Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Cylindrical Coordinates Security Visualization for multiple domain command and control botnet detection

Ilju Seo [a], *Heejo Lee* [b], *Seung Chul Han* [c,*]

[a] *Secugraph Inc., Seoul, South Korea*
[b] *Department of Computer Science and Engineering, Korea University, Seoul, South Korea*
[c] *Department of Computer Engineering, Myongji University, Yongin, South Korea*

ARTICLE INFO

Article history:
Received 29 March 2014
Received in revised form
22 June 2014
Accepted 28 July 2014
Available online 12 August 2014

Keywords:
Security visualization
Botnet detection
DNS traffic
Human cognition
Graph isomorphism
Visual signature

ABSTRACT

The botnets are one of the most dangerous species of network-based attack. They cause severe network disruptions through massive coordinated attacks nowadays and the results of this disruption frequently cost enterprises large sums in financial losses. In this paper, we make an in-depth investigation on the issue of botnet detection and present a new security visualization tool for visualizing botnet behaviors on DNS traffic. The core mechanism is developed with the objective of enabling users to recognize security threats promptly and mitigate the damages by only visualizing DNS traffic in cylindrical coordinates. We compare our visualization method with existing ones and the experimental results show that ours has greater perceptual efficiency. The ideas and results of this study will contribute toward designing an advanced visualization technique that offers better security. Also, the approach proposed in this study can be utilized to derive new and valuable insights in security aspects from the complex correlations of *Big Data*.

1. Introduction

A *bot* is a software application that an attacker, a.k.a. *botmaster*, controls over the Internet through *command and control* (C&C) infrastructures. *Botnet* refers to a group of hosts infected with a bot and communicate with a botmaster to perform coordinated cybercrimes, such as distributed denial of service (DDoS) attacks, spamming, scanning, or phishing. Botnets are regarded as one of the most dangerous species of network-based attack today because they can cause severe network disruptions through massive coordinated attacks, and the threat of this disruption can cost enterprises large sums in financial losses. Recent reports show that, over 9 million computers connected to the Internet in a botnet in year 2012 and caused monetary damage to the extent of $500 M Wyke (2012); Kovacs (2013).

While there are numerous types of botnets known, most of them are controlled by remotely through C&C infrastructures. In botnets that use the chat style of command and control, botmaster and infected hosts subscribe to the same C&C server. The botmaster issues commands to the bots via the server, which is usually an IRC server. The C&C server may or

* *Corresponding author*.
E-mail addresses: iljuseo@secugraph.com (I. Seo), heejo@korea.ac.kr (H. Lee), bongbong@mju.ac.kr, dr.seungchul@gmail.com (S.C. Han).

http://dx.doi.org/10.1016/j.cose.2014.07.007
0167-4048/

may not be a compromised machine.[1] Since infected hosts are controlled remotely through C&C infrastructures, the key issue of a botmaster is how to rally the bots to the C&C server.

In the early forms of botnet, the fixed IP address or domain name of C&C server was hard-coded to the binary codes of bots and all the bots communicate with the assigned C&C server to receive and execute the commands from botmaster and to pass the harvested results. In such cases, it is easy to detect bots by their communications with hosts whose IP address (or domain name) is on the blacklists and to take down a particular botnet by blocking this particular address of C&C server. Even though the address of C&C server is obfuscated to prevent reverse engineering analysis, the hard-coded address is unchangeable, it cannot provide any mobility. If a C&C server is not mobile, a single alarm or misuse report can provoke the C&C server to be quarantined or the whole botnet suspended.

In order to overcome this drawback, recent botnets have been based on a different approach. A botmaster arranges several C&C servers and frequently switches to different domains to make it difficult to keep blacklists up-to-date. Bots use a technique called *domain flux*, such as *domain generation algorithm* (*DGA*), and *dynamic DNS* (*DDNS*) to search a C&C server instance. We call these kinds of botnets as *multiple domain C&C botnets*. Unlike early botnets, they do not have a static single point of failure, and can plan and operate themselves even when their C&C servers are blocked. Moreover, they are very difficult to be detected because of the characteristics of the domains generated by the DGA, which only last for a short period Stone-Gross et al. (2009). These techniques make the multiple domain C&C botnets intelligent and hard to be handled through existing security tools. Examples of multiple domain C&C botnet include *Rustock* (a.k.a. *Spambot*) Chiang and Lloyd (2007), *Conficker* (a.k.a. *Downadup* or *Shadow*) Porras et al. (2009), and Torpig Stone-Gross et al. (2009).

The botnet detection is a very active research domain, and it has gained a lot of attention in recent years. Many valuable initiatives exist for capturing or monitoring botnet activities Dagon (2005); Gu et al. (2007); Villamarín-Salomón and Brustoloni (2008); Gu et al. (2008); Fink et al. (2004); Abdullah et al. (2005); Ren et al. (2006). Most of existing detection mechanisms focused on a particular symptom of botnets or a signature of bot programs. Collecting data related to Internet threats has thus become a common task for security managers or network operators. However, effectively analyzing the vast amounts of data collected proves challenging because it is very resource-intensive and scalability is a growing problem. The volume and the diversity of the raw data and computations involved can rapidly overwhelm people or system in charge of analyzing those data sets. Hence, it is hard to quickly identify and respond to the threats posed by a multiple domain C&C botnet in time, allowing the botnet to cause considerable damage. This fact motivates us to study a new approach for multiple domain C&C botnet detection.

In this paper, we note the benefits of analyzing DNS traffic to detect botnets and notice that the DNS querying relationship between bots and C&C servers in a multiple domain C&C botnet can be formulated as a *bipartite isomorphism problem* which is *graph isomorphism complete*. Then, we present a new security visualization tool, *CCSvis* (*Cylindrical Coordinates Security Visualization*). Our goal is to intuitively recognize symptoms of botnet activities by only visualizing DNS traffic. In order to successfully achieve this objective, we introduce two innovative features. First, we discuss the problems faced in existing visualization mechanisms and the design criteria for solving these problems. The proposed mechanism is designed by considering not only the system-oriented perspective but also the human-oriented perspective. Unlike other related works, CCSvis does not use any botnet detection algorithm, but it enables human to recognize a botnet, by visualizing the complex correlation structure of botnets. Second, CCSvis employs a new mechanism that visualizes botnet behaviors and defines a variety of threat patterns on cylindrical coordinates. We categorize them into five visual signatures for interpretation of threats and adapt a human-cognitive approach which is better at catching user's attention, makes it easier to perceive the salient parts of the graph, making it easier to detect botnets. We compare CCSvis with four other generic types of visualization for two multiple domain C&C botnets, *Conficker* and *Rustock*. The experimental results show that our method generates a larger, more focused area of saliency, it has greater perceptual efficiency and thus is more effective and intuitive for botnet detection.

The rest of the paper is organized as follows. In Section 2, some useful preliminaries are provided. In Section 3, we review related works on visualization techniques and botnet detection. In Section 4, we present a new tool for visualizing botnet behaviors. We evaluate and compare it with other visualization techniques through experiments in Section 5. Finally, the conclusions are drawn in Section 6.

2. Preliminaries

2.1. DNS traffic analysis for botnet detection

In a multiple domain C&C botnet, botmaster wants its C&C servers to be invisible and portable. By using DNS services, the botmaster can hide and frequently switch it's server without modifying bot code. As a result, each bot periodically generates many DNS queries to locate its C&C server or to lookup victims. Following three cases show the situations of the DNS queries generated in botnets: (1) C&C server rallying process, (2) attacking process: Some attacks such as DRDoS attack and spam mailing are accompanied with the DNS transmit, (3) C&C server migration: Botmaster migrate one to another server, the corresponding IP address or domain name can be changed at any time. If the IP address or domain name of the C&C server changed, the bots cannot connect the old one, so they generate the DNS query to access new server.

The DNS queries of multiple domain C&C botnets are distinguishable from legitimate DNS queries as follows: (1) most of legitimate DNS queries occur continuously and do not occur simultaneously but DNS queries from bots occur temporary and simultaneously, repetitive queries may indicate bots, (2) the size of botnet is normally fixed, on the other hand, the size of clients of legitimate DNS query is quite variable and

[1] There are many public IRC servers that host unmonitored channels.

irregular, (3) the botnet uses DDNS for C&C server usually, but legitimate cites do not commonly use DDNS. Existing methods distinguish legitimate DNS queries from the botnet queries by using various types of metrics, e.g., the amount of DNS queries going outside the local networks, the frequency of dynamic DNS usage, similar DNS behavior of host groups etc Kovacs (2011).

The benefits of analyzing DNS traffic are many. For instance, (1) monitoring DNS traffic has less overhead than monitoring the entire network traffic, (2) the coordinated DNS transmission is one of the most frequently observed group activities in a botnet lifecycle, and (3) DNS monitoring enables botnet detection at its early stage because the DNS traffic is generated when bots contact C&C servers prior to launching malicious activities.

