HIDING THE NETWORK BEHIND THE NETWORK. BOTNET PROXY BUSINESS MODEL

Alexandru Maximciuc, Cristina Vatamanu & Razvan Benchea
Bitdefender, Romania

Email {amaximciuc, cvatamanu, rbenchea}@bitdefender.com

ABSTRACT

Over the years, botnet creators have implemented various methods for protecting their networks and especially their command and control (C&C) servers. Since hiding the C&C server usually means that a botnet will remain active for longer, there exist specialized hosting services that are able to hide a server behind many proxies.

During one of our investigations, we discovered a network of this type, which at the time of writing this paper has 10 'clients' (i.e. 10 servers distributing different malware families). This proxy network has two types of redirection: one on the HTTP standard port (protecting the C&C servers) and the other on the UDP standard port (protecting a dedicated server that handles the DNS resolution for domains generated by Domain Generation Algorithms or chosen at will).

This infrastructure is designed in such a way as to allow critical changes to be made in the shortest time possible, so any abuse report regarding the proxy nodes is handled immediately. The so-called 'cleaning' is done by making some minor changes to the configuration of the proxy nodes. This is usually achieved by changing the proxies between clients. Therefore the financial loss caused by interruption of the malware distribution, is minimized.

In this paper, we will describe the architecture of this network and the changes made during the time we have been monitoring it. We will also present some examples of malware families that make use of it.

GENERAL INFRASTRUCTURE

This network model has at least two levels of proxies protecting the real C&C servers. Currently, it has 10 'clients', serving different types of malware to unsuspecting users. Each client has its own set of machines, responsible for handling the UDP traffic. They act as name servers for that specific client and its corresponding HTTP traffic. The infrastructure is designed in such a way as to hide the real C&C servers and to easily be able to apply changes, when needed.

Figure 1 shows the components of the infrastructure.

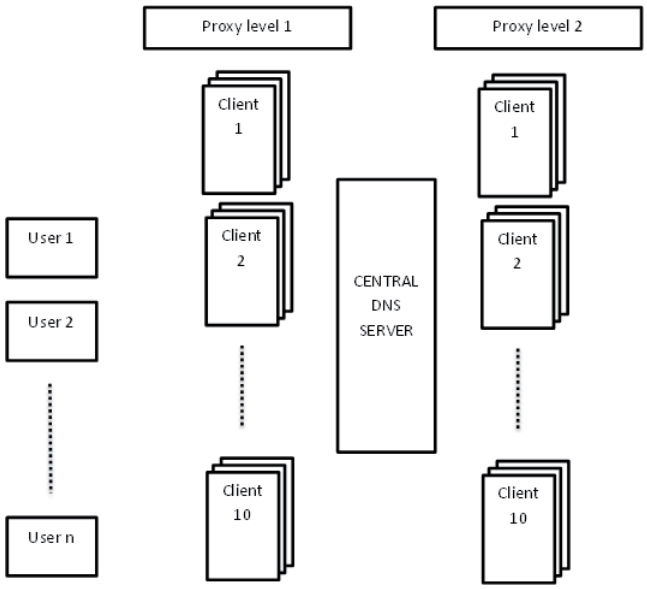

Figure 1: Network's general infrastructure.

PROXY LEVEL 1

Proxy level 1 is responsible for redirecting the UDP traffic (on port 53) and the HTTP traffic (on port 80). Figure 2 shows the

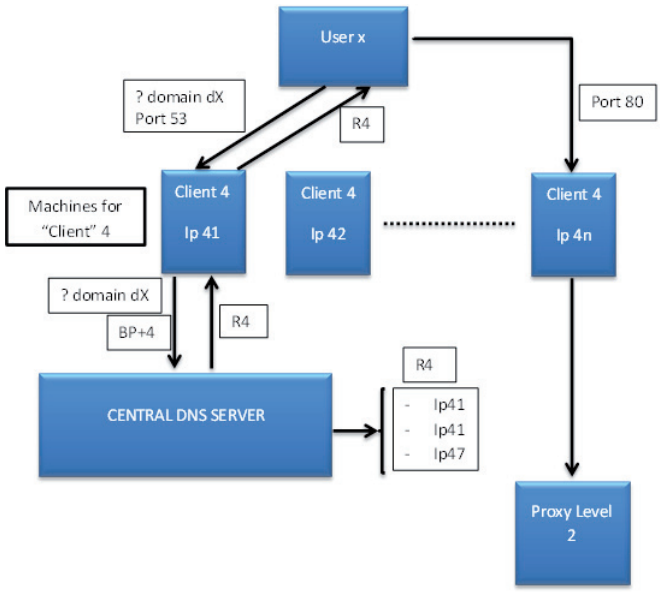

Figure 2: DNS and HTTP redirections (BP=BASE_PORT).

