the three common tunnels discussed in the previous paragraphs, hackers can modify any protocol that filters through a firewall to behave as a tunnel. Assuming deep-packet inspection requires that a given protocol at least match the appropriate syntax (headers must match, no arbitrary data, etc.), tool writers can coerce even FTP, SMTP, and the like into becoming covert channels. All such tunnels have one thing in common, however: it is apparent where their destinations lie.

The task of tracking down specific tunnels and at least shutting down those that are readily apparent is a quick step: identify the destination and block it. This task can become much more difficult with advanced implementations such as intermediary hosts, however. For example, it is possible to create an HTTP tunnel that does not just connect directly to its destination, but rather drops a payload onto some public or common service (forums, comments on blogs, image hosts, etc.)—services that may have legitimate uses. Once the payload arrives at the intermediate service, the destination side of the tunnel picks it up and delivers a reply—all without ever revealing the malicious origin to the exfiltrated network. Recently, researchers discovered such a scheme on Twitter.[20]

The intermediary problem can be even more complex. Steganography is the practice of hiding messages and data in content that is not readily apparent and is a form of security through obscurity. For example, steganographic software and tools can encode messages and data into images[21] so that only users who know where the data exists can retrieve it. Tunnels that use intermediaries for data exchange can deposit payloads that are steganographically encoded to make it harder to detect the covert communication. The Twitter example would have been much more difficult to identify if the payload had been English text with common misspellings, another potential form of steganography. Payload is not even required with such tools. For example, the timing of HTTP requests to some arbitrary Web server, where a recipient can observe the request rate, can be a form of communication in and of itself, without the need to embed the payload. Similarly, timed communication can be accomplished using lower-level protocols, such as raw IP packets.

2.1.2.5 Detection and Prevention The potential of covertly extending a network to the outside world is a clearly unacceptable risk. While the firewalls and IDS that are in place today have their roles to play, they may not be able to identify or prevent tunneling. Tunnels abuse protocols in a way that matches the syntax or the rules of the specifications but not the intent, so despite the efforts of vendors using static signatures for detection—for example, iodine signatures available for Snort—it is trivial to "hack" tunnels to foil the current crop of defenses. Attackers can easily modify open-source tools to appear slightly different from the original, thus defeating a static rule. Any protocol can quickly become a tunnel harbor. Packet inspection firewall rules and IDSs can go only so far in identifying and blocking the threats. Tunnels do have a weakness. They almost never adhere to historical or trended traffic patterns. While HTTP normally has small transfers outbound with larger transfers inbound, tunneling may cause this to reverse or become nearly equal. The duration of connections may also buck the trend, as tunnels need things like keep-alive messages and timeouts. In the case of DNS tunnels, the amount of requests per client, or a set of clients, or even across the enterprise may jump significantly.[22] In the case of ICMP, packet sizes may not match the expected norms, and with any other tunnels, the ratios of different protocols, frequency, and volume can all be indicators of anomalies. These markers point to the need for traffic analysis. Using net flows and other packet capture and aggregation tools, it becomes a statistical problem to map enterprise network trends and identify anomalies. While the crop of commercial tools is limited, several open source solutions are available to begin the process of at least watching and understanding what is going on in the network. Tools like SilK[23] and Sguil[24] can become the gateway to better understanding. They can provide a foundation for trending the network and baselining behavior. Although it may be a labor-intensive process, using it is better than not knowing.

Now may be the time to start thinking about what may lurk in the shadows of networks. Covert exfiltration of information through tunnels is bound to increase as the tools to detect existing software and controls of existing methods become stronger. Broad-based dynamic analytics need to be part of any network user's strategy to ensure identification of the not-so-obvious threats that may be emerging.

2.2 Fraud Techniques

2.2.1 Phishing, Smishing, Vishing, and Mobile Malicious Code

Many phishing attacks against mobile devices use short message service (SMS, or smishing) and voice-over Internet protocol (VoIP, or vishing) to distribute lures and collect personal information. Attackers often send fraudulent SMS messages containing a URL or phone number using traditional phishing themes. Responders either enter their personal information into a fraudulent website, as with traditional e-mail phishing, or, if calling phone numbers, may even provide their information directly to other people. To limit exposure to these growing threats, organizations should not send contact information to users via SMS but instead should be sure phone numbers are readily available on their websites. In addition, financial institutions should carefully consider using mobile devices as two-factor authentication devices, given that customers may use the same mobile device to access the online banking system.

Phishing by way of mobile phones introduces new challenges for attackers and administrators alike. Many phishing attacks against mobile devices use SMS (smishing) and VoIP (vishing). Attackers often send fraudulent SMS messages to many users attempting to gain private information or distribute malicious files. The messages include a URL or a phone number with themes similar to those of traditional phishing messages. Upon calling a phone number, the user may interact with an actual person or a voicemail system—both of which are risks to the user's personal information.

Many legitimate services suffer from doubt and uncertainty related to sending legitimate SMS messages. Organizations should avoid repeating mistakes made with e-mail, which for many organizations is no longer a viable means of communicating with customers due to the pervasiveness of phishing and other fraud.

2.2.1.1 Mobile Malicious Code Although rare and only a more recent occurrence, SMS messages sent to mobile devices may also attempt to convince users to install a mobile malicious code. On or before February 4, 2009, Chinese mobile phone users began reporting a new virus that affects Symbian S60.[25] A signature is required on all code that runs on the S60 third edition, and this virus is no exception; it uses a certificate from Symbian licensed to "ShenZhen ChenGuangWuXian." After the user installs the program, it spreads to other users by sending SMS messages that contain URLs, such as the following, for users to download and install the code:

- hxxp://www.wwqx-mot.com/game
- hxxp://www.wwqx-cyw.com/game
- hxxp://www.wwqx-sun.com/game

The "Sexy View" virus attempts to convince recipients to download and install a Symbian Installation file (SISX) at the URL, but it does not use any exploits to install automatically. Details on this virus are publicly available.[26]

2.2.1.2 Phishing against Mobile Devices Most instances of SMS phishing (smishing) target banks or financial institutions by sending a phone number that the victim calls after receiving the message, resulting in a vishing attack (see Exhibit 2-8).

In the past, attackers used vishing against random targets and were successful at evading defensive filters. For instance, actors have used SMS gateways that allow users to send e-mails instead of spending money per SMS message. In this way, actors send messages to all possible SMS recipients for a gateway. As an example, the SMS gateway receives e-mail messages sent to the phone number 111-222-3333 at the e-mail address 1112223333@mobile.gateway.example.com. SMS gateway providers have responded to abuse by rejecting excessive numbers of messages or fraudulent messages. This is dependent upon the cooperation of the Internet service providers (ISPs) themselves, rather than defensive tools on a mobile device. Uncooperative or unwilling ISPs could cause this type of filtering to fail.

There are several common themes in smishing messages. The following examples all include phone numbers for victims to call. The messages may originate from either a phone number or an e-mail address, both of which an attacker can spoof.

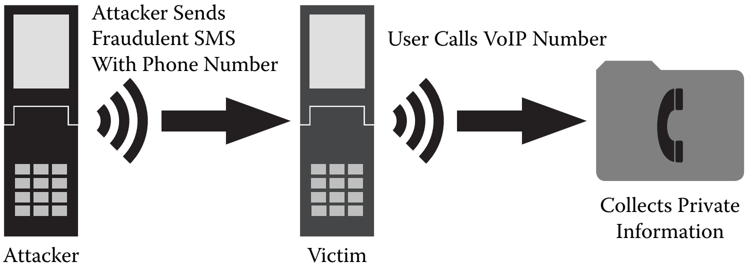


Exhibit 2-8 Example flow of a smishing or vishing attack to steal private information.