2.2. Graph isomorphism and multiple domain C&C botnet detection problem

Two graphs $G = (V, E)$ and $G' = (V', E')$ are isomorphic if and only if there is a one-to-one mapping $\varphi: V \rightarrow V'$ such that $(u, v) \in E$ if and only if $(\varphi(u), \varphi(u)) \in E'$ for every pair of vertices u, $v \in V$. We denote by $G \sim G'$ if G and G' are isomorphic. The *bipartite isomorphism problem* is to determine if $G \sim G'$ for given bipartite graphs G and G'.

Lemma 2.1. *The bipartite isomorphism problem is graph isomorphism complete* Uehara et al. (2005).

Considering the features of the multiple domain C&C botnet DNS queries, we can discover the existence of a multiple domain C&C botnet as follows. Let $C_{t_i} = \{c_1, \ldots, c_n\}$ be a set of clients sending DNS queries and $D_{t_i} = \{d_1, \ldots, d_m\}$ be a set of domains queried by $c \in C_{t_i}$, during a certain period of time t_i. The DNS querying relationships between clients and domains can be represented as a bipartite graph, $B_{t_i} = (C_{t_i}, D_{t_i}, E_{t_i}), E_{t_i} = \{(u, v) | u \in C_{t_i}, v \in D_{t_i}\ v \text{ is queried by } u\}$. Suppose a subgraph $B'_{t_i} = (C'_{t_i}, D'_{t_i}, E'_{t_i})$ of B_{t_i} and a subgraph B'_{t_j} of $B_{t_j}, (i \neq j)$. If $B'_{t_i} \sim B'_{t_j}$, then it is very likely that the clients of C'_{t_i} are infected by a bot. Hence, the botnet detection is equal to find $B'_{t_i} \sim B'_{t_j}$ for a certain time period of $i,\ldots, j$, and it is at least as hard as the bipartite isomorphism problem by Lemma 2.1, therefore, there is no polynomial time algorithm known.

2.3. Security visualization and human visual system

Even today, an automated system cannot replace human expertise completely in network control which is required for various security reasons. Many administrators are faced with the problem of analyzing the huge amounts of real-time data being generated, but they are often not even aware of security threats.

Security visualization uses graphical approaches to help a user easily understand a large amount of abstract data and intuitively perceive the situation Conti and Abdullah (2004). It also helps to identify types of threats and mitigate the damage caused by them Choi et al. (2007). The advantages are derived from the biological structure of human brain. As shown in the results of neuroscience studies Resko et al. (2006); Born and Bradley (2005); Moran and Desimone (1985), a large portion of the primary neocortex appears to be directly devoted to and specialized in visual processing. In other words, human can rapidly process visual information. According to the case of CAPTCHAs (Completely Automated Public Turing test to Tell Computers and Humans Apart) Von Ahn et al. (2003), human being's capability for the visual pattern processing still exceeds that of computers. All these studies highlight the necessity and powerful capability of *human visual system*[2] (HVS).

In this study, we consider the aspects of visual processing according to the neuroanatomical structure of the human brain and propose a new visualization tool that enables the detection of botnets in a cooperative manner between human and computer. As mentioned in the previous section, multiple domain C&C botnet detection problem is very difficult and theoretically *graph isomorphism complete*, our tool does not use any detection algorithm, but it helps human to recognize botnets intuitively by visualizing the complex correlation structures of botnet.

3. Related works

3.1. DNS based botnet detection techniques

In this section, we review previous works on DNS based botnet detection techniques.

Salomon et al. (Villamarín-Salomón and Brustoloni, 2009) proposed and evaluated a Bayesian approach for bot detection based on the similarity of their DNS traffics to that of known bots. The basic assumption of the approach is that bots in the same botnet generate similar DNS traffics which are distinguishable from legitimate DNS traffic. Similar work was also presented in Ishibashi et al. (2012). However, the true positive rate (TPR) and the false positive rate (FPR) largely depend on the thresholds that dynamically change. It is difficult to find the proper thresholds.

Manasrah et al. (2009) proposed a DNS based mechanism that captures botnet group activities from DNS traffic. However, their approach has limited coverage because they use a MAC address as an identifier of a host rather than an IP address. The MAC address is visible only to hosts on the same subnet. Therefore, it is not appropriate for large-scale networks.

Brustoloni et al. (2009) described NDS Flagger, a device for ISP bot detection. DNS Flagger matches subscribers DNS traffic against IP and DNS signature with the IP addresses and domain names of blacklisted servers.

Antonakakis et al. (2010) proposed Notos, a dynamic reputation system for DNS that uses passive DNS query data and analyzes the network and zone features of domains. It builds models of known legitimate domains and malicious domains, and uses these models to compute a reputation score for a new domain indicative of whether the domain is malicious or legitimate.

Bilge et al. (2011) uses passive DNS analysis, examines a wide set of DNS traffic features and incorporates machine learning techniques.

[2] It is a part of the central nervous system, which is responsible for visual processing. The HVS requires attention in the visual search process. Attention is the process of selectively focusing on one of the multiple stimuli while ignoring other stimulus.

Unlike ours, these works only focus on the system-oriented perspectives, they require up-to-dated blacklists and database for network information or modeling to detect botnets. However, considering the amount of calculations involved and the high dependency on the blacklist, they do not seem practical solutions.

3.2. *Security visualization techniques*

Security visualization can be viewed from two different perspectives: *visualization methods* and *visualization data*. Visualization methods are for the perspective of "*how to show*" and visualization data are for the perspective of "*what to show*" (See Table 1).

Color map: A color map can be used for showing time-series data, but it is not effective for showing relational data. With the passage of time, too much information is displayed on the screen in various colors and sizes, and this can distract users, in turn leading to a decrease in detection efficiency Colombe and Stephens (2004); Samak et al. (2008).

Node-link graph: A node-link graph can be used to effectively visualize relationships using lines that represent inter-relationships between nodes. However, it suffers from edge-crossing[3] and information loss problems that are caused by the overplotting of connecting lines among nodes Iliofotou (2009); Ball et al. (2004).

Histogram: A histogram is a well-known visualization method that visualizes information using a color spectrum. It

Table 1 – Comparison of visualization techniques.

Categories	Techniques	Pros	Cons	Related works
Visualization methods	Scatter plot	• Useful to discover outliers and anomalies.	• Improper to represent relational data • Information loss due to over-plotting.	Suo et al. (2008)
	Tree map	• Useful to represent a hierarchical data	• Improper to represent multi-dimensional data or large-scale data	Fischer et al. (2008)
	Color map	• Useful to represent a time-series data	• Improper to represent relational data • Causing distraction (dazzle effect)	Samak et al. (2008)
	Node-link graph	• Useful to represent inter-relationships between nodes.	• Causing edge-crossing • Information loss due to over-plotting	Iliofotou (2009);
	Histogram	• Useful to represent discrete data	• Improper to represent multi-dimensional data • Information loss due to categorization	Muelder et al. (2006)
	Parallel coordinates	• Useful to represent a multidimensional data	• Causing edge-crossing • Information loss due to over-plotting	Li and Luo (2009);
Visualization data	Network-data based	• Useful to represent a global view of network	• Large amount of data and calculations involved	Choi et al. (2009)
	Security-related log based	• Useful to detect anomalies	• High false positive and negative	Abdullah et al. (2005)

3.2.1. *How to show: visualization methods*

There are various types of visualization methods; however, most existing security visualization mechanisms tend to use existing visualization methods to modify or combine with them. We thus address the six most widely used and distinct visualization methods among them. Visualization methods can be classified in to six categories; *scatter plots, tree maps, color maps, node-link diagrams, histograms*, and *parallel coordinates*.

Scatter plot: A scatter plot displays information using nodes having various sizes, shapes, and colors. It is useful for detecting outliers and anomalies from a large data set, but it is inconvenient for displaying many objects on a screen because this may cause overplotting, in turn leading to information loss. Moreover, it is difficult to represent the relationships among nodes using a scatter plot Suo et al. (2008).

Tree map: A tree map represents hierarchical information using a group of tiles having various sizes and colors. It offers a global view of data having certain inter-relationships. However, it is inconvenient for visualizing complex multidimensional data as well as large-scale data such as IP addresses and port numbers. Thus, it is often used in conjunction with other visualization methods Williams et al. (2008); Fischer et al. (2008).

is widely used and includes relatively low visual noise, but it is not effective for representing multidimensional data Erbacher and Garber (2004); Muelder et al. (2006).