way in which redirection works, using the 'client 4' as an example. A user, infected with a piece of malware spread by client 4, tries to connect to a domain, dX. The machines used in the first level of the proxy are set as authoritative name servers for these domains. So, any DNS resolution request for the dX domain arrives here. All the traffic received on port 53 is redirected to a central DNS server. The port used for this redirection is BASE_PORT + client_id, where BASE_PORT is a high number. This approach enables the central DNS server to differentiate between the clients. This server responds with four active IPs, randomly chosen from the list of IP addresses allocated to the current client. The victim's computer chooses one of these IP addresses (IP 4n in Figure 2) and sends an HTTP

request to the machine corresponding to it. This machine will redirect the request on a machine from the second level proxy, usually on port 80.

At the time of writing this paper, this network has 10 clients. When we discovered the network, it had one inactive client and eight active clients, called 'ozerside', 'special', 'owner', 'nobody_xxx', 'owl', 'eclipse', 'curl' and 'victor' (see Table 1).

Around 23 November 2013, the ninth client was activated under the name 'Lee(iq)' (see Table 2), and on 20 February 2014 the 10th client was added, without a name (see Table 3).

Tables 1, 2 and 3 illustrate information displayed by the 'users.php'script located on the central DNS server.

We noticed that some changes had also been made to other clients: client 4 is now called 'demien (otkaz)' and is no longer

UID	HTTP bots	DNS bots	HTTP	Port	Comment	Action	DNS stat	Used bots
1	2 (3 up)	2 (3 up)	ip_server_1	80	ozerside	[Edit]	[Show]	26075
2	2 (1 up)	2 (2 up)	ip_server_2	18230	special	[Edit]	[Show]	28586
3	2 (2 up)	2 (2 up)	ip_server_3	80	owner	[Edit]	[Show]	25907
4	2 (2 up)	2 (2 up)	ip_server_4	80	nobody_xxx	[Edit]	[Show]	7502
5	2 (2 up)	2 (2 up)	ip_server_5	80	owl	[Edit]	[Show]	13917
6	2 (2 up)	2 (2 up)	ip_server_6	80	eclipse	[Edit]	[Show]	3866
7	2 (3 up)	2 (2 up)	ip_server_7	80	curl	[Edit]	[Show]	1136
8	2 (1 up)	2 (1 up)	ip_server_8	80	victor	[Edit]	[Show]	3778
9	0 (0 up)	0 (0 up)	8.8.8.8	80		[Edit]	[Show]	4646

Table 1: 'Clients' list when we discovered the network.

UID	HTTP bots	DNS bots	HTTP	Port	Comment	Action	DNS stat	Used bots
1	2 (3 up)	2 (3 up)	ip_server_1	80	ozerside	[Edit]	[Show]	26075
2	2 (1 up)	2 (2 up)	ip_server_2	18230	special	[Edit]	[Show]	28586
3	2 (2 up)	2 (2 up)	ip_server_3	80	owner	[Edit]	[Show]	25907
4	2 (2 up)	2 (2 up)	ip_server_4	80	nobody_xxx	[Edit]	[Show]	7502
5	0 (0 up)	0 (0 up)	ip_server_5	80	owl	[Edit]	[Show]	13917
6	2 (2 up)	2 (2 up)	ip_server_6	80	eclipse	[Edit]	[Show]	3866
7	2 (3 up)	2 (2 up)	ip_server_7	80	Curl	[Edit]	[Show]	1136
8	2 (1 up)	2 (1 up)	ip_server_8	80	victor	[Edit]	[Show]	3778
9	2 (2 up)	2 (2 up)	Ip_server_9	2224	Lee(iq)	[Edit]	[Show]	4646

Table 2: 'Clients' list late November 2013.