> ApplicationCenter/This is an automated message from Lafayette F.C.U. .Your ATM card has been suspended. To reactivate call urgent at 1-567-248-8594[27]

> From: Jennifer [@] fortheloveofmarketing.com
> Your Treasury Department FCU account is expired, for renewal please call us toll free 818.462.5049

> jAPANESS MOTORS AUTO AFRIC, You have won a Brand new Toyota landcruiser VX, in our annual draw. Call Mr. Peter Aganyanya through this No. +254727925287.[28]

> Announcement from PETRONAS MLSY. CONGRATULATIONS your phone number has won a prize of RM 11000. (About US$3,200) Please contact the following number at 0062858853982xx tomorrow morning at 8.00am. Thank you

> Official Microsoft ANNOUNCEMENT: Congratulations! Your mobile phone has won US$ 10 Million prize money. To claim your money, call this number XXXXXXXX tomorrow at 8 AM. Thank you.[29]

Many of these systems use voicemail systems to steal user information, including bank account information. There have been attacks where vishers answer the phones themselves. F-Secure documented one such incident regarding the 0062858853982xx phone number with a transcript and audio files.[30] Similar to traditional phishing attacks, smishing and vishing attacks frequently use fake rewards and fake account alerts.

In January 2008, the Facebook application Secret Crush began phishing users by requesting their mobile phone number through the social-networking website. Subsequently, it would send them messages from a premium SMS service that costs $6.60 per message according to one user afflicted by the scam. Users that reply to the premium rate number (19944989) would receive the bill to their mobile phone.[31]

Whocallsme.com is a resource where users frequently report issues related to phone numbers. Users often report SMS scams, banking fraud, and other incidents to this website based upon the originating phone number. A few examples include

> Dear Credit union customer, we regret to inform you that we had to lock your bank account access. Call (647) 827-2796 to restore your bank account.
>
> !!Urgent! Your number has been selected for a $5000 prize guaranteed! To claim your prize call +423697497459

Organizations should monitor their own SMS number services via sites like whocallsme.com to see if users are suspicious of their services. Such suspicions could indicate mistrust in the legitimate service or attackers who are spoofing the number of the affected organization to improve their chances of gaining trust.

Smishing and vishing are serious problems. Antiphishing products are designed to filter e-mails, but mobile phishing is more difficult to filter for both users and automatic products. SMS messages contain much less tracking information; therefore, recipients will not be able to determine from where they originate. Mobile phone browsers and SMS programs also lack integrated phishing defenses built into today's e-mail clients and browsers. Smishers also often spoof the source address and use a large number of different phone numbers to perform vishing. Mobile browsers also make it difficult to determine the legitimacy of a URL. The small-form factor and limited display are incapable of displaying full URLs, and it can take as many as ten clicks to access the security information of a site. Most mobile browsers lack support for protections normally available on desktop systems such as URL filtering, phishing toolbars, and extended validation (EV) SSL certificates. Based upon these concerns, it seems likely that users of mobile devices have an increased risk of falling victim to a phishing attack when they surf with mobile browsers or receive fraudulent SMS messages.

2.2.1.3 Conclusions To combat the uncertainty caused by smishing and vishing, organizations that plan to contact users via SMS should not encourage users to depend upon caller ID, phone numbers, or the contents of a message. To limit exposure to these problems, organizations should clearly advertise their legitimate SMS numbers via their website and avoid sending phone or SMS contact numbers within messages whenever they contact users.

Concerning mobile phishing threats, financial institutions should take great care to educate their customers regarding how they plan to offer services and communicate via mobile devices. Additionally, customers should avoid accessing online banking through mobile devices until the platforms implement stronger antiphishing measures that are on par with desktop solutions. Some institutions choose to implement custom applications for mobile access to online banking, which may mitigate this threat when consumers use that as the sole mobile access method. Finally, financial institutions should carefully consider using mobile devices as two-factor authentication devices, given that customers may use the same mobile device to access the online banking system.

2.2.2 Rogue Antivirus

During the past year, fake antivirus programs have become dramatically more prevalent and are now a major threat to enterprises and home users. Moreover, attackers often bundle this software with stealthier malicious programs. Fortunately, in attackers' attempts to get users' attention, rogue antivirus software also alerts administrators to system compromises and inadvertently exposes other malicious software.

Attackers aggressively target users with Trojan applications that claim to be antivirus programs. These rogue antivirus applications, once installed, falsely report security issues to mislead victims into purchasing a purported "full" version, which can cost each victim up to US$89.95. Victims have had little success when contacting the payment providers for refund and removal.[32] PandaSecurity estimates that rogue antivirus applications infect approximately 35 million computers each month and that cyber criminals earn US$34 million each month through rogue software attacks.[33]

"Antivirus XP" and numerous other rogue security applications are some of the most prevalent pieces of malicious code that have appeared in the first half of 2009 (see Exhibit 2-9). According to Luis Corrons of PandaLabs, his company observed a significant growth in rogue antivirus applications from January to June 2009, the highest being in June 2009 with 152,197 samples.[34]

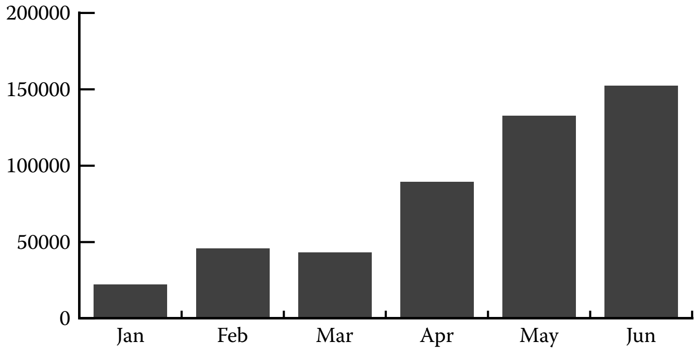
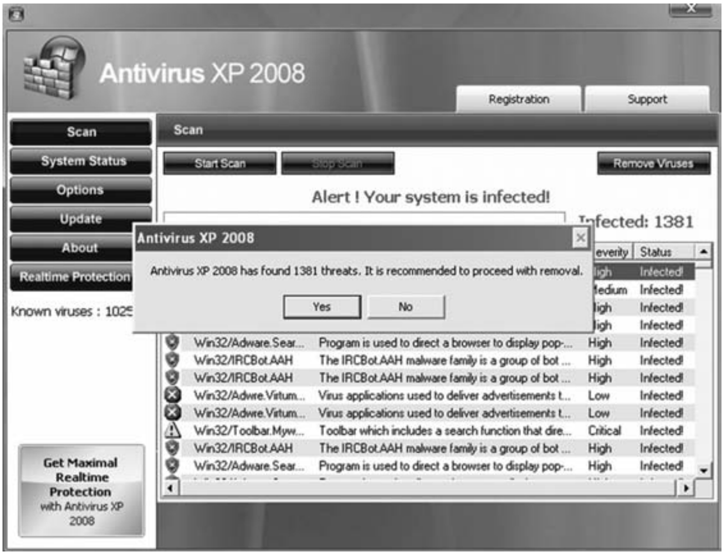


Exhibit 2-9 Rogue antivirus programs in 2009. *Source*: http://www.antiphishing.org/reports/apwg_report_h1_2009.pdf.[35]

One possible reason for the increase is that pay-per-install and affiliate programs encourage more attackers to install such software. According to some pay-per-install rogue antivirus sites, affiliate programs offer an attacker approximately half of the purchase price for each victim who buys the software.[36] This encourages a diverse group of attackers to distribute the software. Though rogue antivirus software emerged in 2004, iDefense has observed a huge increase in this type of malicious activity between 2007 and 2010.

Although the main goal of the Antivirus XP 2008 program (see Exhibit 2-10) is to convince users to purchase fake software, attackers who bundle it with other malicious programs are a major concern to enterprises. The noisy nature of rogue antivirus programs can be beneficial to organizations who take appropriate actions to remove dangerous software that attackers bundle with it. Since the rogue antivirus application often changes a user's background, displays pop-up windows, modifies search behavior, and displays fake windows and security center messages, it often makes its presence repeatedly visible to users. This can be a benefit, if system administrators aggressively audit infected computers for other malicious programs with which it is bundled. iDefense has observed attackers distributing rogue antivirus applications in conjunction with rootkits, banking Trojans, e-mail viruses, or other information-stealing Trojans. These include, but are not limited to, Zeus and Torpig.

Attackers that install rogue antivirus applications often use social-engineering techniques to trick victims. To spread, some variants are



Alert ! Your system is infected!