Parallel coordinates: Parallel coordinates align multiple axes parallel to each other with a vertical or horizontal orientation. They can be used to represent multidimensional data, and they have been used for visualizing attack patterns in many security visualization mechanisms Krasser et al. (2005); Li and Luo (2009); Choi et al. (2009). However, It, too, suffer from edge-crossing and information loss problems that are caused by overplotting.

3.2.2. *What to show: visualization data*

Data types can be classified into two categories, *network-data* and *security-related logs*, depending on whether or not data is processed by the security system.

Network-data based approaches: Network Eye Fink et al. (2004) is a visualization tool developed for network administrators. VISUAL Ball et al. (2004) used as a prototype of Network Eye is useful for displaying network communications or

[3] It is a problem caused by intersecting multiple lines, which is one of the most important criteria for graph theory.

interactions between external and internal networks, which are otherwise difficult to represent because they require many external host symbols. PCAV Choi et al. (2009) uses parallel coordinates for visualizing network data. It forms visual signatures that represent each type of network attack from the collection of lines plotted on multiple vertical axes. NFlowVis (Fischer et al., 2008) shows the visualized result of netflow analysis using a relational database. It uses a tree map to visually represent a massive distributed SSH attack and low-volume attack pattern and to analyze service usage.

Security-related log based approaches: IDS RainStorm Abdullah et al. (2005) visually represents network conditions on an x–y plane using IDS events. The x-axis represents the source/destination IP addresses whereas the y-axis represents the time information. Ren et al. (2006) proposed a technique that visualizes blacklisted domain name querying behavior and abnormal querying frequency overtime from DNS query logs using animated text having various colors.

4. Cylindrical Coordinates Security Visualization

4.1. *Visualization parameters*

There are two types of DNS traffic: query and response (Mockapetris). We only consider DNS response packets because they contain DNS query data as well as additional reply section. We extract the following five fields from each DNS response packet (Table 2).

Client IP address. This is an unsigned 32-bit value in the `IP Header Section`, which contains the IP address of the client sending the DNS query.

Query type. This is an unsigned 16-bit value for the `QTYPE` field in the `DNS Question Section`, which specifies the type of query. This parameter is used to identify the type of behavior.

Domain name. This is the variable-length string for the `QNAME` field in the `DNS Answer Section`. This parameter is used to identify the botnet's C&C server or the domain name of the target.

Timestamp. This is an unsigned 32-bit value for the response time captured from the DNS server, which is used to measure the intensity of queries created by a client. However, this value requires too many resources to update every second. Accordingly, we only use the time variation $\Delta t_i(c)$ for a client c between the initial time $t_0(c)$ and the certain time $t_i(c)$ to reduce the variable size as follows:

Table 2 – Symbols of visualization parameters.

Symbol	Description
Δt_c	Timestamp of the queries sent by client c
$\lambda(c)$	Average number of queries per second sent by client c
$F(c)$	Flag error rate of the client c
$\lvert\varepsilon_c\rvert$	Number of flag errors that occurred by client c
$\lvert\forall\varrho(c)\rvert$	The number of entire queries from client c
$\mathscr{C}(d)$	Set of client IP addresses that querying domain name d
$\tau_{\mathscr{N}}$	Threshold by network scale
$\tau_{\mathscr{C}}$	Threshold for number of queries
$\tau_{\mathscr{F}}$	Threshold for flag error rate
$\tau_{\mathscr{I}}$	Threshold for query intensity

$$\Delta t_i(c) := t_i(c) - t_0(c)$$

Flags. This is a 16-bit value in the `DNS Header Section`, which consists of fields that containing state information. We only used the lower 4 bits of the flags, which indicate `RCODE`. `RCODE` is used to indicate whether a query was answered successfully or not. Recent botnets use DGA, which is extremely location-resilient Stone-Gross et al. (2009); Porras et al. (2009); Yadav et al. (2010). Bots may send many malformed DNS queries using DGA. Thus, this parameter is used to measure the error rate in order to find a botnet or to detect *DNS cache poisoning* attack.

In addition, we derive three features from above five fields, such as *cardinality*, *intensity*, and *flag error rate*.

Cardinality. This value indicates the cardinality of a set of the client IP addresses that queries a specific domain name. Botnets, unlike normal clients, maintains relatively constant cardinality over a period time. Thus, we can visually cluster botnets by cardinality to detect botnet. Let d be a domain names and $C = \{c_1, c_2, \ldots, c_n\}$ be a set of client IP addresses that queries the domain name d. Then, we define the cardinality of d as follows:

$$|\mathscr{C}(d)| \doteq n$$

Intensity. This value indicates the average number of queries sent by a client per second. Malicious behaviors such as spam sending, DNS cache poisoning attack, and *distributed reflection denial-of-service* (*DRDoS*) attacks generate many DNS packets in a short period. Thus, we can measure the intensity to identify a client showing abnormal behavior. The average number of queries sent by a client per second is denoted as follows:

$$\lambda(c)$$

Flag error rate. If an attacker performs *DNS cache poisoning* attack, many error flags will occur. The flag error rate of a client is thus used to detect an attack or malicious behavior. The flag error rate is determined as follows:

$$F(c) := \frac{|\varepsilon(c)|}{|\forall\varrho(c)|}$$

where, $|\forall\varrho(c)|$ denotes the number of entire queries from client c, and $|\varepsilon(c)|$ means the number of flag errors in the response to the queries.

4.2. *Visualization mechanism*

Most existing security visualization mechanisms represent a large range of values (e.g., IP address) on the linear axis or plane. However, representing such a large range of values on a linear axis or plane is ineffective because it is difficult to distinguish each IP address due to the limited display resolution. To solve this problem, we indicate each client IP address on a circle. Each domain name queried by a client is located at the center of the circle. However, if all domain names are represented at the center of the circle (like a star topology), the objects representing the domain names will be many in number and will overlap. Similarly, if each circle is at a distance from the other, there will be many edges indicating the relationships between clients and domain names that will

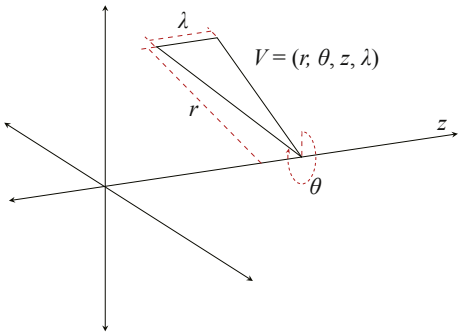


Fig. 1 – Four visualization components of cylindrical coordinates.

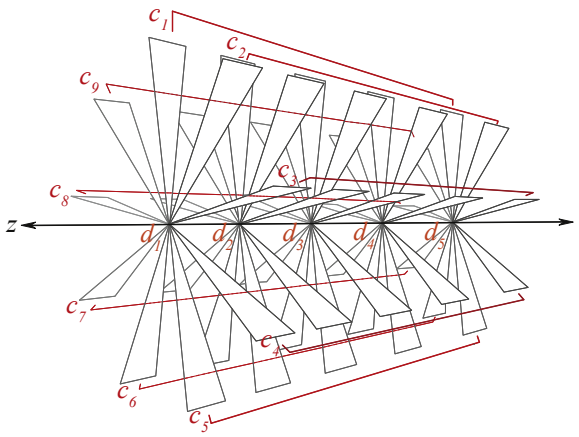


Fig. 2 – Visualizing DNS queries on cylindrical coordinates.

intersect with each other. Therefore, we use *cylindrical coordinates*. Each domain name is represented on a linear axis, and the linear axes are placed orthogonal to each other on the circle (in a cylindrical shape). Generally, a point P in a cylindrical coordinates is represented by r, θ and z (or h), respectively denoting the radius, azimuth, and height of the cylindrical shape, in the three-dimensional space.

To indicate the query intensity of a client, CCSvis represents each query with triangular shape by using additional coordinate λ. We use triangles because other polygonal shapes require more vertices, in turn requiring more computational power. To visually identify each client IP address, each triangular shape is represented by a unique azimuth depending on its client IP address, and each point on the z-axis indicates a domain name that is queried by a client (see Fig. 1).