UID	HTTP bots	DNS bots	HTTP	Port	Comment	Action	DNS stat	Used bots
1	2 (3 up)	2 (3 up)	ip_server_1	80	ozerside	[Edit]	[Show]	3
10	0 (0 up)	0 (0 up)	1.1.1.1	80		[Edit]	[Show]	0
2	2 (1 up)	2 (2 up)	ip_server_2	18230	special	[Edit]	[Show]	5
3	2 (2 up)	2 (2 up)	ip_server_3	80	6504650	[Edit]	[Show]	10
4	0 (0 up)	0 (0 up)	1.1.1.1	80	demien(otkaz)	[Edit]	[Show]	0
5	2 (2 up)	2 (2 up)	ip_server_5	80	owl	[Edit]	[Show]	1
6	0 (0 up)	0 (0 up)	ip_server_6	80	dokben, 777	[Edit]	[Show]	0
7	2 (3 up)	2 (2 up)	ip_server_7	80	rxtitans	[Edit]	[Show]	1
8	2 (1 up)	2 (1 up)	ip_server_8	80	victor	[Edit]	[Show]	1
9	0 (0 up)	0 (0 up)	8.8.8.8	80		[Edit]	[Show]	0

Table 3: 'Clients' list late February 2014.

active; clients 6 and 7 changed their names to 'dokben, 777' and 'rxtitans', respectively.

With the help of some local authorities we managed to analyse some data. An encrypted binary file (ELF) named 'map' gave us an insight into the network. The decryption uses the waterfall technique: 13 areas are decrypted, and each decryption depends on the previous one through an index table.

The last area is a bash script that has two main functionalities:

1. Self-update
2. Update for service.xml, the service responsible for traffic redirection.

The self-update procedure is very simple. The updated binary file is downloaded via a request to http://IP_MAP_UPDATE/update. Integrity checks are performed using MD5 hashes. The MD5 hash for the binary file is also retrieved from the same link. The bash script checks the data consistency (that is, whether the computed MD5 for the freshly downloaded file is the same as the retrieved one). The next step is to find out whether an update is necessary. If the MD5 for the downloaded binary file differs from the MD5 for the existing binary file, then the old version is replaced with the new one and the process is restarted.

The second functionality is very similar. Through a request to http://IP_DNS_SERVER/<file1>?ip=<currentMachineIP>, an xml file is downloaded and saved, temporarily, as 'service.xml'. The file contains rules for UDP and HTTP forwarding, as can be seen in Figure 3. The IP used for this update is the IP for the central DNS server. The xml file is parsed, the extracted 'tunnelling' rules are applied in the system, and the services affected by those rules (nginx and 3proxy) are restarted.

The structure of the service.xml file is illustrated in Figure 3.

We noticed that every UDP request is redirected to the central DNS server. The UDP traffic is forwarded to the port BASE_PORT + client_id and, in this way, the DNS server can identify which client sent the request.

The DNS server will respond with a list of IPs corresponding to the identified client, a list of four IPs, not necessarily unique and not necessarily the same for different queries (as mentioned previously, a random list of active IPs). The infected computer will choose one of the IP addresses and send an HTTP request that will be redirected by the level 1 proxy to the IP specified in service.xml for the HTTP protocol. Usually, the value for 'to_port' is 80, but it can vary (for example, 'client 2' was assigned an HTTP port with value 18230), as we can see from the clients management interface illustrated in Figure 4. The HTTP traffic is actually redirected to an IP from the second proxy layer.

Central DNS server

The DNS server has three main roles: it resolves DNS queries, as discussed in the 'Proxy Level 1' section; it serves updates for service.xml; and it represents the management interface for all the clients.

During our investigation we have been monitoring the 'map' file updates for proxy level 1 machines, and we have observed the DNS server being moved from one machine to another. With the help of the local authorities, we managed to gather information about the old server. A collection of PHP scripts was analysed, most of which were related to a management interface.

```
<?xml version="1.0" encoding "UTF-8" ?>
<tunnel>
      <tunnel from='http' from_port='80' to='http' to_port='80'>IP_PROXY_LEVEL_2</tunnel>
      <tunnel from='udp' from_port='53' to='udp' to_port='BP+client_id'>IP_DNS_SERVER</tunnel>
</tunnel>
```

Figure 3: Service.xml file (BP=BASE_PORT).