Infected: 1381

Antivirus XP 2008

Antivirus XP 2008 has found 1381 threats. It is recommended to proceed with removal.

		everity	Status
		igh	Infected!
		ledium	Infected!
		igh	Infected!
		igh	Infected!
Win32/Adware.Sear...	Program is used to direct a browser to display pop-...	High	Infected!
Win32/IRCBot.AAH	The IRCBot.AAH malware family is a group of bot ...	High	Infected!
Win32/IRCBot.AAH	The IRCBot.AAH malware family is a group of bot ...	High	Infected!
Win32/Adwre.Virtum...	Virus applications used to deliver advertisements t...	Low	Infected!
Win32/Adwre.Virtum...	Virus applications used to deliver advertisements t...	Low	Infected!
Win32/Toolbar.Myw...	Toolbar which includes a search function that dire...	Critical	Infected!
Win32/IRCBot.AAH	The IRCBot.AAH malware family is a group of bot ...	High	Infected!
Win32/Adware.Sear...	Program is used to direct a browser to display pop-...	High	Infected!

Exhibit 2-10 Fake antivirus application displays false threats.

bundled with mass-mailing capabilities to send URL links or attachments through e-mail messages. Others attempt to perform search engine poisoning, either through sponsored links or by promoting their search terms associated with recent events. To update their websites with the most common search terms, actors performing search engine poisoning bundle their rogue antivirus applications with other programs that monitor and collect user search terms. Some instances of social-engineering attacks use fake Adobe Flash codecs or other themes to trick victims.

Many other examples of rogue antivirus applications install by using Web exploit kits. Web exploit kit operators may choose to install rogue antivirus applications to make money, or they may allow third-party groups to purchase installs. In either case, the operator may install multiple different malicious programs.

The business model around rogue security applications encourages third parties to distribute code and participate in the revenue stream. As a result, there are a variety of different attacks that install rogue antivirus applications. No single group is responsible for distributing the software because of the shared profits. The use of the pay-per-install model is a strong motivator for attackers who wish to make money from installing software. The huge success of this model of separation between deployment and exploitation is similar to other successful business models like fast-flux and other pay-per-install networks. This type of model will guarantee increased activity and new actors in the near future.

Due to its shared benefit, many attackers can make additional revenue from installing rogue antivirus applications. These applications do not require attackers to alter existing behavior, and they can install multiple different programs at the same time. Due to the frequent bundling of rogue antivirus applications with other malicious programs, organizations should evaluate whether the attacker installed any other malicious code.

2.2.2.1 Following the Money: Payments Most of the rogue antivirus incidents that iDefense investigated use third-party payment organizations. These organizations accept credit card payments and create a layer of protection and security for attackers who use them. These payment processors typically use legitimate SSL certificates and claim to handle fraud requests and operate on a permanent 24/7 basis. The payment processors' connection with rogue antivirus vendors is not exclusive; therefore, law enforcement cannot always shut them down immediately. In past instances, iDefense reported the abuse to the appropriate certificate authorities. Afterward, authorities were able to take the payment processors offline.

In many instances that iDefense investigated, several similar payment providers exist on the same IP address. The payment providers are highly suspicious because they use multiple registration names, domains, and contact addresses and countries, despite their singular purpose to accept money for rogue antivirus payments.

Several of the payment provider sites do not list a phone number unless replying to an authorized customer. They also list in their terms of service that they avoid taking responsibility for customer content.

2.2.2.2 Conclusion A large variety of attacks that install rogue antivirus applications exists. Many use social engineering because it seems somewhat more likely that attackers will be able to convince a victim of social engineering to pay for rogue antivirus software. However, attackers install it using a variety of other techniques and themes due to the pay-per-install model. Organizations that fall victim to rogue antivirus software should evaluate infected computers for bundled software that often accompanies the malicious programs and that may go unnoticed by victims who attempt to disinfect their computers.

iDefense expects to see continued growth in volume and innovation in the illegal distribution of rogue security applications. The number of methods used to distribute the software will continue to expand and the number of techniques used to scare users into buying the software will increase, possibly including regional variations in different languages. Shutting down payment processors is marginally successful over long periods, indicating the illegitimate nature of those connected with rogue antivirus applications. Customers who are suspicious of third-party payment providers may attempt to view secure sockets layer (SSL) certificate registration dates or domain registration dates to determine how new the payment provider is and whether they can expect them to faithfully handle their payments.

2.2.3 Click Fraud

Having provided revenue for a substantial portion of online activity, advertising on the Web has largely been a success for advertisers and online companies. Not surprisingly, fraudsters abuse ad networks by generating invalid traffic for profit, competitive advantage, or even retribution. Advertisers complaining about charges for false traffic have made combating click fraud a major issue for online advertisers.

As with most "free" content in other media, advertising funds much of the World Wide Web; however, unlike the world of television and print ads, it is very easy to become an ad publisher on the Internet. The Web is interactive and allows advertisers to know exactly how many potential customers viewed an ad and how many clicked the ad. This knowledge leads to an advertising model known as *pay-per-click* (PPC), in which advertisers pay ad publishers each time a potential customer clicks an ad on the publisher's website. This direct relationship between the number of clicks and the amount of money earned by the publisher has resulted in a form of fraud best known as *click fraud*. Anchor Intelligence reports that in the second quarter of 2009, 22.9 percent of ad clicks are attempts at click fraud.[37] In this section, we will look at how criminals make money through click fraud and how compromised computers make preventing this type of activity very difficult.

2.2.3.1 Pay-per-Click Any advertising transaction has three primary parties: the advertiser, the publisher, and the viewer. The advertiser is a company that produces content it would like to display to potential customers. This content is an advertisement for a specific product or service that is likely to generate revenue for the advertiser. The publisher is a creative outlet that produces content that will draw visitors to its medium. These visitors view the ad and, ideally, purchase the advertised product or service. The advertiser pays a fee for a specific number of "impressions," which is the estimated number of times a viewer will see the ad. This model is essentially the same across all forms of media, including print, radio, and television.

PPC uses the same general model as other forms of advertising, but introduces an interactive component. While an especially impressive car commercial may entice a television viewer into purchasing a new sedan, it is difficult for the advertiser to link a particular ad directly to that sale. The Internet makes this possible because when a viewer finds an ad compelling, he or she can click it to get more information or purchase the product. If, and only if, the viewer clicks on the ad, the advertiser will pay the publisher a fee. The direct correlation between the viewer's action and the cost to the advertiser is the primary distinction between PPC and impression-based advertising.

The ultimate goal for the advertiser is to convert ad clicks to actions that generate more revenue than the advertising campaign costs. When the viewer takes the desired action, be it signing up for a newsletter or purchasing a new car, a conversion has occurred. This conversion completes the PPC business model. Exhibit 2-11 shows how money flows in this business model.

With the advent of PPC advertising networks like Google AdWords and Yahoo! Search Marketing, anybody with a website can become an ad publisher. Publishers who use these networks are affiliates. Affiliates add HTML code to their website, which draws ads from the advertising network and displays them inline with the affiliate's content. The affiliate and the advertising network then split the PPC fee each time a viewer clicks an ad.

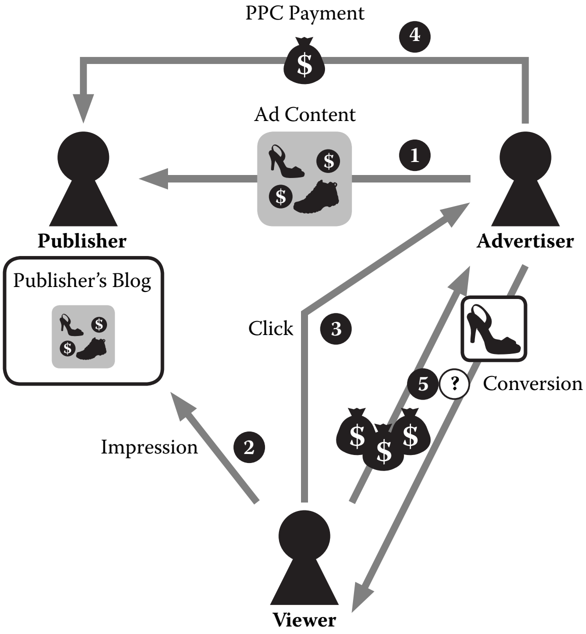


Exhibit 2-11 Pay-per-click business model.