We denote each octet of a single client IP address c_i by IP_1, IP_2, IP_3 and IP_4. Then, the IP address can be expressed as follows:

$$\underbrace{\mathrm{IP}_1(c_i)\cdot 2^{24}}_{1st\ octet} + \underbrace{\mathrm{IP}_2(c_i)\cdot 2^{16}}_{2nd\ octet} + \underbrace{\mathrm{IP}_3(c_i)\cdot 2^{8}}_{3rd\ octet} + \underbrace{\mathrm{IP}_4(c_i)}_{4^{th}octet} \tag{1}$$

From Equation (1), we calculate the angle of the client IP address as follows:

$$\theta(c_i) := \left(\sum_{k=1}^{4} IP_k(c_i)\frac{360}{2^{8(4-k)}}\right)\frac{\pi}{180} \tag{2}$$

Thus, each client IP address can be mapped to a radial θ. The height of the triangular shape, r, is defined depending on $|\mathscr{C}(d_m)|$ as follows:

$$r(c_i) := \ln[|\mathscr{C}(d_m)|] + \tau_{\mathscr{N}} \tag{3}$$

where the threshold $\tau_{\mathscr{N}}$ can be determined by the network scale or display resolution. The location of the triangular shape on the z-axis, z, is arranged in descending order of $|\mathscr{C}(d_m)|$, where the greater the $|\mathscr{C}(d_m)|$ value, the smaller is the z value. Each $|\mathscr{C}(d_m)|$ is stored in a red–black tree. Let the function $rank(\cdot)$ return the rank of $|\mathscr{C}(d_m)|$ in the tree. Then, the z value is defined as follows:

$$z(c_i) := rank(|\mathscr{C}(d_m)|) \tag{4}$$

Let V be the set of vertices that contains the components of each triangular shape; then, the range of coordinates of the vertex set $V = \{r, \theta, z, \lambda\}$ is defined:

$$V = \begin{cases} 0 < r \le |\mathscr{C}(d)| \\ 0 < \theta \le 2\pi \\ 0 < z < |D| \\ 0 < \lambda < \tau_{\mathscr{J}} \end{cases} \tag{5}$$

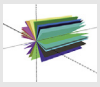
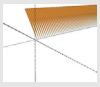
As a result, as shown in Fig. 2, DNS queries can be visualized on cylindrical coordinates.

4.3. *Visual signatures*

CCSvis visualizes a variety of threat patterns on cylindrical coordinates. We define them as five *visual signatures* (see Table 3) for interpretation of threats and adopt a human-cognitive approach which is better at catching user's attention, makes it easier to perceive the salient parts of the graph, making it easier to detect threats.

The disc-shaped pattern in Table 3 indicates that infected clients (or bots) are trying to find the *C&C* server. The triangular shapes represent each client that is stretched out from the z-axis to form a disc-shaped pattern. Of course, the disc-shaped pattern may appear in a normal case as well, but unlike botnet behavior, the cardinality of a normal set is very irregular, and because it tends to have a low intensity value, the botnet pattern can be differentiated distinctly from normal behavior by the size, color, and thickness of the disc-shaped pattern (*Type-I*). If a botnet has multiple domain names for C&C, the botnet behavior appears in the form of a striped cylindrical pattern (*Type-II*). If a bot sends many DNS queries, the triangular shape's width increases (*Type-III*). If a graphical pattern appears in the form of a disc-shaped pattern that consists of large triangular shapes, it indicates a *DRDoS* attack or abnormal behavior. CCSvis can also indicate *DNS Cache Poisoning* or different types of abnormal behavior. Suppose that an attacker generates many DNS queries to guess the DNS transaction ID or sends DNS queries to many domains (*Type-IV*). Then, these behaviors appear in the form of panel-shaped patterns along the z-axis, because each triangular shape has the same tilt on the z-axis. Depending on $F(c)$, it appears as an orange-colored panel-shaped pattern, with the rest being represented by a translucent white color. In the case of a domain name listed on the blacklist, the shape is not regular but is irregular and red in color (*Type-V*).

Table 3 – Visual signatures.

	Visual signature	Correlation ($c : d : \lambda$)	Type name
Type-I		M:1:1	Single-domain C&C botnet
Type-II		M:M:1	Multiple-domain C&C botnet
Type-III		M:1:M	DRDoS or abnormal behavior
Type-IV		1:M:1	DNS cache-poisoning or abnormal behavior In this case, the colors of the triangles forming panelshape are set to orange.
Type-V		Irregular	Blacklisted domain (known C&C domain)

4.4. *CCSvis framework*

The framework of CCSvis consists of five modules: (1) traffic collector, (2) parameter extractor, (3) DNS response loader, (4) domain filter, and (5) visualizer as Fig. 3.

The traffic collector collects the DNS response packets from each sensor and arranges them by timestamp. Then, the parameter extractor parses DNS response packets and extracts visualization parameters, such as client IP address, domain name, query type, timestamp, and flags.

The DNS response loader is one of the main parts, its purpose is to group all the client IP addresses by domain name. The DNS response loader has two-level hash table structure. The first-level is the domain table $H_D\langle d, H_C\rangle$ which has a domain name d as a key and an IP table H_C as a value. The second-level is IP table $H_C\langle c, c_\psi\rangle$ which has a client IP address c as a key and a struct c_ψ containing query type array $\vec{q}$, timestamp variation array $\Delta\vec{t}$, and flag array $\vec{f}$ as a value. The details of the DNS response loader module elucidated in Algorithm 1.

Algorithm 1 DNS response loader

MODULE-DNS-RESPONSE-LOADER(R)

Input: $R=\{(c_1,d_1,q_1,t_1,f_1), \ldots,(c_n,d_n,q_n,t_n,f_n)\}$, where c_i is client IP address, d_i is domain name, q_i is query type, t_i is timestamp, and f_i is flag.

Output: $H_D\langle$key, value$\rangle:=\langle d, H_C\rangle$, H_D is a hash table that has domain name d as a key and a hash table H_C as a value. H_C, $H_C\langle$key, value$\rangle:=\langle c, c_\psi\rangle$, is a hash table that has a client IP address c as a key and a struct $c_\psi := \{\vec{q}, \Delta\vec{t}, \vec{f}\}$ as a value — where, $\vec{q}$ is query type array, $\Delta\vec{t}$ is timestamp variation array, and $\vec{f}$ is flag array.

```
while R ≠ ∅ do
    Δt_i ← t_i − t_0
    if H_D.HASH-SEARCH(d_i) then
        if H_C.HASH-SEARCH(c_i) then
            H_C[c_i].c_ψ := (q_i, Δt_i, f_i)
        else
            Creates a new struct c_ψ'
            H_C.HASH-INSERT(c_i, c_ψ' := (q_i, Δt_i, f_i))
        end if
    else
        Creates a new struct c_ψ'
        H_C.HASH-INSERT(c_i, c_ψ' := (q_i, Δt_i, f_i))
        H_D.HASH-INSERT(d_i, H_C)
    end if
end while
```

The DNS response loader takes a set of visualization parameters as an input and outputs the domain table H_D. It checks if the incoming d_i already exists in the domain table H_D, then, checks whether c_i exists in the corresponding IP table H_{C_i}. If yes, then, inserts $(q_i, \Delta t_{t_i}, f_i)$ to the existing array $c_\psi.\vec{q}$, $c_\psi.\Delta\vec{t}$ and $c_\psi.\vec{f}$, respectively, else it creates a new struct c'_ψ, inserts

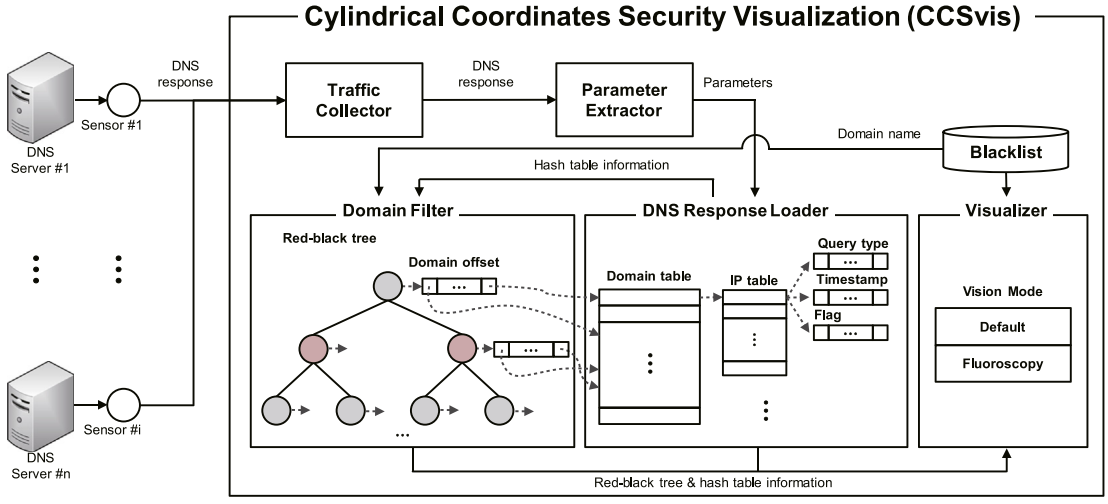


Fig. 3 – CCSvis framework.