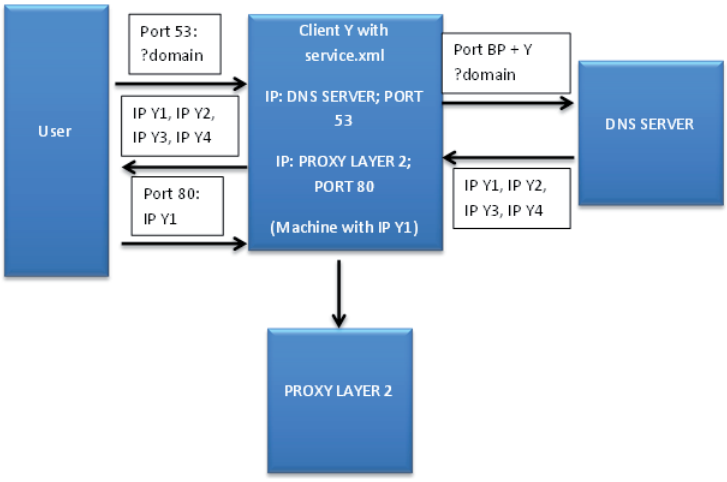

Figure 4: Traffic tunnelling (BP=BASE_PORT).

These files were divided in three main categories: admin, checker and system, as can be seen in Figure 5.

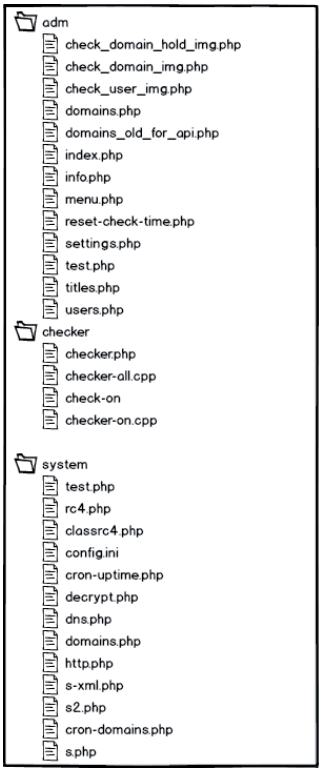

Figure 5: Directory tree from central DNS server.

Admin

Most of the PHP scripts from the admin section have the role of managing the system by implementing the CRUD operations (create, read, update and delete) for all the entities used. The most important PHP scripts are 'domain.php', 'user.php' and 'index.php'.

The 'index.php' script (Table 4) handles the data for the servers corresponding to the proxy level 1 machines. It allows the deleting, editing and displaying of different information regarding the servers corresponding to the infected bots, such as their IP addresses, the country where they are located, the period of time elapsed since their last valid response, clients they are assigned to, etc. Table 5 shows a snapshot of the information displayed.

The 'users.php' script is used to add, edit or show information regarding the clients from the network. The parameters are described in Table 6. Each client has a corresponding file ('users/%uid%', where uid is the client id). These files contain the client's name, the IP for the second-level proxy server that handles the HTTP traffic for that client, the port for the HTTP communication, and so on. This script uses and saves information in these files and in the MySQL database in the 'servers' table.

The 'domain.php' script is used to manage the data regarding the assigned domains (Table 7). It allows the deleting, adding and displaying of information about them. The registration of a domain is done automatically through the cnobin.com service. Table 8 shows the information displayed on the web page by domains.php.

Checker

These scripts are responsible for checking some of the characteristics of the servers from the proxy level 1 network, such as the state (on/off) and the speed (Table 9). These characteristics are updated in the 'servers' table in the MySQL database. For example, 'checker.php' checks whether the bots are responding in a specific way. If yes, they are 'on'; otherwise they are 'off'.

index.php	
Command	**Description**
Del	Delete IP from servers.
Edit	Edit IP from servers.
<without parameters>	Prints the number of online bots and the number of active HTTP and DNS servers. For each IP it shows: country, IP, HTTP, DNS, speed, ping, loss, UpTime, LastCheck, Other, UID, actions.

Table 4: Parameters for the index.php script.

Country	IP	HTTP	DNS	Speed	Ping	Loss	UpTime	LastCheck	UID		Action
Ukraine	IP1	off	off	0	0	0	21 days 21 hours 7 mins 54 secs	2013-11-22 14:50:47	6	[Delete]	[Edit]
Russia	IP2	on	on	264	93	0	18 days 34 mins 20 secs	2013-11-22 14:52:05	7	[Delete]	[Edit]

Table 5: Information about servers from the first level of proxy.

users.php	
Command	**Description**
edit<sent>	Parameters: HTTP (no. of HTTP bots), DNS (no. of DNS bots), IP, port and comment, received through POST, are saved in users/%uid% (where %uid% is the client ID received through GET) and showed on the web page.
edit	Gets the previously mentioned parameters for the received %uid%, shows them on the web page, and allows their actualization.
add<sent>	Parameters: HTTP (HTTP bots), DNS (DNS bots), IP, port and comment, received through POST, are saved in users/%uid% and showed on the web page.
add	Displays, on the web page, a form that allows the saving of information for the new client.
dns-stat	Displays, on the web page, statistics for a certain client for the current day and month.
<without parameters>	Shows on the web page information for all the clients.