2.2.3.2 Click Fraud Motivations Click fraud occurs when an ad network charges an advertiser for a click when there was no opportunity for a legitimate conversion. There are many possible motivations for a person to click an advertisement without any intention to purchase a product or service. Publishers perform the most obvious and common form of click fraud. Clicking an ad on one's own website directly generates revenue for the publisher. Clicking the ad fifty times generates even more revenue.

While a publisher can click his or her own ad, he or she could just as easily ask friends to click the ads. For instance, a blogger who wants to increase revenue might make a post simply asking his or her readers to click every ad on his or her website each time they visit. While they are legitimate users, these clicks will not result in a conversion for the advertiser.

An advertiser's competitor might also be inclined to commit click fraud. If each click costs the Acme Corp. money, Acme's chief rival might click the ad a few hundred times a day to cost them as much money as possible. In this case, the publisher benefits from the fraudulent clicks, but the motivation is merely to harm the advertiser.

Competing publishers might also be motivated to commit click fraud. Because click fraud has become such a widespread problem, most advertising networks work very hard to detect it and will ban affiliates suspected of committing click fraud. A competing publisher can click ads on a competitor's website to frame them for click fraud. Once detected, the ad network may ban the competitor, which will result in an increased share of the advertising revenue for the actual click fraudster.

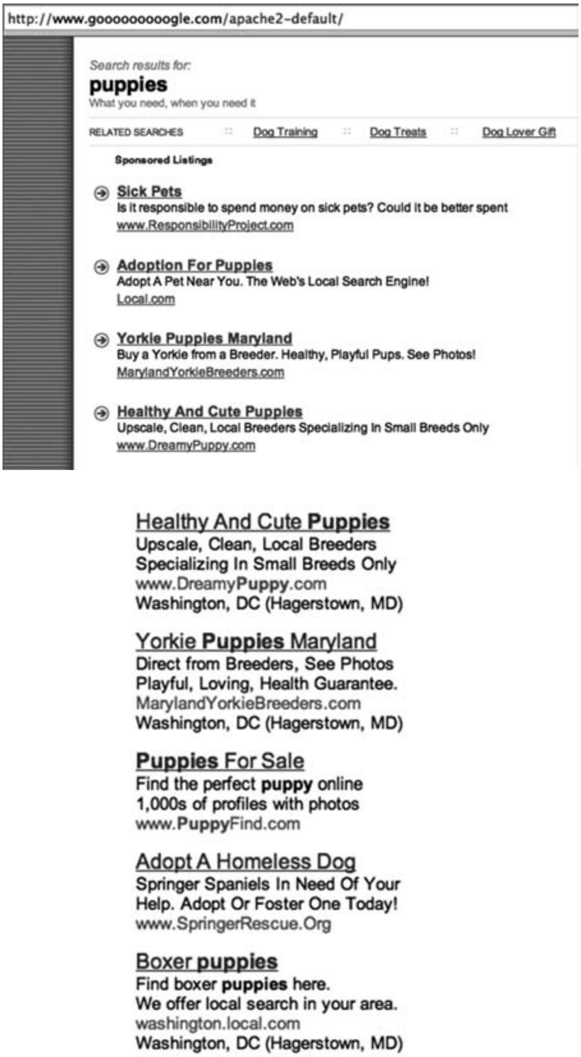
Nonfinancial motivations might also cause a person to commit click fraud. If a person disagrees with how Acme Corp. treats its workers, they might click Acme ads to cost the company additional money. As in the case of clicks from a competitor, the intent is to harm the advertisers, but the outcome also benefits the publisher.

2.2.3.3 Click Fraud Tactics and Detection The simplest form of click fraud involves manually clicking advertisements through the browser. While this method is effective at generating a small number of additional clicks, fraudsters have developed sophisticated methods to produce the volume necessary to earn higher revenue.

First, the fraudster must create a website that displays advertisements. A popular way to do this is to create a search engine that only displays advertisements relevant to a queried word. One such search page uses a very unlikely typo of google.com, *gooooooooogle.com*. The top portion of Exhibit 2-12 shows the results returned when searching this page for "puppies," and the bottom portion shows advertisements displayed on Google's search page when querying for the same word.

All of the results returned by gooooooooogle.com are actually advertisements, and many of them are the same ads returned by a Google search for the same term. A portion of the fee that advertisers pay for each click will go to the owners of gooooooooogle.com.

With the advertisements in place, the fraudster must now find a way to click as many of the ads as possible without the ad network noticing the abuse. Botnets, the Swiss Army knife of the modern Internet miscreant, are the key to a successful click fraud campaign. As the click fraud problem grew, ad networks began developing fraud detection mechanisms that made simple click fraud impossible. For instance, when a single IP address registers multiple clicks in a 30-minute period, the ad network may simply discard all but the first click when



http://www.goooooooogle.com/apache2-default/

Search results for:

puppies

What you need, when you need it

RELATED SEARCHES :: Dog Training :: Dog Treats :: Dog Lover Gift

Sponsored Listings

Sick Pets
Is it responsible to spend money on sick pets? Could it be better spent
www.ResponsibilityProject.com

Adoption For Puppies
Adopt A Pet Near You. The Web's Local Search Engine!
Local.com

Yorkie Puppies Maryland
Buy a Yorkie from a Breeder. Healthy, Playful Pups. See Photos!
MarylandYorkieBreeders.com

Healthy And Cute Puppies
Upscale, Clean, Local Breeders Specializing In Small Breeds Only
www.DreamyPuppy.com

Healthy And Cute **Puppies**
Upscale, Clean, Local Breeders
Specializing In Small Breeds Only
www.Dreamy**Puppy**.com
Washington, DC (Hagerstown, MD)

Yorkie **Puppies** Maryland
Direct from Breeders, See Photos
Playful, Loving, Health Guarantee.
MarylandYorkieBreeders.com
Washington, DC (Hagerstown, MD)

Puppies For Sale
Find the perfect **puppy** online
1,000s of profiles with photos
www.**Puppy**Find.com

Adopt A Homeless Dog
Springer Spaniels In Need Of Your
Help. Adopt Or Foster One Today!
www.SpringerRescue.Org

Boxer **puppies**
Find boxer **puppies** here.
We offer local search in your area.
washington.local.com
Washington, DC (Hagerstown, MD)

Figure 2.12 goooooooogle.com search results for "puppies" (top) and Google search ads for the same word (bottom) result in many of the same ads.

charging the advertisers. Ad networks can also use browser cookies to determine when the same user is clicking an ad multiple times. Botnets solve these problems for the fraudster because each infected computer has a different IP address and a different browser. In 2006, Google discovered the Clickbot.A botnet, a click fraud botnet consisting of more than 100,000 nodes.[38]

While the distributed nature of botnets benefits click fraud, it can also be a detriment. Advertisers display many ads only in countries or regions in which their ads are relevant. For instance, a U.S.-based restaurant chain may not want to advertise to viewers in China or Russia. Clicks emanating from IP addresses in countries that should not see the ads might indicate click fraud.

Behavior patterns after the fraudster clicks an ad are another way in which ad networks and advertisers can detect potential click fraud. In the PPC industry, an ad click that does not result in any additional clicks on the website is a *bounce*, and the percentage of visitors who exhibit that behavior is the *bounce rate*. While a poor-quality website might have a high bounce rate because visitors do not find it interesting, if clicks from a particular publisher have a much higher bounce rate than others, it may indicate click fraud.

A click fraud botnet can generate clicks in multiple ways. The botnet may simply download a list of key words and visit a random ad returned by the query for each word. Another technique is to redirect actual searches made by the infected system. When an infected user makes a query to a search engine, the malicious software will alter the results returned so that clicking them results in an ad click controlled by the fraudster. This technique may be more effective at evading detection because real users may actually click additional links on the page and potentially even purchase products.