$(q_i, \Delta t_{t_i}, f_i)$ to the $c'_\psi.\vec{q}$, $c'_\psi.\Delta\vec{t}'$ and $c'_\psi.\vec{f}$, respectively. Then, it inserts into the IP table H_C with c_i as the key and c'_ψ as the value. If d_i does not exist in H_D, creates a new struct c'_ψ, inserts $(q_i, \Delta t_{t_i}, f_i)$ to the $c'_\psi.\vec{q}$, $c'_\psi.\Delta\vec{t}'$ and $c'_\psi.\vec{f}$, respectively. Then, it inserts into the IP table H_C with c_i as the key and c'_ψ as the value, inserts into the domain table H_D with d_i as the key and H_C.

The domain filter module filters domain names and groups them by cardinality, the size of client IP address set that queries a domain name. It takes the domain table H_D from DNS response loader as an input and builds a red–black tree T as an output. As shown in Algorithm 2, T has a cardinality $|\mathscr{C}(d)|$ as a key and an array of the offsets of domain names in H_D as a value.

Algorithm 2 Domain filter

MODULE-DOMAIN-FILTER(H_D)
Input: $H_D\langle$key, value$\rangle := \langle d, H_C\rangle$, H_D is the hash table that has a domain name d a key and a hash table H_C as a value. B is a blacklist of domains.
Output: $T\langle$key,value$\rangle := \langle|\mathscr{C}(d)|, \vec{o}\rangle$, T is a red-black tree that has a cardinality $|\mathscr{C}(d)|$ as a key and an array of offsets $\vec{o}$ as a value.
// $|\forall\varrho_c|$ is the number of entire queries from client c.
// δ_D is an iterator for H_D.

```
1:  δ_D ← H_D.BEGIN()
2:  while δ_D ≠ H_D.END() do
3:      d := δ_D → key
4:      if (|∀ϱ_c| ≥ τ_ϱ) ∨ (d ∈ B) then
5:          |𝒞(d)| := δ_D → value → H_C.GET-SIZE()
6:          if T.TREE-SEARCH(|𝒞_d|) then
7:              o⃗.ADD(H_D[d].GET-OFFSET())
8:          else
9:              Creates a new array o⃗'
10:             o⃗'.ADD(H_D[d].GET-OFFSET())
11:             T.TREE-INSERT(|𝒞(d)|, o⃗')
12:         end if
13:     end if
14: end while
```

While the domain filter module traverses the domain table H_D, it checks if the total number of queries by a client IP address $\forall\varrho(c)$ is greater than the threshold $\tau_{\mathscr{C}}$, or if d is listed on the blacklist B DNS-BH. $|\forall\varrho(c)|$ herein can be easily calculated from H_C. In particular, this conditional statement can filter a considerable amount of meaningless data because the DNS query distribution follows a Zipf's law Jung and Sit (2004). If the condition is satisfied, it checks if $|\mathscr{C}(d)|$ exists in T or not, if it is not, creates a new array $\vec{o}'$. Then, it inserts the offset of $H_D[d]$ into the corresponding $\vec{o}'$, and it inserts into T with $|\mathscr{C}(d)|$ as the key, and $\vec{o}'$ as the value.

The visualizer module takes three inputs, domain table H_D, red–black tree T, and B, then outputs a set of triangle vertices of the cylindrical coordinates. The module starts traversing from T and traces back to H_D and H_C to calculate the vertex set $V = (r, \theta, z, \lambda)$ as described in Section 4.2. The details are described in Algorithm 3. In order to improve the perceptual efficiency, it provides two display modes: *default* and *fluoroscopy*. In *default* mode, visualizer module specifies triangle color (in the web version) by using the 2nd, 3rd and 4th octets of c. On the contrary, in *fluoroscopy* mode, it checks whether d is B, then, it sets the triangle color to red. If $F(c)$ is greater than or equal to $\tau_{\mathscr{F}}$, the color is set to orange. Otherwise, the triangle color is set to white. Then, the assigned color is alpha-blended. Fig. 4 shows the screenshot of CCSvis and the demonstration can be found at YouTube (http://youtu.be/7NlELrutT0A).

5. Evaluations

In this section, we present experimental results demonstrating the benefits of CCSvis. First, we explain experimental environment with the details of the datasets.

5.1. *Experimental environment*

We conducted evaluations for CCSvis using the obtained trace from an actual ISP network. The dataset is a captured DNS

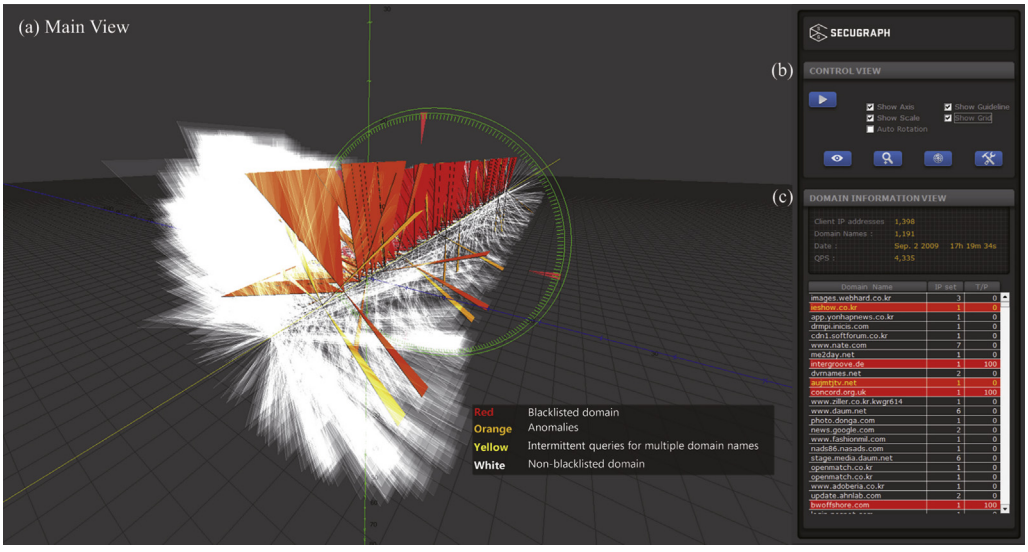


Fig. 4 – Screenshot of CCSvis.

Algorithm 3 Visualizer algorithm

MODULE-VISUALIZER (T, H_D, B)
Input: $T\langle$key, value$\rangle$:=$\langle|\mathscr{C}(d)|, \vec{o}\rangle$, T is a red–black tree that has cardinality $|\mathscr{C}(d)|$ as a key and an array of offsets $\vec{o}$ as a value. $H_D\langle$key, value$\rangle$:=$\langle d, H_C\rangle$, H_D is the hash table that has a domain name d a key and a hash table H_C as a value. B is a blacklist of domains.
Output: $V = (r, \theta, z, \lambda)$, V is set of triangle vertices on the cylindrical coordinates.
// δ_C: iterator for IP table.
// δ_T: iterator for red–black tree.
// $\tau_{\mathscr{F}}$: threshold for flag error rate.
// $\tau_{\mathscr{I}}$: threshold for query intensity.
// pre-compute $\lambda(c)$
// pre-compute $F(c)$

```
δ_T := T.BEGIN()
while δ_T := T.END() do
    |C(d)| := δ_T→key++
    while i ≤ δ_T→value→|o⃗| do
        d* := δ_T→value→o⃗[i]
        d := H_D.GET-KEY-AT(d*)
        δ_C := H_D[d].H_C.BEGIN()
        while δ_C ≠ H_C.END() do
            c := δ_C→key
            r(c) := ln(|C(d)|) + τ_F
            θ(c) := (Σ_{m=1}^{4} IP_m(c) 360/2^{8(4−m)}) π/180
            z(c) := rank(|C(d)|)
            switch DisplayMode do
                case Default
                    SETRGB(IP_2(c), IP_3(c), IP_4(c))
                case Fluoroscopy
                    if d ∈ B then
                        SETRGB(0xFF, 0x00, 0x00)
                    else if (λ(c)≥τ_I)∨(F(c)≥τ_F) then
                        SETRGB(0xFF, 0x80, 0xFF)
                    else
                        SETRGB(0xFF, 0xFF, 0xFF)
                    end if
                    ENABLE-ALPHA-BLENDED()
            Draw V
        end while
    end while
end while
```

response traffic from two network interface card (NIC) of a DNS server working on an ISP network during July 7–17 and September 2, 2009. Among them, we used the DNS response traffic which was captured at the peak hours of a day between 4:54 p.m. to 7:20 p.m. There are 48,109,101 packets, 114,744 distinct IP addresses, 1,380,760 unique domain names in 12.9 GB of the captured DNS response traffic data.