Table 6: Parameters for the users.php script.

domain.php	
Command	**Description**
Del	Deletes the domain.
add<sent>	If the 'sent' parameter is used, it will register domains through cnobin.com and insert the data (domain, uid, type, ns1, ns2, ns3, ns4) into the database (domains table).
add	Displays a form that allows data to be submitted as domain, uid, ns1, ns2, ns3, ns4.
<without parameters>	Displays information about all the domains registered in the database (domain table).

Table 7: Parameters for the domains.php script.

Domain	**80 port**	**Holding**	**UID**	**Type**	**NS**	**Action**
domain_1	On	On	2	2	ns1: ip_1 ns2: ip_1 ns3: ip_1 ns4: ip_1	[Del]
domain_2	On	On	2	2	ns1: ip_2 ns2: ip_2 ns3: ip_2 ns4: ip_2	[Del]
domain_3	On	On	2	2	ns1: ip_3 ns2: ip_3 ns3: ip_3 ns4: ip_3	[Del]

Table 8: Information about the registered domains.

checker.php	
Column (servers)	**Setting conditions**
http	[on/off] if a certain request answers with a message containing %http_test_msg%, then it sets the column to on, otherwise it is set to off.
dns	[on/off] if a certain request answers with a message containing %dns_test_ip%, then the column is set to on, otherwise it is set to off.
http_good	If http is [on], and the values for ping and speed are within a certain range (ping and speed are computed with Net/Ping.php), then the column is set to [on], otherwise it is set to [off].
dns_good	If dns is [on], and the values for ping and speed are within a certain range (ping and speed are computed with Net/Ping.php), then the column is set to [on], otherwise it is set to [off].

Table 9: Parameters for the checker.php script.

System

Here, we found scripts for RC4 encryption and decryption, config files, and scripts that delete from the database the servers that have an 'expired' LastCall (where the time elapsed since the last valid response is greater than a certain timeout).

We also found the template file for the service.xml update (Figure 6).

config.ini

```
<tunnels>
   <tunnel from='http' from_port='80' to='http'
to_port=%http_port%><%http_ip%></tunnel>
   <tunnel from='udp' from_port='53' to='udp'
to_port=%dns_port%><%dns_ip%></tunnel>
</tunnels>
```

Figure 6: Template for the service.xml file.

Abuse reports

This complex network architecture proves to be very effective in the case of abuse reports. We submitted two types of abuse report, and on each occasion the network recovered very quickly.

The first type of abuse report was related to the servers from the first proxy level. This issue can be resolved only by switching between users' IP lists. This change is made in the central DNS server, in the 'servers' table of the MySQL database. An update for the service.xml file is also provided to the affected machines in order for them to change IP addresses for the HTTP redirection (Figure 7).

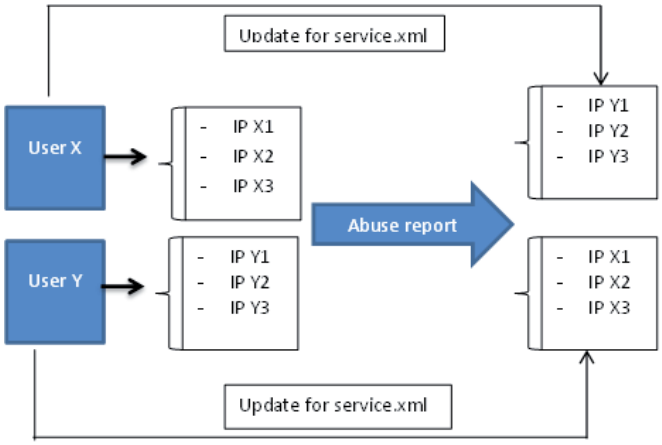

Figure 7: Solution for recovering from abuse reports.