2.2.3.4 Conclusions While click fraud appears to be a problem with a scope limited to just advertisers and ad networks, fraudsters' use of infected computers to click ad links makes click fraud a problem for everyone with a computer. Being part of a click fraud botnet consumes a system's bandwidth and displays additional advertisements to the user, which is usually undesirable.

Companies should be cautious when spending advertising money on the Internet and should check which techniques their publishers use to detect and prevent click fraud. Organizations that already advertise on the Internet and are concerned that they may be victims of click fraud can use the techniques described in this section to detect some forms of click fraud. Companies such as Click Forensics[39] and Anchor Intelligence[40] provide third-party solutions to assist in discovering and weeding out invalid ad clicks.

2.3 Threat Infrastructure

2.3.1 Botnets

Systems connected to the Internet are at risk of infection from exposure to social-engineering attacks or vulnerability exploitation. Regardless of the infection vector, compromised machines can wait for commands from the attacker, which turns the system into a bot. A bot is a single node added to a network of other infected systems called a botnet.

A botnet is a network of infected systems controlled by an administrator known as a *botmaster*. A botmaster controls many bots by issuing commands throughout the botnet infrastructure. The ability to run commands on many systems makes botnets practical for malware authors seeking a management solution and provides multiple capabilities.

Botnets would not be capable of performing any activities without communication between the botmaster and bots. The type of communication protocol depends on the network topology of the botnet. While botnets use many different topologies, all botnets fall into two main categories, centralized and decentralized; however, some botnets implement elements from both categories to create a hybrid structure.

A centralized topology receives its name due to the central location of the command-and-control (C&C) server(s). The most basic form of this topology uses a server to C&C all bots within the botnet; however, other more advanced forms of centralized networks exist and fall into two subcategories to describe the differences in infrastructure. Exhibit 2-13 shows the infrastructures of the different centralized botnets.

A multiserver builds on the basic centralized botnet topology by using more than one server for C&C. Multiple C&C servers make botnets more reliable and less vulnerable to takedown attempts. This

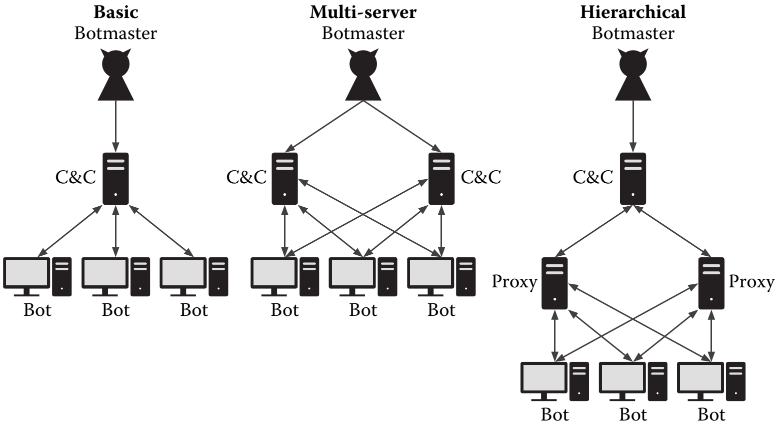


Exhibit 2-13 Centralized botnet infrastructures.

type of topology allows bots to receive commands even if one server is unreachable. If a server goes offline, the botmaster can still communicate with bots through other C&C servers. The Asprox botnet was an example of this type of botnet as it issued a configuration file to each bot that included a list of C&C servers.

Another type of centralized topology uses a hierarchical infrastructure. This type of topology builds on the multiserver technique by using layers of servers to proxy communications between the bots and C&C servers. This setup promotes reliability and longevity within the botnet, as the proxying servers cover the true location of the C&C servers. Drawbacks to this configuration include increased infrastructure complexity and higher network communication latency due to the addition of the intermediary proxies. An example of a hierarchical botnet is Waledac, which has its bots communicate directly with repeater nodes that act as proxies. The repeater nodes proxy requests to another layer known as *transport security layer* (TSL) *servers* (the name derives from a registry key added to infected systems) that act as C&C servers. If the TSL server cannot handle the request, it passes the request upward to the final layer of C&C known as *upper-tier servers* (UTSs).[41] This exemplifies the layered structure of a hierarchical botnet.

Regardless of the type of centralized topology, nodes within the botnet need to locate the C&C server before receiving commands. Most bots do not listen on ports for commands from the botmaster because administrators can easily detect these unknown listening ports and network devices could block inbound connection attempts. Instead, bots will initiate communication with the C&C server to appear legitimate and bypass network controls.

To allow bots to locate the C&C server, centralized botnets can use hardcoded IP addresses. Hardcoded IPs supplied with the malware inform the bot of the server's address immediately after infection; however, this method suffers if the server is unreachable or taken down, which renders the bot useless.

Most botnets rely on the domain name system (DNS) and domain name lookups for bots to locate the C&C server. To use the DNS, a bot queries its name server for the IP addresses that resolve the domain name of the C&C server. The DNS allows the botmaster to introduce reliability and resiliency for a server takedown by using multiple IP addresses or fast-flux to resolve domain names. A botmaster can increase the reliability of a bot locating the server by using multiple IP addresses to resolve the domain name. This allows bots to reach a C&C server in the event that some IP addresses are unreachable.

To increase resiliency to server takedown attempts, a botnet can use fast-flux to cycle IP addresses that resolve a domain name as described in the "State of the Hack" article on fast-flux.[42] The use of fast-flux domains thwarts server takedown attempts, but is still vulnerable to domain takedown attempts. Several botnets, such as Conficker and Kraken, address this issue by introducing a domain generation algorithm. In essence, bots generate a list of possible domains to use in locating the C&C server, and the botmaster registers new domains based on this list. This technique thwarts domain takedown efforts, as the domain used by the C&C server constantly changes.

The second botnet category, called *decentralized*, differs dramatically from a centralized configuration. A decentralized botnet does not have a particular server or set of servers designated to control bots. These advanced botnets use peer-to-peer (P2P) communications to send commands between bots throughout the botnet. With no centralized location, this type of botnet does not use the same techniques to locate commands as centralized botnets. A bot must locate other peers within the botnet to receive commands by using the P2P protocol's peer discovery mechanisms. This type of botnet is very difficult to dismantle without disinfecting each bot but

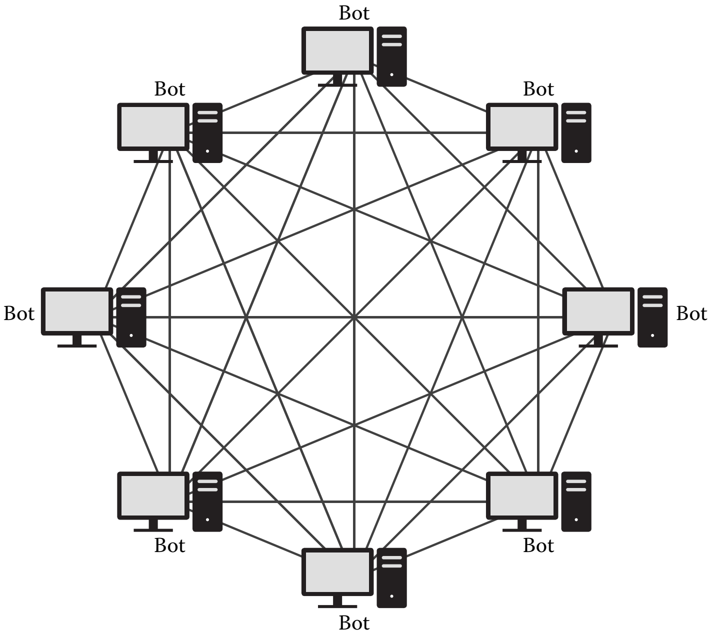


Exhibit 2-14 Decentralized botnet architecture.

introduces complexity and latency before all bots receive commands. Exhibit 2-14 shows a decentralized botnet and exemplifies the complexity and number of communication paths that introduce command latency. For example, a version of the Conficker worm incorporated P2P communications for C&C. Bots would scan the entire Internet for other peers and attempt to share updated binaries with each other based on version number.[43]

Communication within a botnet is crucial, and many botnets use Internet Relay Chat (IRC) as a rally point to manage infected machines. Typically, IRC botnets allow a botmaster to issue commands to bots connected to an IRC channel. For example, the IRC botnet known as SDBot used NOTICE, TOPIC, or private message (PRIVMSG) commands to control infected machines in an IRC channel.[44] In most network environments, the occurrence of IRC activity is anomalous because they rarely experience legitimate IRC traffic.

