http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 9 (6): 1124-1132, 2010
ISSN 1812-5638
© 2010 Asian Network for Scientific Information

TrojanURLDetector: A Statistical Analysis Based Trojan Detection Mechanism

Liming Wang, Haoying Mu, Lin Xu, Jinglin Chen, Xiyang Liu and Ping Chen
Software Engineering Institute, Xidian University, P.O. Box 168, No. 2 South Taibai Road, Xi'an, China

Abstract: Trojans' threats are on the rise: malwares attempt to install and run automatically through the way so-called drive-by downloads. To enhance security of web applications, in present study, TrojanURLDetector is proposed for the detection and blocking of Trojans URLs. Traditionally, defenders detect malware simply based on signature, which is client-based, can't share the data with other machines of Internet. TrojanURLDetector detects malicious URL based on every URL's suspicious degree. It calculates suspicious degree per time period and ultimately marked the malicious URL. TrojanURLDetector is in server-side and therefore it has sufficient data from plenty of machines accessed to web to determine whether a URL is malicious or not. Both theoretic proof and simulation results manifest that TrojanURLDetector is high efficient in Trojan URL detection meanwhile it is low misdeclaration-rate.

Key words: Trojan URL, website traffic, statistics law, suspicious degree, web security

INTRODUCTION

Malware is designed to infiltrate a computer system unknowingly to the owner. Because of the popularity of the Internet, malware spreads rapidly and causes major disruptions from attacks. Web-based malware can be classified into two categories according to the techniques used for delivery (Provos *et al*., 2008). The first uses social engineering techniques to entice users to download and run malware (Provos *et al*., 2008). The second exploits vulnerabilities of web browser to automatically download and run malware unconsciously while the user is visiting a website (Provos *et al*., 2008). Malware in the second category is also referred to as drive-by downloads and is more concealed and dangerous. Trojans which created a growing threat to web security are a form of drive-by downloads. Unlike attack behaviors of other malwares, Trojans operate by stealing usernames and passwords or scanning victims' drives for sensitive data. What's more, a Trojan can also carry rootkits or backdoors.

Nowadays, browser vulnerabilities are exploited to propagate drive-by downloads, which has been a serious emerging threat to computer systems. Strider Honey Monkey from Wang *et al.* (2006) is a pipeline of active client-side VM-based honeypots. It performs large scale, systematic and automated web patrol to detect browser based on vulnerability exploit. Provos *et al.* (2008) make use of Strider Honey Monkey and VM-based technique to process millions of web pages: Every suspicious URL is assigned to a VM for verification. They also construct the Malware distribution network to understand the properties of the web malware serving infrastructure to the internet.

In present study, we found that the threat brought by Trojans is related to the traffic of the malicious exploited websites. Therefore, we innovatively adopt the statistical law of the website traffic to detect Trojan URL. Based on which, we proposed a Trojan detection mechanism, TrojanURLDetector, which is a set of well organized servers where, at a fast speed. Users visit URLs, at the same time, it records all the URLs accessed, scans the health states itself and sends the data to the TrojanURLDetector servers. The servers marked the Trojan URLs according to the users' data and the algorithms in this study. It is demonstrated by simulate experiments that the TrojanURLDetector can detect Trojan URLs at a faster speed, furthermore with a low misdeclaration-rate.

TROJANS IN DEPTH

Trojans processes: A Trojan is a malware that appears to perform a desirable function, but in fact, facilitates unauthorized access to the user's computer system. Trojans are not self-replicated which distinguishes them from viruses and worms. Additionally, they require interaction with an attacker to fulfill their purpose. Trojans allow an attacker to gain remote access to a target computer system. In general, the process for a Trojan to infect computers includes the following steps.