All visualizations were performed on a desktop PC with a 3.0 GHz Intel ® Core™ 2 Duo E8400 processor (6 MB L2 cache and 1333 MHz FSB), 2 GB DDR2 main memory, NVIDIA ® PCIe video card (GeForce 8600 GT™, 128-bit memory bus and 512 MB memory capacity), and 100 Mbps bandwidth. CCSvis is written in C/C++ using OpenGL.

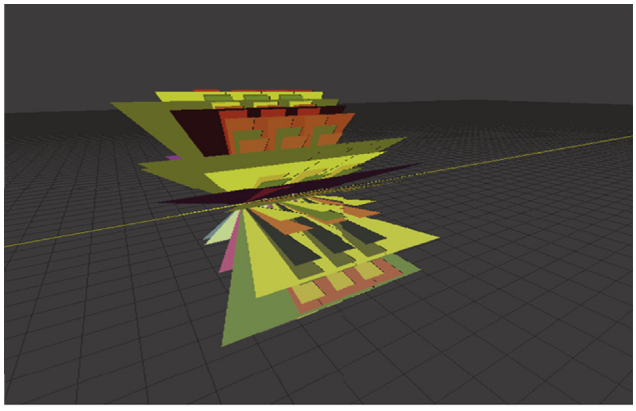
5.2. *Visualizing multiple domain C&C botnets*

We detected two different types of *multiple domain C&C botnets*: *Rustock* and *Conficker*. Both *Rustock* and *Conficker* are some of the most widely propagated botnets. *Rustock* (also known as *Spambot*) first appeared in November 2005, and by the end of 2010, it was considered responsible for 40% of all spam sent worldwide Symantec (2010). *Conficker* (also known as *Downadup* or *Shadow*) is known for installing yet another variant through thousands of domain names that are generated by DGA.

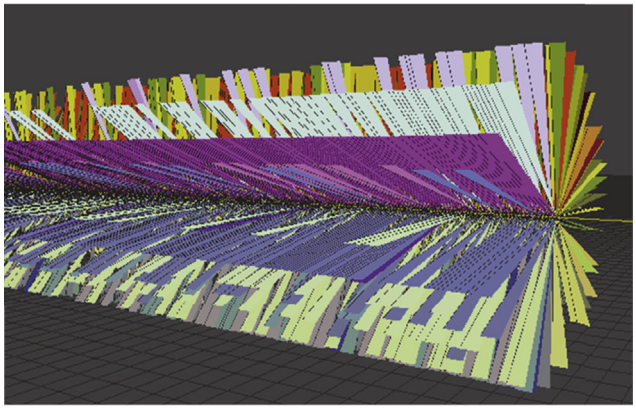
Rustock. Fig. 5(a) shows 77 distinct bots querying 3 C&C domains. In our case study, *Rustock* can be classified as the *multiple domain C&C botnet pattern* of the five signatures (*Type-II*).

Conficker. Fig. 5(b) shows 775 distinct bots querying 644 C&C domains. *Conficker* is also classified as the *multiple domain C&C botnet*.

The cylindrical shape consists of many triangular shapes, and by using the visualized DNS traffic, we can intuitively understand more about these shapes. Each shape is stretched out from one single point on the z-axis (a single yellow line (in the web version)) penetrates the cylindrical shape, as shown in Fig. 5(a) and (b), with the area of the shape representing the query intensity of the client. Multiple triangular shapes on the z-axis form one cylindrical shape of short length that represents a group of multiple clients querying a single domain. If a



(a)



(b)

Fig. 5 – Visualizing Rustock and Conficker using CCSvis: (a) Rustock. (b) Conficker.

group of multiple clients queries many domains simultaneously, a long cylindrical shape will be displayed in the main view. This is indicative of a simultaneous and organized attack; under normal conditions, any form does not appear. In addition, CCSvis is able to visualize these botnet in a matter of minute. However, automated mechanisms are difficult to detect in real time due to high time complexity.

Fig. 6 shows the visualizations of *Rustock* and *Conficker* using four other techniques described in Section 3.2.1. In addition, all images of Figs. 5 and 6 were visualized using the same dataset (as previously discussed in Section 5.1).

5.3. *Comparison of perceptual efficiency*

To compare the perceptual efficiency, we carry out the assessment based on *saliency map* analysis. Salience (or saliency) implies the state or quality by which something stands out relative to surrounding objects, and it is represented by a scalar quantity on a saliency map. Saliency map is widely used to measure the perceptual quality of images or video in various fields.

In our evaluation, we specifically used a model that resembles the human visual system (HVS) called *Graph-Based Visual Saliency* (GBVS) Harel et al. (2007). The GBVS is a computational model of standardized approaches in the field of neuroscience and shows remarkable accuracy Itti and Koch (2001); Itti et al. (1998); Parkhurst et al. (2002). Hence, we consider this model to be suitable for measuring the perceptual efficiency and derive quantitative results for the perceptual efficiency by using it.

To assess how effective CCSvis is compared to other visualization techniques, we chose the following widely used different conventional mechanisms: scatter plot, node-link graph, parallel coordinates, and tree map. We then performed a comparative evaluation in terms of perceptual aspects, as shown in Fig. 7.

In our evaluation, Figs. 5 and 6 show the original images, whereas Fig. 7 shows the corresponding saliency maps. In the saliency maps, a salient region is overlaid on the original. The color (in the web version) of the salient region varies from red to blue depending on the saliency (red color indicates the most salient region).

Despite some noise due to normal, our evaluation has shown a high resilience to noise and greater perceptual efficiency than the other techniques. As shown in Fig. 7, in the saliency map of the other four techniques, the distribution of the salient region was too small compared to the target region and it was in an inapposite area (nontarget area) due to visual distractors. If the distribution of the salient region is too small, users will not be able to clearly recognize the overall outline of a pattern. Furthermore, users will not obtain any useful information if the distribution of the region appears outside the target area. On the other hand, when using CCSvis, the distribution of the salient region was quite large compared to the target region. If the distribution of the salient region is large, it will attract the attention of users (see Fig. 7(a) and (b)). The comparisons with each visualization mechanism are described below.

Node-link graph. In a node-link graph, several algorithms can be used for determining the graph layout, among which, we carefully selected the most well represented result. We used the *Fruchterman–Reingold* graph layout algorithm Fruchterman and Reingold (1991), a type of force-directed layout algorithm. This algorithm arranges the graph by assuming that each node exerts repulsive force and each edge exerts attractive force. It has the most well-represented results. As shown in Fig. 6(a) and (b), client IP addresses are represented by white-colored circles; domain names, by orange-colored (in the web version) circles; and the relationships between client IP addresses and domain names, by semitransparent white-colored lines. In Fig. 7(c) and (d), the most salient region only appeared in the group of orange-colored circles near the center of the screen. A user cannot easily recognize the botnet because the distribution of the salient region is too small to recognize the overall outline of the pattern.

Parallel coordinates. We evaluated various orders of each axis in parallel coordinates, and we selected the most well-expressed result, as shown in Fig. 6(c) and (d). In this figure, client IP addresses are indicated on the first vertical axis; the querying intensity of the client, on the second vertical axis; domain names, on the third vertical axis in alphabetical order; and the cardinality of the client IP address set, on the fourth

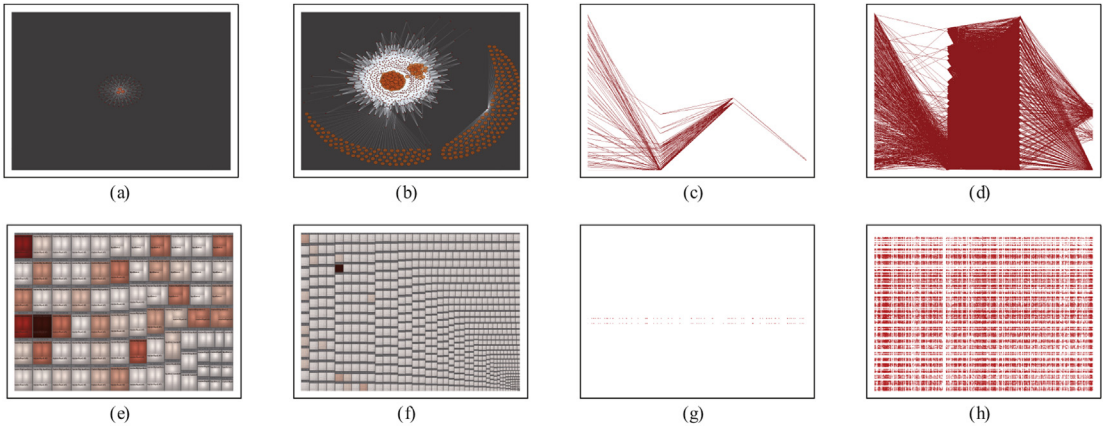


Fig. 6 – Visualizing Rustock and Conficker using other techniques: (a) Nodelink graph showing Rustock. (b) Nodelink graph showing Conficker (c). Parallel coordinates showing Rustock. (d) Parallel coordinates showing Conficker. (e) Tree map showing Rustock. (f) Tree map showing Conficker. (g) Scatter plot showing Rustock. (h) Scatter plot showing Conficker.