Other botnets employ common protocols, such as HTTP, for communication. By using common protocols, botnet communications can slide under the radar and blend in with legitimate traffic on a network. If the botnet uses HTTP for communication, the bot makes GET or POST requests to a server and the server replies with commands. In some cases, the botnet uses a compromised Web server for C&C by issuing commands within HTML comments. This further legitimizes the appearance of botnet communications.

In other cases, a botnet uses a custom protocol for communication. Botnets with a decentralized topology generally adapt a P2P protocol to fit their needs. The Storm botnet used a variant of a P2P protocol called Kademlia. Storm used this modified version, known as Overnet, to discover other peers within the botnet and share commands introduced by the botmaster with other peers.[45] Other botnets use the port of legitimate services such as HTTPS but provide their own communication schemes to circumvent firewalls.

Botnets are very capable at performing malicious deeds for a prolonged period. The most obvious capability available through the control of a large number of infected systems is the capability to carry out distributed denial of service (DDoS) attacks. The botmaster can issue a command to have bots repeatedly send requests to a server or network. With enough bots and enough requests, the botnet can create a denial of service (DoS) condition for the targeted server or network as discussed in the "State of the Hack" article on DoS conditions.[46]

Botmasters can also use their bots to send spam e-mail messages. According to recent research, botnets account for 87.9 percent of all spam sent on the Internet.[47] A botnet can achieve these numbers by having each bot send e-mails at a rapid rate, which can result in millions or billions of e-mails per day. Spam from these systems is difficult to block due to the number of unique systems and the constant addition of new systems.

In addition to using bots to send spam, botmasters can steal sensitive information and credentials from the infected systems within the botnet. The botmaster can use sensitive information for identity theft or to generate revenue by selling the data on the underground market. Stolen credentials allow the botmaster to access user accounts, such as online banking, social-networking, and e-mail accounts, which can lead to online banking fraud or sensitive information theft.

Botnets also provide a vast resource for hosting by using each bot as a server. Botmasters use their botnet as a content delivery network (CDN) to host phishing pages and malware on infected systems to make server takedown attempts difficult. The vast resources also allow botmasters to provide services to other individuals looking to carry out malice. These services allow individuals without available resources to perform their desired attacks.

Administrators attempting to detect infected bots within their networks need to monitor traffic for botnet communication. Repeated anomalous traffic over IRC or nonstandard ports from a system can indicate an infected machine participating in a botnet. Detecting botnet activity over common ports is more difficult and requires filtering out legitimate traffic. To filter out legitimate traffic, the administrator needs specific knowledge of the botnet's communication and how it appears while traversing the network. Armed with this knowledge, an administrator can create an intrusion detection system (IDS) signature to detect the botnet communications.

Takedown attempts depend on the botnet's structure. Generally, taking down a centralized botnet requires removing all of the C&C servers. By removing all the C&C servers, the botmaster cannot update his or her bots with new servers to contact, effectively disabling the botnet. A decentralized botnet requires a different approach, as there are no specific C&C servers to take down. The overly simplified takedown approach requires dispatching a cease-and-desist command to the peers within the botnet. When an administrator is unable to effect a takedown of the control servers, he or she can and should block the known IPs and domain names associated with a botnet.

The efficiency of attackers successfully exploiting systems exposed a need for a management system for large quantities of infected machines. Attackers evolved their malware to set up logical networks of compromised systems to create a powerful tool to carry out malice and generate revenue. As the security community scrambles to analyze, track, and attempt to take down botnets, the botmasters continue to modify their networks to remain one step ahead.

2.3.2 Fast-Flux

This section explains fast-flux attacks, which utilize temporary records in the domain name system (DNS) to achieve a distributed network of hosts. Resulting in constantly rotating IP addresses, single-flux attacks require disabling the DNS server to take down a domain. Even more robust, double-flux attacks rotate the IP addresses of both hosts and domain name servers, thereby making takedown even more difficult. In addition, fast-flux attacks create a layer of anonymity, further frustrating enforcement efforts. This technique is not new, but it grew dramatically in popularity within the past two years and is now a common occurrence.

Fast-flux is a technique that creates a resilient and balanced network of compromised computers to carry out malicious actions. Fast-flux utilizes DNS to continually update valid domain names with A and NS records that resolve to an ever-changing set of IP addresses. Earning the phrase *flux*, changing the IP addresses within DNS records allows a domain name to point to different IP addresses at different times. By changing the resolving IP addresses, domains point to the currently active set of infected computers.

Coined as being "fast" due to the speed of IP changes, this technique employs short time-to-live (TTL) periods to force frequent requests to the authoritative name server for IP addresses that resolve the domain. Due to a short TTL, clients and caching servers make frequent requests to the name server to update the IP addresses associated with the domain. Exhibit 2-15 shows a fast-flux domain using a short TTL to cycle new A records every 600 seconds.

Fast-flux domains incorporate a collection of bots into a network of resolvable servers by command-and-control (C&C) servers known as *motherships*. These motherships have important duties in controlling and maintaining fast-flux by issuing commands to bots and adding and removing bot IP addresses from DNS records. By cycling IP addresses of infected computers in and out of DNS records, the mothership is able to use active bots to host content and services. The IP cycling in DNS records also combats unreachable compromised hosts due to routing issues, firewall filtering, and infection remediation,

```
;; ANSWER SECTION:
  mtno.ru.   600   IN   A   98.196.113.58
  mtno.ru.   600   IN   A   65.102.56.213
  mtno.ru.   600   IN   A   98.209.249.15
  mtno.ru.   600   IN   A   68.202.88.12
  mtno.ru.   600   IN   A   68.92.101.61
  mtno.ru.   600   IN   A   74.138.219.230
```

Exhibit 2-15 DNS response showing short time-to-live (TTL).

and insures a high probability of reaching an active bot. In addition to controlling infected computers and maintaining DNS records, motherships act as the main servers and respond to client requests. Compromised computers act as an intermediary for the mothership by accepting and forwarding inbound requests to the fast-flux domain. Exhibit 2-16 shows the interaction between a client and the fast-flux domain shown in Exhibit 2-15.

Exhibit 2-16 describes a concept known as *single-flux*, which utilizes static name servers (DNS NS records) to update DNS A records

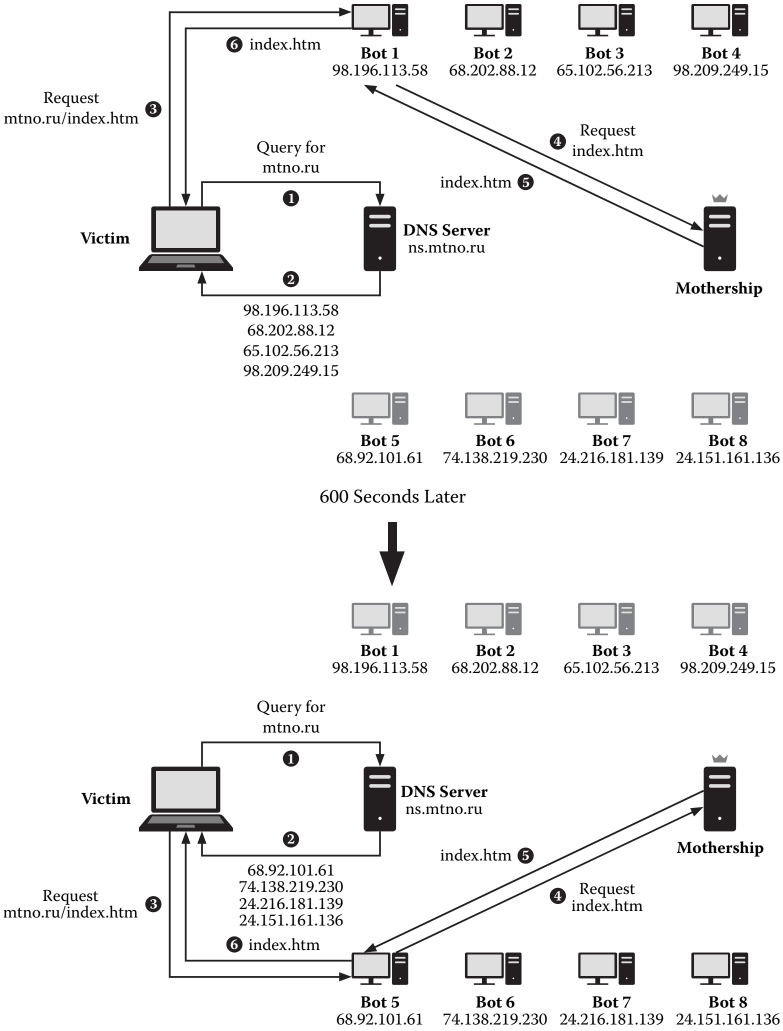


Exhibit 2-16 Victim interaction to fast-flux infrastructure.