Corresponding Author: Liming Wang, Software Engineering Institute, Xidian University, P.O. Box 168, No. 2 South Taibai Road, Xi'an, China Tel: +86-29-88204612

- **Configuration:** Once a Trojan is designed, numerous techniques are employed to make the Trojan undetectable
- **Propagation:** Trojans can spread widely and quickly by E-MAIL or software downloads. Today, an easy way to populate the Trojan is to insert its URL into a hot web page
- **Execution:** As soon as users visit the Trojan URL, the Trojan will be installed into the users' computers automatically

Information leakage: The well-designed Trojans possess an information feedback mechanism.

Connection and remote control: Trojan contacts its control side automatically and gives it remote access authority. Thus, attackers in the control side can freely use the victim machine.

Characteristics of Trojans: For further discussion, we first give some definitions and reasonable assumptions. The most important term is the Trojan URL. A Trojan URL is a URL that triggers a Trojan to be downloaded immediately and automatically installed into the victim machine without the knowledge or consent of the user and it is a URL that does not redirect the web browser to any other URLs. All other URLs are deemed as normal ones, including URLs exploited by a Trojan to redirect the web browser to Trojan URLs or normal ones. In Fig. 1, Eq. 1 is a Trojan URL while Eq. 2-5 are normal ones. Once the immediate linkage to a Trojan URL is cut off, users can be protected from the Trojan threat. Another term is the visiting tree. The URLs visited by a user can be constructed into several trees which record the track of the web browser. Such a tree is named visiting tree, where the web browser starts from the root (i.e., normal URLs), goes through all non-leaf nodes and terminates at the leaf nodes (i.e., the Trojan URL). An example of a visiting tree is also shown in Fig. 1.

In order to facilitate the experiments, we present the following three reasonable supports which describe the Trojan URLs and Trojan behaviors and then the explanation followed:

- **Support 1:** Trojan URLs only exist in the leaf of the Visiting Tree
- **Support 2:** Once a Trojan is downloaded, new process creation or file system or registry modification occurs
- **Support 3:** Neither a Trojan URL changes into a normal URL, nor a normal URL can degenerate into a Trojan one

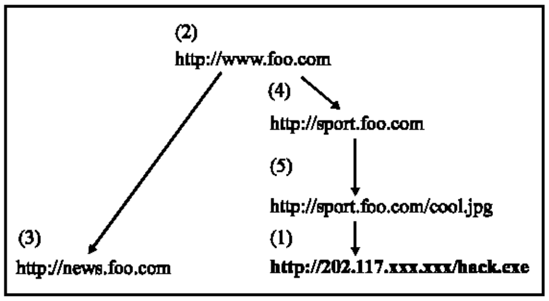

Fig. 1: An example of visiting tree

Attackers mostly insert the Trojan URL into a hot web page by using tags such as iFrame. Thus, it is clear that the web browser must take a number of redirection steps before it eventually contacts the Trojan URL. Trojan URLs never appear in the root or non-leaf nodes of the visiting tree, unless the users visit the known Trojan URL proactively, support 1 is proved. As for support 2, process creation, file system and registry modification are necessary but not sufficient evidence for judging whether a Trojan is downloaded and run. In other words, occurrence of new process creation, file system and registry modification in a system cannot account for the downloading of a Trojan. However, once a Trojan is downloaded, there must be some changes in the system (i.e., a Trojan file added at least), having these changes, computer system states can be monitored. For support 3, it is obvious, in Fig. 1, Eq. 1 is a Trojan URL and others are normal ones, Eq. 5 is a exploited URL, when user access Eq. 5, it will redirect to Eq. 1, then hack.exe will be downloaded and run automatically, even so, Eq. 5 is not a Trojan URL, it is just exploited by malicious code. As for another situation, if hack.exe is replaced by other malwares (http:xxx.xxx.xxx.xxx/another_hack.exe) is deemed as a new Trojan URL.