The second type of abuse report is related to the IPs corresponding to the second level of proxies. The events we observed in this case did not last more than 24h, so the downtime for the malware is very small. Usually, after the abuse is reported, the machine is stopped (it doesn't respond as it should), but after approximately three to four hours, a new IP appears in the system and, in less than 24 hours, the malware is back in business. The changes needed consist of modifying some data from the central DNS server and a service.xml file update. On the central DNS server, in the client's corresponding file ('users/%uid%'), the old IP is replaced with the new one. Also, all the clients from the first-level proxy, that redirected the HTTP traffic to the affected IP, will receive a new service.xml file, updated with the new IP.

Examples

Malware families served by these clients include CryptoLocker, Citadel, FakeAVs, Casino, and various scams.

Statistics

This network is very dynamic – IP addresses are moved from one user to another, new IP addresses are added, and old IP addresses are removed. The charts shown in Figures 8 and 9 illustrate the IP changes that occurred on the first proxy level during the period in which we were monitoring the network.

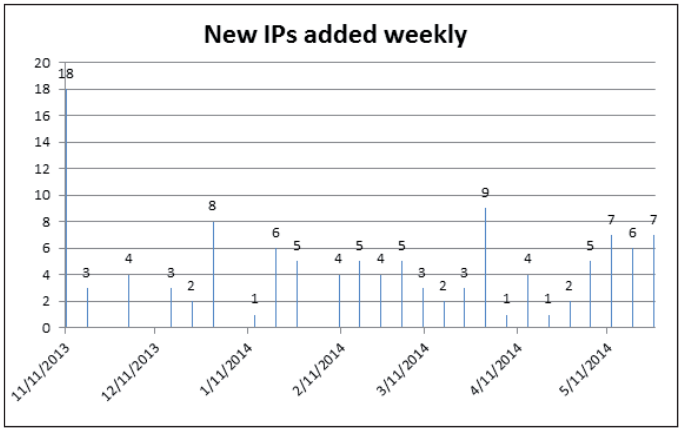

Figure 8: Number of IPs added weekly.

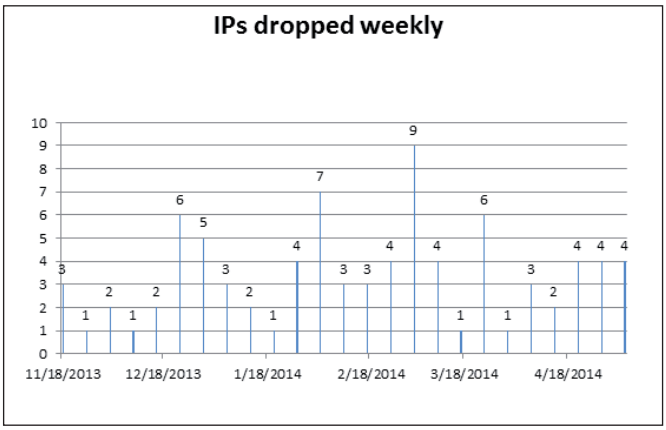

Figure 9: Number of IPs removed weekly.

A map for the proxy level 1 IP addresses in November 2013 is illustrated in Figure 10. There were four serves in the Ukraine, 12 in Russia, and five in Kazakhstan.

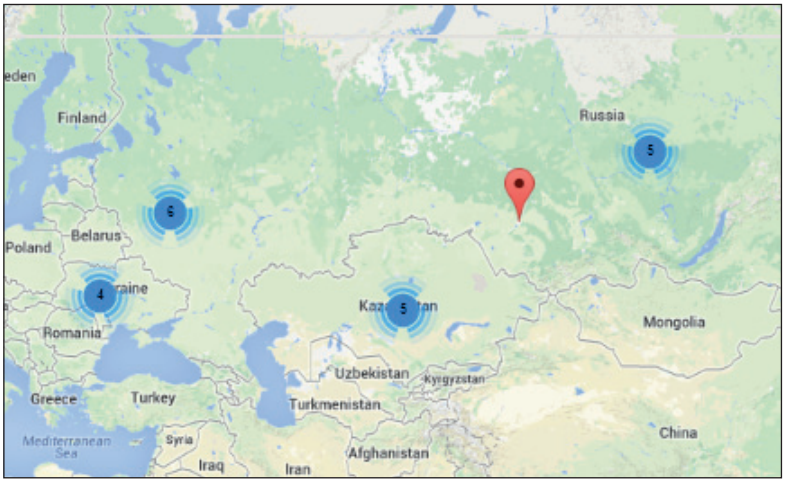

Figure 10: Servers added in November 2013.