with IP addresses of infected computers and continuously cycle new IP addresses. Client requests for a single-flux domain received by the domain's name server are resolved to multiple bots (bots 1–8 in Exhibit 2-16). All requests to the domain go through infected computers to the mothership. Acting similar to a reverse proxy, the infected computers receive the mothership's response to the client's initial request and forward it to the client. Multiple other variations of fast-flux currently exist, and all incorporate the same techniques with different levels of complexity. Double-flux adds a second layer of complexity by cycling IP addresses to name servers, and hydra-flux introduces multiple motherships.[48]

Regardless of the type of fast-flux used, domains become attractive to malicious actors due to the benefits that a fast-flux domain provides. The infected computers form a protective barrier in front of the backend servers by cloaking the true origin of the malicious activity. The cloaking provided from this barrier allows mothership owners to avoid detection and apprehension by authorities. The intermediary bots are the only element of the fast-flux system visually exposed during malicious acts. This allows motherships to remain active for long periods. The lengthened lifetime of motherships increases the resiliency of the entire fast-flux system. The system evades takedown attempts due to bots accepting sole responsibility for actions performed by backend motherships. Other benefits that fast-flux domains offer include load balancing by spreading requests to multiple servers and high availability to the malicious content.

Attributes of a fast-flux domain render it a viable solution to carry out a variety of malicious duties. Phishing campaigns equipped with fast-flux domains remain active for long periods and are difficult to take offline. Exploit toolkits hosted at these domains attempt to exploit vulnerable visitors to install malicious code and ultimately turn the victim into a bot used in the fast-flux system. Malicious code distribution also utilizes fast-flux domains as it allows a centralized location for malicious code downloads without exposure. Spam e-mail also incorporates fast-flux domains to hide mail servers to lengthen the campaign and link to malicious content. Bulletproof hosting takes advantage of fast-flux systems due to the high availability of content and the lack of a mass-blocking technique. Unlike traditional bulletproof hosting providers with specific IP ranges to block, filtering on dispersed IP addresses used in fast-flux domains is futile.

Fast-flux systems are resistant to takedown attempts due to the number of systems involved and the anonymity of the true source of such systems. In a traditional server takedown, an administrator contacts the Internet service provider (ISP) hosting malicious content and provides evidence of abuse. The ISP shuts down the server in response to the report of illegitimate activity. It is impossible to take down a fast-flux system with this traditional process due to the lack of a single ISP to contact. As a result, administrators must contact the domain registrar and provide evidence of malicious content hosted at the domain. The registrar removes access to the fast-flux domain, which stops the current activity, but does not stop the fast-flux operator from registering a new domain name. Adding a new domain name begins the cycle over and allows the fast-flux infrastructure to continue malicious activity.

During the domain takedown procedures, administrators must implement safeguards and countermeasures to protect their assets. The number of IP addresses resolving the domain and the short turnaround time renders appliances that block traffic at the network layer useless. The best method of blocking traffic to fast-flux domains is to black hole the domain at local DNS servers. Black holing the domain blocks client requests and stops the communication with the malicious servers. Filtering traffic to known fast-flux domains at local Web proxies provides a blocking mechanism for HTTP traffic.

Fast-flux domains allow actors to carry out malicious deeds anonymously and for relatively long periods. These domains continue to spread malicious code, send spam, host phishing, and exploit victims, and are a danger to any enterprise. Innovative uses of fast-flux continue to change in the wild and require reactive countermeasures from the security community.

2.3.3 Advanced Fast-Flux

Section 2.3.2 described the workings of a basic fast-flux infrastructure, which uses DNS records to obfuscate the location of malicious sites and frustrates takedown efforts. Variants of the fast-flux technique further complicate tracking and takedown by using multiple domain name servers or even multiple command-and-control (C&C) servers, also known as motherships. These advanced fast-flux methods are known as double-flux and hydra-flux.

The preceding section described fast-flux networks and their general structures, uses, and resiliency to take down. The lack of in-depth detail regarding the types of fast-flux systems requires a second look. The three types of fast-flux existing today are known as single-, double-, and hydra-flux. All three types of fast-flux utilize domain name system (DNS) record updates, occurring on name servers or with domain registrars or both, to conceal the source of malicious activity in attempts to evade detection and takedown. This section describes advanced fast-flux techniques, how they work, and the additional protection each variation provides.

Regardless of type, all fast-flux domains involve a botnet infrastructure that includes one or more C&C servers called *motherships* and infected computers called *bots*. The mothership is responsible for managing the DNS infrastructure associated with the domain, controlling bots, and serving malicious content. Managing the domain involves updating the domain registrar and name servers. The registrar receives updates from the mothership in the form of NS records, which point to name servers that answer queries for the domain. The mothership also updates the configuration file, known as a *zone file*, on these designated name servers with A records that point to bots that resolve the domain. The zone file on the name server also includes a time-to-live (TTL) value that specifies how many seconds the client caches IP addresses for a domain before querying again. To cycle bot IP addresses into A records and to bypass caching features, fast-flux domains use short TTL values to force clients to frequently query the name server for a new set of A records. Bots designated by A records receive content requests sent to the domain, and act as reverse proxies by sending requests to the mothership and relaying the malicious content hosted on the mothership back to the original requester. These bots provide a layer of protection by obscuring the true source of the malicious content and introduce multiple points for takedown.

Exhibit 2-17 shows the protective layers that each type of flux provide the source of the malicious content or actor. The diagram shows single-flux using groups of bots, a single DNS server, and a single mothership that provide three layers of protection between victims

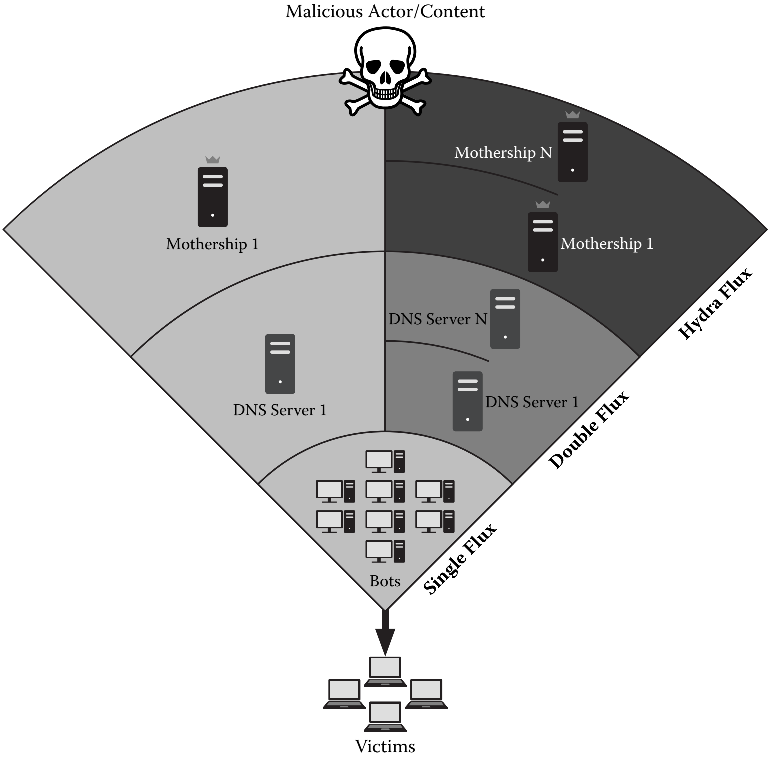


Exhibit 2-17 Diagram of the protective layers that flux domains provide.