URL EVALUATION METHODOLOGY

Website traffic distribution: The threat of a Trojan is measured by its prevalence which is determined by the traffic of the exploited website. In this study, we collect the top 1200 sites on the web and rank them from 1 to 1200 by the traffic of each site. Then, we fit the curve of the website traffic against the website rank using the Curve Fitting Tool of Matlab and the result (after normalization) is as follows, where P(x) is the probability of access to website x.

$$P(x) = 35.86e^{-0.2051x} + 1.84e^{-0.007326x} \tag{1}$$

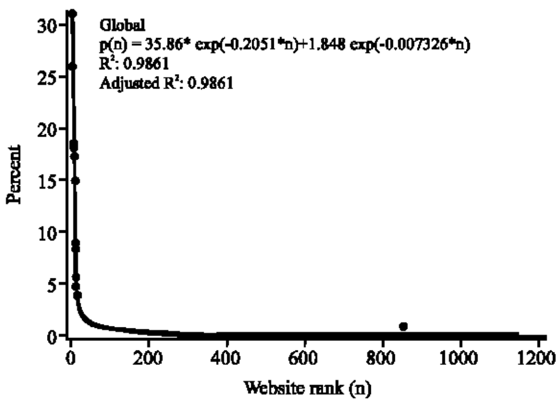

Fig. 2: The fitting curve of traffics of world's top 1200 sites

Equation 1 illustrates the traffic of the website is the exponential function of website rank. The graph of P(x) is shown in Fig. 2.

URL suspicious degree evaluation: According to support 3, if a URL set is safe, it does not contain Trojan URLs. If unsafe, it may contain zero, one or more Trojan URLs. If a URL maybe a Trojan one, we call it a suspicious URL and the variable sus_gree represents the suspicious degree of a URL, i.e., the number of occurrences a URL in suspicious URL sets. The larger a URL's sus_gree, the more likely it will be a Trojan URL. A safe URL's suspicious degree is defined as -1, if the sus_gree of a URL is -1, it must be safe no matter how many times it appears in suspicious URL sets. The sus_gree of each URL is initialized to zero. Figure 3 shows the calculation process of the sus_gree.

Suppose there are eight URLs numbered from 1 to 8 and two users. All the 8 URLs' initial sus_gree value is zero. During time period 1, user1 visited URL 1,3,4,5,7 which all appear in the unsafe URL set, therefore, their sus_gree values are 1 and user2 visited URLs 3,5,7,8 which all appear in the safe URL set, then their sus_gree values were -1. Thus, the calculated sus_gree values from URLs 1 to 8 are 1, 0, -1, 1, -1, 0, -1, -1, respectively. Combined with the calculated sus_gree values in time period 1, we got those values in time period 2 2, -1, -1, -1, -1, 1, -1, -1 shown in Table 1.

P(x) in Eq. 1 can be used approximately as the access probability of the URLs. Therefore, we focus on the variation of the sus_gree of URLs with the passage of time according to Fig. 3.

Suppose A is a set of Trojan URLs and a ∈A, Ā is the complementary set of A, B is a set of safe URLs and b∈B and sus_gree(x) represents the suspicious degree of URL x. In addition, we assume that different users' accesses to URLs are independent. Based on algorithm shown in Fig. 3, we get the following three expressions.

```
Algorithm 1: The calculation process of Sus_gree
  for each URL in Suspicious_SET
      if sus_gree(URL) > = 0
          sus_gree(URL) = sus_gree(URL)+1
      end_if
  end_for
  for each URL in Security_Set
      sus_gree(URL) = -1
  end_for
```

Fig. 3: The calculation process of sus_gree

Table 1: Changes of URLs' suspicious degree

Time period	User 1 URLs	User 1 URL set	User 2 URLs	User 2 URL set	Calculated sus_gree values of 8 URLs
1	1, 3, 4, 5, 7	Unsafe	3, 5, 7, 8	Safe	1, 0, -1, 1, -1, 0, -1, -1
2	2, 4, 8	Safe	1, 3, 6	Unsafe	2, -1, -1, -1, -1, 1 -1, -1

$$\psi(\text{sus\_gree}(a) \leftarrow \text{sus\_gree}(a)+1) = P(a) \quad (2)$$

$$\psi(\text{sus\_gree}(b) \leftarrow \text{sus\_gree}(b)+1) = P(b)\times\sum(\delta \mid \delta\in A) \quad (3a)$$