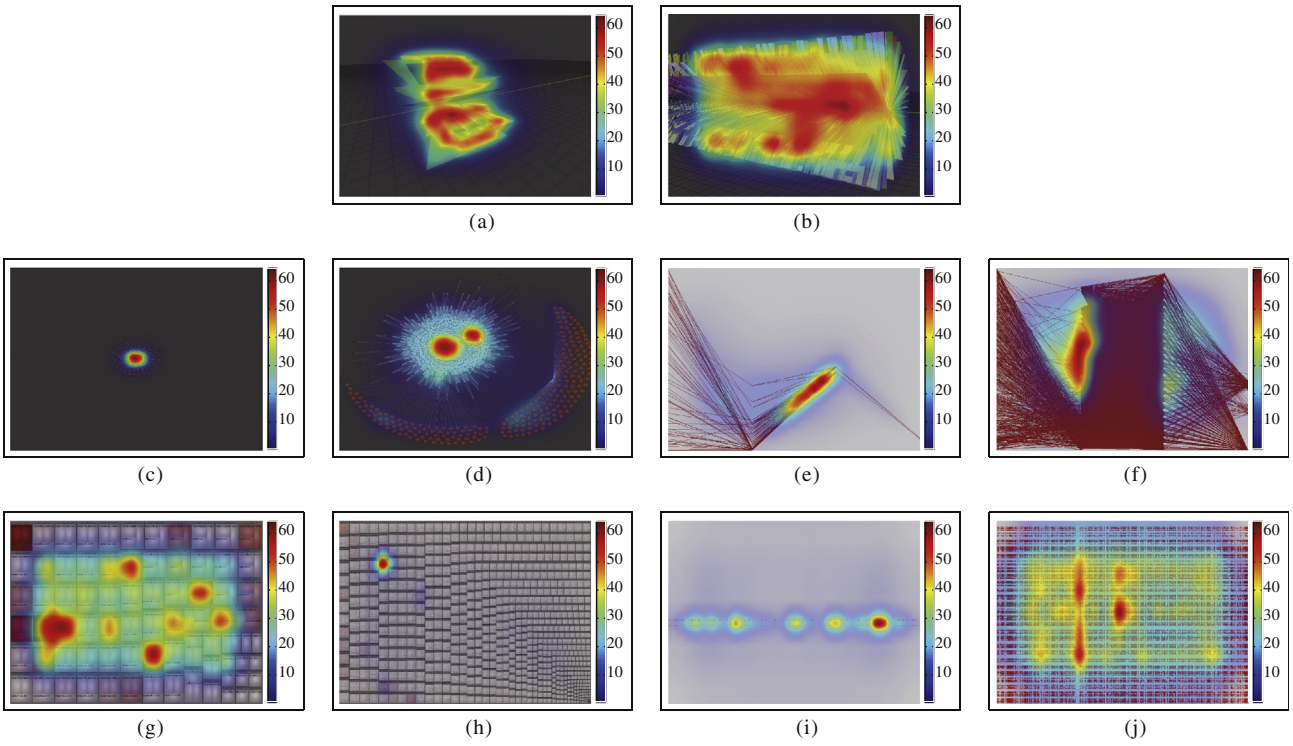


Fig. 7 – Saliency maps of Rustock and Conficker visualizations. (a) Saliency map of CCSvis showing Rustock. (b) Saliency map of CCSvis showing Conficker. (c) Saliency map of nodelink graph showing Rustock. (d) Saliency map of node-link graph showing Conficker. (e) Saliency map of parallel coordinates showing Rustock. (f) Saliency map of parallel coordinates showing Conficker. (g) Saliency map of tree map showing Rustock. (h) Saliency map of tree map showing Conficker. (i) Saliency map of scatter plot showing Rustock. (j) Saliency map of scatter plot showing Conficker.

vertical axis. As shown in Fig. 7(e), many *edge-crossings* occur between the first and the second axis. In addition, as shown in Fig. 7(f), occlusions occur between the second and the third axis. In Fig. 7(e), the most salient region is represented by the area of dense lines near the center of the screen. In particular, in Fig. 7(f), the most salient region appears close to the second axis. Overall, these results show the worst effectiveness for perception.

Tree map. We evaluated various layouts of each parameter in the tree map, and we selected the most well-expressed result as shown in Fig. 6(e) and (f). In this figure, each client IP address is labeled by a gray-colored box, where the client IP address is indicated at the top of the box. In addition, the size of the box depends on the number of distinct domain names that are queried by the client. Domain names are represented as rectangular shapes with a radial gradation, and their size depends on the cardinality of all the client IP addresses that query the domain name. The color of each rectangular shape in the gray-colored box is determined by the querying intensity of the client. When using a tree map to identify botnets, a user needs to read all the labeled text; however, actually doing so requires considerable time even for a small-scale botnet. In Fig. 7(g), the salient region appears near the darker-colored (in the web version) shapes, because these stand out compared to other (white-colored) shapes. However, darker-colored shapes simply indicate the intensity, and they do not help in identifying the botnet. Notably, in Fig. 7(h), dark-colored shapes serve as visual distractors that confuse users. In addition, users cannot find any information because the objects are too small.

Scatter plot. To express multi-dimensional data, a scatter plot could be expressed as a scatter matrix; however, here, we deal with the correlation between the client IP address and the domain names. Fig. 6(g) and (h) show *Rustock* and *Conficker* as they appear in a scatter plot. In both figures, the horizontal direction (x-axis) represents the client IP address and the vertical direction (y-axis), the domain names arranged in alphabetical order. Further, we did not display each axis in the figures because they can serve as visual distractors. As shown in Fig. 6(g), the pattern is too dim for a small-scale botnet, and its salient region is small (Fig. 7(i)). As shown in Fig. 7(j), the most salient region appears near the dense dots near the center of the screen and in the gap between the points, but its distribution is too small to recognize the overall outline of the pattern.

6. Conclusions

In this paper, we make an in-depth investigation on the issue of multiple domain C&C botnet detection. We present a new visualization mechanism for visualizing botnet behaviors. This mechanism is developed with the objective of enabling users to recognize security threats promptly and mitigate the

damage by visualizing information with a focus on greater perceptual efficiency. To successfully achieve this goal, we have introduced two innovative features. First, we discussed the problems faced in existing visualization mechanisms and the design criteria for solving these problems. CCSvis does not use any botnet detection algorithm, but it enables human to recognize a botnet by visualizing the complex correlation structure of botnets. CCSvis is the first security visualization system that considers both system-oriented and the human-oriented perspective. Second, CCSvis employs a new mechanism that visualizes botnet behaviors and defines a variety of threat patterns on cylindrical coordinates. We categorized them into five visual signatures for interpretation of threats and adopt a human-cognitive approach which is better at catching user's attention. Experimental results show that CCSvis has noticeable advantages over the related works.

CCSvis can be operated on a standard personal computer without requiring the use of high-end workstations or general-purpose GPU-based parallel computing devices. In the future, we intend to carry out empirical studies in various workplaces. The results of such studies will contribute toward designing an advanced visualization technique that offers better security. Also, the approach proposed in this study can be utilized to derive new and valuable insights in security aspects from the complex correlations of *Big Data*.

Supplementary data related to this article can be found online at http://dx.doi.org/10.1016/j.cose.2014.07.007.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (2012M3A2A1051118, 2012-0005552, 2011-0015187, 2011-0003930, 2010-0016970, 2009-0069191).

REFERENCES

Abdullah K, Lee C, Conti G, Copeland J, Stasko J. Ids rainstorm: visualizing ids alarms. In: Proceedings of the IEEE workshop on Visualization for Computer Security, VizSec'05. IEEE Computer Society; 2005. p. 1.

Antonakakis M, Perdisci R, Dagon D, Lee W, Feamster N. Building a dynamic reputation system for DNS. In: USENIX Security Symposium; 2010. p. 273–90.

Ball R, Fink G, North C. Home-centric visualization of network traffic for security administration. In: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security (VizSEC/DMSEC). ACM; 2004. p. 55–64.

Bilge L, Kirda E, Kruegel C, Balduzzi M. Exposure: finding malicious domains using passive DNS analysis. In: NDSS; 2011.

Born R, Bradley D. Structure and function of visual area MT. Annu Rev Neurosci 2005;28:157–89.

Brustoloni J, Farnan N, Villamarín-Salomón R, Kyle D. Efficient detection of bots in subscribers' computers. In: Communications, 2009. ICC'09. IEEE International Conference on. IEEE; 2009. p. 1–6.