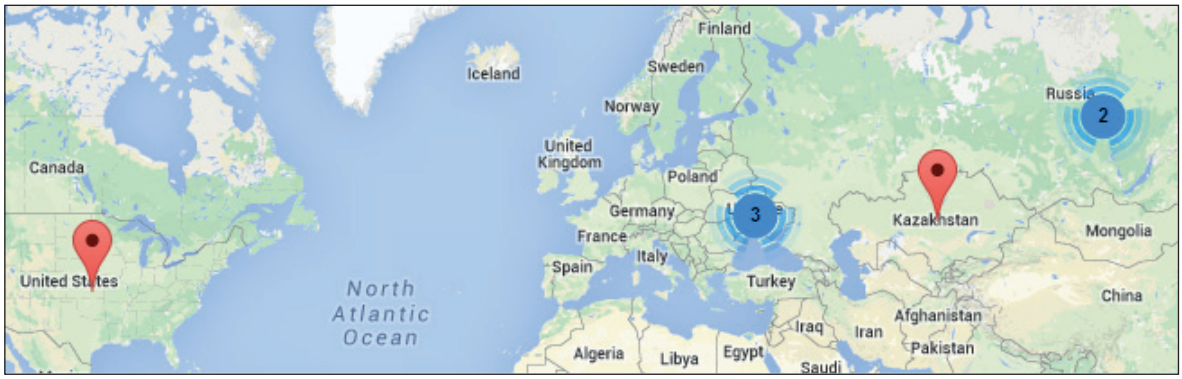

Figure 11: Servers added in January 2014.

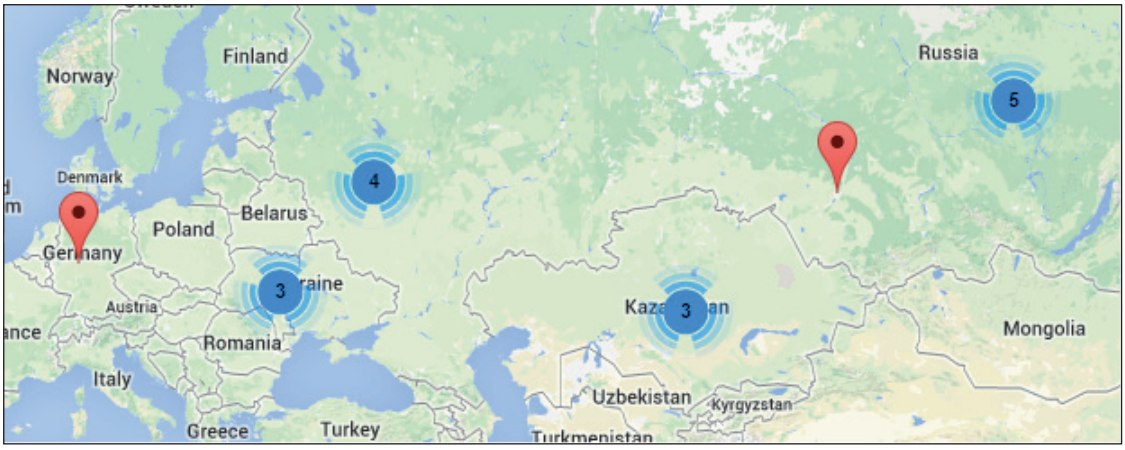

Figure 12: Servers added in February 2014.

In January 2014, they extended their business to the United States (Figure 11). In February 2014, they added a server in Germany (Figure 12). In May 2014, a new server appeared in the Netherlands (Figure 13).

A map illustrating all the IPs which passed through the system during our investigation is shown in Figure 14.

CONCLUSION

This network has proven to be well thought out and maintained. The structure hides the real C&C servers very well and has the ability to recover from abuse reports in less than 24 hours. It has a great dynamic, adding new servers and removing old ones almost every day. It also seems to be very well known among