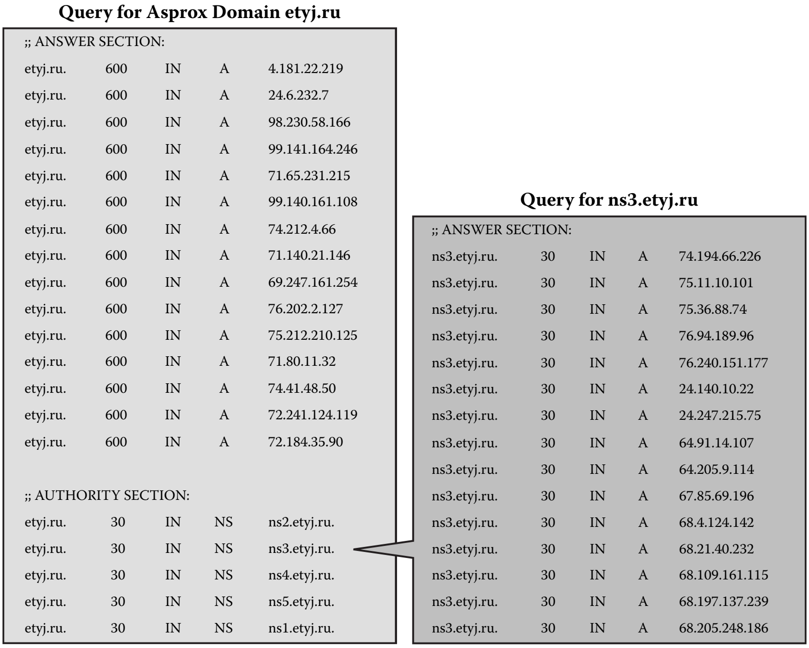
and the malicious actor or content. The use of a single DNS server is a shortcoming in single-flux by presenting a single point on which to focus efforts to stop malicious activity. Double-flux addresses this flaw by adding multiple DNS servers, which in turn adds another layer of protection for a total of four layers. The multiple DNS servers increase the complexity of the infrastructure to conceal the content source further, but double-flux's weakness lies in the use of one mothership. Once discovered, investigators can take down the single mothership to stop malicious activity related to the double-flux domain. Hydra-flux fixes this vulnerability with an extra layer on top of double-flux by implementing multiple motherships. With five layers of protection, hydra-flux is the most advanced type of fast-flux and causes hardship to those attempting to stop malicious activity. With these layers of multiple bots, DNS servers and motherships render server takedown procedures ineffective. The best method to cease malicious hydra-flux activity requires the domain registrar to suspend the domain name.

The three types of fast-flux utilize similar techniques to set up and maintain resilient networks to carry out malicious activity. To implement single-flux, a domain name is registered and assigned a single DNS server to the domain by providing one static NS record to the registrar. The mothership adds and removes IP addresses of bots to the zone file on the name server assigned during domain registration. As a result, the name server responds to a requester's query to resolve the domain with addresses of different bots that subsequently proxy communication between the requester and the mothership. In addition to single-flux techniques, double-flux uses the mothership to update the domain registrar continuously with NS records to change the name servers that answer queries to resolve the domain. Hydra-flux builds on double- and single-flux by implementing multiple motherships along with multiple DNS servers and infected computers. Adding multiple motherships requires updating the configuration on each bot with current motherships to forward requests.

In the wild, the Asprox botnet incorporated a hydra-flux infrastructure to carry out phishing, DDoS, and spam campaigns. The left portion of Exhibit 2-18 shows the results of a query for an Asprox domain name. Asprox fast-flux domains used 14 infected bots within A records, cycling in a new set every 600 seconds, according to the TTL value seen in the second column. The authority section shows the name servers responsible for answering requests to resolve the domain. The right portion of Exhibit 2-18 shows a subsequent request to resolve one of the name server domains. The answer shows multiple A records with a TTL value of 30 seconds, which exemplifies the use of double-flux techniques.

Exhibit 2-19 shows a decoded configuration file sent from a mothership to an infected computer, which updated the bot with current mothership IP addresses. This configuration file is required for the hydra-flux infrastructure, as the IP addresses within the <S> tags instructed the infected computer to proxy requests to the motherships at these addresses.

Single-, double-, and hydra-flux all use the same techniques to evade detection and takedown, but each addresses weaknesses in its predecessor by adding a layer of complexity. The layers of complexity prolong malicious campaigns by obscuring the true source of the



Query for Asprox Domain etyj.ru

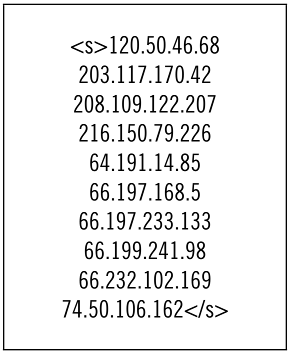
```
;; ANSWER SECTION:
etyj.ru.    600    IN    A     4.181.22.219
etyj.ru.    600    IN    A     24.6.232.7
etyj.ru.    600    IN    A     98.230.58.166
etyj.ru.    600    IN    A     99.141.164.246
etyj.ru.    600    IN    A     71.65.231.215
etyj.ru.    600    IN    A     99.140.161.108
etyj.ru.    600    IN    A     74.212.4.66
etyj.ru.    600    IN    A     71.140.21.146
etyj.ru.    600    IN    A     69.247.161.254
etyj.ru.    600    IN    A     76.202.2.127
etyj.ru.    600    IN    A     75.212.210.125
etyj.ru.    600    IN    A     71.80.11.32
etyj.ru.    600    IN    A     74.41.48.50
etyj.ru.    600    IN    A     72.241.124.119
etyj.ru.    600    IN    A     72.184.35.90

;; AUTHORITY SECTION:
etyj.ru.    30     IN    NS    ns2.etyj.ru.
etyj.ru.    30     IN    NS    ns3.etyj.ru.
etyj.ru.    30     IN    NS    ns4.etyj.ru.
etyj.ru.    30     IN    NS    ns5.etyj.ru.
etyj.ru.    30     IN    NS    ns1.etyj.ru.
```

Query for ns3.etyj.ru

```
;; ANSWER SECTION:
ns3.etyj.ru.    30    IN    A    74.194.66.226
ns3.etyj.ru.    30    IN    A    75.11.10.101
ns3.etyj.ru.    30    IN    A    75.36.88.74
ns3.etyj.ru.    30    IN    A    76.94.189.96
ns3.etyj.ru.    30    IN    A    76.240.151.177
ns3.etyj.ru.    30    IN    A    24.140.10.22
ns3.etyj.ru.    30    IN    A    24.247.215.75
ns3.etyj.ru.    30    IN    A    64.91.14.107
ns3.etyj.ru.    30    IN    A    64.205.9.114
ns3.etyj.ru.    30    IN    A    67.85.69.196
ns3.etyj.ru.    30    IN    A    68.4.124.142
ns3.etyj.ru.    30    IN    A    68.21.40.232
ns3.etyj.ru.    30    IN    A    68.109.161.115
ns3.etyj.ru.    30    IN    A    68.197.137.239
ns3.etyj.ru.    30    IN    A    68.205.248.186
```

Exhibit 2-18 Asprox domain A and NS records.



```
<s>120.50.46.68
203.117.170.42
208.109.122.207
216.150.79.226
64.191.14.85
66.197.168.5
66.197.233.133
66.199.241.98
66.232.102.169
74.50.106.162</s>
```

Exhibit 2-19 Bot configuration file from a mothership.

activity. Registrar domain takedown procedures are typically drawn out, but provide the most effective solution in stopping malicious activity related to fast-flux. New domains registered for fast-flux reuse the existing infrastructure and restart the takedown process in an endless cycle.