Chiang K, Lloyd L. A case study of the rustock rootkit and spam bot. In: Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets. USENIX Association; 2007. p. 10.

Choi H, Lee H, Kim H. Fast detection and visualization of network attacks on parallel coordinate. Comput Secur 2009;28(5):276–88.

Choi H, Lee H, Lee H, Kim H. Botnet detection by monitoring group activities in DNS traffic. In: Proceedings of the 7th IEEE International Conference on Computer and Information Technology, 2007. CIT'07. IEEE; 2007. p. 715–20.

Colombe J, Stephens G. Statistical profiling and visualization for detection of malicious insider attacks on computer networks. In: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security. ACM; 2004. p. 138–42.

Conti G, Abdullah K. Passive visual fingerprinting of network attack tools. In: Proceedings of CCS Workshop on Visualization and Data Mining for Compute Security. ACM; 2004. p. 45–54.

Dagon D. Botnet detection and response. In: OARC workshop, vol. 2005; 2005.

DNS-BH. *Malware prevention through domain blocking*. http://www.malwaredomains.com.

Erbacher R, Garber M. Fusion and summarization of behavior for intrusion detection visualization. In: Proceedings of the IASTED International Conference On Visualization, Imaging, and Image Processing; 2004.

Fink G, Ball R, Jawalkar N, North C, Correa R. Network eye: end-to-end computer security visualization. In: Submitted for consideration at ACM CCS Workshop on Visualization and Data Mining for Computer Security (VizSec/DMSec); 2004.

Fischer F, Mansmann F, Keim D, Pietzko S, Waldvogel M. Large-scale network monitoring for visual analysis of attacks. In: Proceedings of the 5th International Workshop on Visualization for Computer Security. VizSec'08. Springer-Verlag; 2008. p. 111–8.

Fruchterman T, Reingold E. Graph drawing by force-directed placement. Softw Pract Exp 1991;21(11):1129–64.

Gu G, Perdisci R, Zhang J, Lee W. Botminer: clustering analysis of network traffic for protocol- and structure-independent botnet detection. In: Proceedings of the 17th Conference on Security Symposium. USENIX Association; 2008. p. 139–54.

Gu G, Porras P, Yegneswaran V, Fong M, Lee W. Bothunter: detecting malware infection through IDS-driven dialog correlation. In: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium. USENIX Association; 2007. p. 12.

Harel J, Koch C, Perona P. Graph-based visual saliency. In: Proceeding of Neural Information Processing Systems. NIPS'09; 2007. p. 545–52.

Iliofotou M. Exploring graph-based network traffic monitoring. In: INFOCOM Workshops 2009, IEEE. IEEE; 2009. p. 1–2.

Ishibashi K, Toyono T, Hasegawa H, Yoshino H. Extending black domain name list by using co-occurrence relation between DNS queries. IEICE Trans Commun 2012;95(3):794–802.

Itti L, Koch C. Computational modelling of visual attention. Nat Rev Neurosci 2001;2(3):194–203.

Itti L, Koch C, Niebur E. A model of saliency-based visual attention for rapid scene analysis. IEEE Trans Pattern Anal Mach Intell 1998;20(11):1254–9.

Jung J, Sit E. An empirical study of spam traffic and the use of DNS black lists. In: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement. ACM; 2004. p. 370–5.

Kovacs E. Inspecting DNS flow traffic for purposes of botnet detection. Technical Report. FBI; 2011.

Kovacs E. Microsoft and financial companies disrupt over 1,400 Citadel botnets. Technical Report. FBI; 2013., http://news.softpedia.com/news/FBI-Microsoft-and-Financial-Companies-Disrupt-Over-1-400-Citadel-Botnets-358830.shtml.

Krasser S, Conti G, Grizzard J, Gribschaw J, Owen H. Real-time and forensic network data analysis using animated and coordinated visualization. In: Proceedings of the 6th IEEE Information Assurance Workshop. IAW'05. IEEE; 2005. p. 42–9.
Li S, Luo Y. Discernibility analysis and accuracy improvement of machine learning algorithms for network intrusion detection. In: Proceedings of the 2009 IEEE International Conference on Communications. IEEE Press; 2009. p. 5430–4.
Manasrah AM, Hasan A, Abouabdalla OA, Ramadass S. Detecting botnet activities based on abnormal DNS traffic. arXiv preprint arXiv:09110487; 2009.
Mockapetris PV. *RFC 1035: domain names-implementation and specification*. http://www.ietf.org/rfc/rfc1035.txt.
Moran J, Desimone R. Selective attention gates visual processing in the extrastriate cortex. Science 1985;229(4715):782–4.
Muelder C, Ma K, Bartoletti T. Interactive visualization for network and port scan detection. In: Recent advances in intrusion detection. Springer; 2006. p. 265–83.
Parkhurst D, Law K, Niebur E. Modeling the role of salience in the allocation of overt visual attention. Vis Res 2002;42(1):107–23.
Porras P, Saïdi H, Yegneswaran V. A foray into conficker's logic and rendezvous points. In: Proceedings of the 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats. USENIX Association, 7; 2009.
Ren P, Kristoff J, Gooch B. Visualizing DNS traffic. In: Proceedings of the 3rd International Workshop on Visualization for Computer Security. ACM; 2006. p. 23–30.
Resko B, Roka A, Baranyi P. Visual cortex inspired intelligent contour detection. J Adv Comput Intell Intell Inform 2006;10(5).
Samak T, Ghanem S, Ismail M. On the efficiency of using space-filling curves in network traffic representation. In: INFOCOM Workshops 2008, IEEE. IEEE; 2008. p. 1–6.
Stone-Gross B, Cova M, Cavallaro L, Gilbert B, Szydlowski M, Kemmerer R, et al. Your botnet is my botnet: analysis of a botnet takeover. In: Proceedings of the 16th ACM conference on Computer and Communications Security. CCS'09. ACM; 2009. p. 635–47.
Suo X, Zhu Y, Owen S. A task centered framework for computer security data visualization. In: Visualization for Computer Security; 2008. p. 87–94.
Symantec. MessageLabs Intelligience:2010 Annual Security Report. Technical Report. Symantec; 2010.
Uehara R, Toda S, Nagoya T. Graph isomorphism completeness for chordal bipartite graphs and strongly chordal graphs. Discret Appl Math 2005;145(3):479–82.
Villamarín-Salomón R, Brustoloni J. Identifying botnets using anomaly detection techniques applied to DNS traffic. In: Proceedings of the 5th IEEE Consumer Communications and Networking Conference. CCNC'08. IEEE; 2008. p. 476–81.
Villamarín-Salomón R, Brustoloni JC. Bayesian bot detection based on DNS traffic similarity. In: Proceedings of the 2009 ACM symposium on Applied Computing. ACM; 2009. p. 2035–41.
Von Ahn L, Blum M, Hopper NJ, Langford J. Captcha: using hard AI problems for security. In: Advances in Cryptology. EUROCRYPT 2003. Springer; 2003. p. 294–311.
Williams L, Lippmann R, Ingols K. Garnet: a graphical attack graph and reachability network evaluation tool. In: Visualization for Computer Security; 2008. p. 44–59.
Wyke J. Over 9 million PCS infected – zeroaccess botnet uncovered; 2012. http://nakedsecurity.sophos.com/2012/09/19/zeroaccess-botnet-uncovered/.
Yadav S, Reddy A, Reddy A, Ranjan S. Detecting algorithmically generated malicious domain names. In: Proceedings of the 10th Annual Conference on Internet Measurement. ACM; 2010. p. 48–61.

Ilju Seo is a founder and CEO at Secugraph, Inc., Seoul, Korea. He has an M.S. degree in Department of Computer Science and Engineering from Korea University in 2012, and a B.S. degree from Myongji University in 2010. His research interests include network, Internet security and visualization for security.

Heejo Lee is a professor at the Division of Computer Communication Engineering, Korea University, Seoul, Korea. Before joining Korea University, he was at AhnLab, Inc. as a CTO from 2001 to 2003. From 2000 to 2001, he was a postdoctorate at CERIAS, Purdue University. Dr. Lee received his B.S., M.S., Ph.D. degrees in Computer Science and Engineering from POSTECH, Pohang, Korea. Dr. Lee serves as an editor of the Journal of Communication and Networks. He has been an advisory member of Korea Internet Security Agency and Korea Supreme Prosecutor's Office.

Seung Chul Han is an associate professor at the Department of Computer Engineering at the Myongji University, Seoul, Korea. He has a Ph.D. degree in Computer Science from the University of Florida in 2007, an M.S. degree in 2003 from Purdue University, and a B.S. degree from Sogang University, Seoul, Korea. His primary research interests include networks, security, and OS.