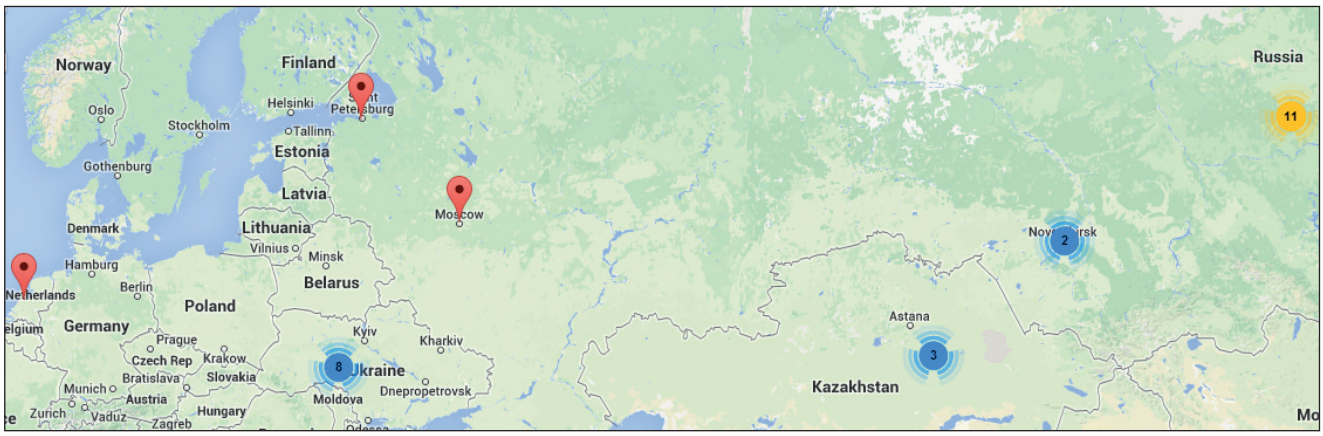

Figure 13: Servers added in May 2014.

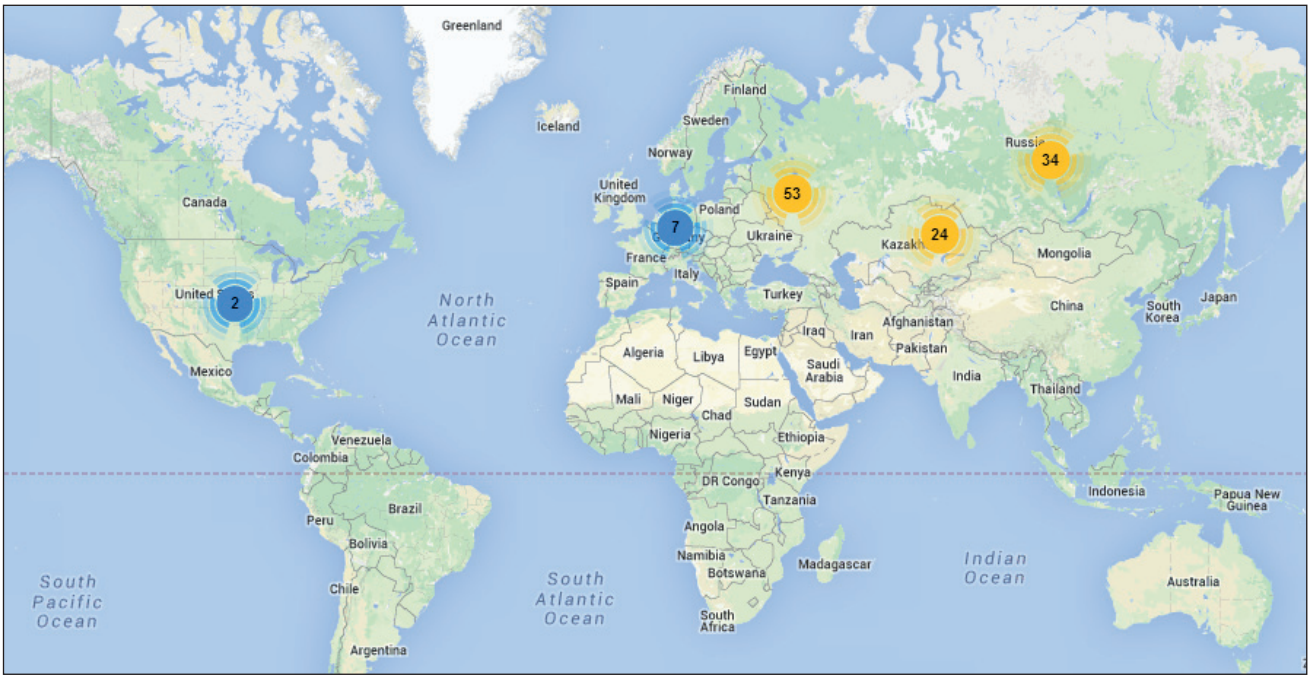

Figure 14: Servers map from November 2013 to May 2014.

malware creators, who use if for distributing various types of malware.

We are currently continuing our investigation in order to find all the nodes of the network with the aim of taking them down.