$$\psi(\text{sus\_gree}(b) \leftarrow -1) = P(b)\times\sum P(\delta \mid \delta\in \overline{A}) \quad (3b)$$

where, Ψ(x) is the occurrence probability of event x. From Eq. 2 and 3a we find that, if a safe URL's traffic is lower than a's, i.e., the Trojan's, the growth of its suspicious degrees is slower than that of a, so the probability of judging a safe URL to be malicious is lower than that of Trojan URLs. According to Eq. 2 and 3b, if a safe URL's traffic is higher than a's, that URL is more easily verified as safe and its sus_gree is set to -1. The explanation follows.

When traffics of safe URLs are higher than a Trojan URL's, we focus on how quickly safe URLs can be verified. Suppose the URL ranked i is a Trojan one and its traffic is the largest among all Trojan ones, so it's easy to know URLs ranked 1 to i-1 are all safe ones. According to Eq. 3b, the higher a safe URL's traffic, the faster it can be verified. Among all safe URLs with traffics sufficiently higher than the URL i, the traffic of URL i-1 is the lowest. If URL i-1 is efficiently verified, all other safe URLs can be verified more efficiently. As long as the traffic of URL i-1 is higher than URL i, Fig. 3 is efficient in the real world. Whether the traffic of URL i-1 is sufficiently higher than URL i is evaluated by the differentiation degree between them. In this study, we define differentiation degree of a discrete function S_n as Eq. 4.

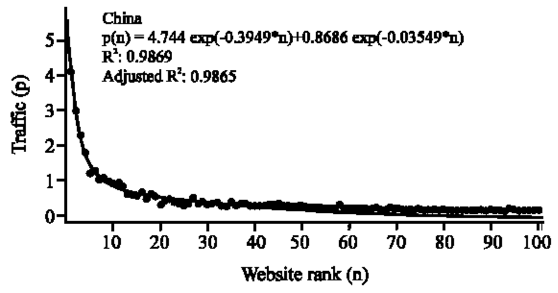

Fig. 4: The fitting curve of traffics of Chinese top 500 sites

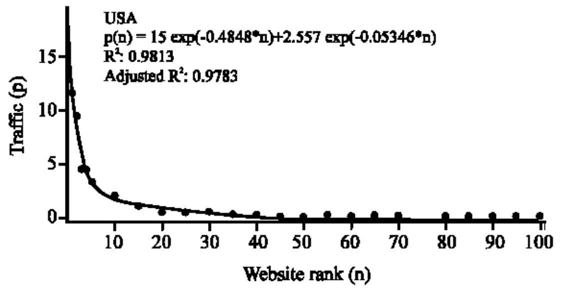

Fig. 5: The fitting curve of traffics of USA top 500 sites

$$\text{diff\_degree}(S_n) = \min\{\frac{S_{i-1} - S_i}{S_i}, i = 2,3,...,n\} \qquad (4)$$

Appendix proves that the differentiation degree of exponential function is the best among all types of functions and it is a constant. The fitting result in Fig. 2 shows that the function of website traffic to website rank is a linear combination of two exponential functions. Its differentiation degree is relatively good and almost stable. So, Fig. 3 is an effective method for Trojan URL detection. Although, websites in Eq. 1 are differentiable to some extent, the differentiation degree is not strictly stable. In practice, frequent users of a certain website always come from the same area. Take the Japanese website yahoo.co.jp for example, 96.5% of its users are in Japan, Internet users from other countries contribute little to its traffic. This affects the stability of the differentiation degree of Eq. 1. If we can cluster all Internet users by their taste in all websites, the efficiency of algorithm shown in Fig. 3 will improve greatly. Here, we cluster Internet users in groups according to their countries and fit website traffic against website rank in each group. Take China for example, we fit the website traffic against its rank according to Chinese people's taste, fitting results are much better, shown in Fig. 4 and the results of USA are similar, shown in Fig. 5.

Appendix proves that differentiation degrees of functions in each group are more stable.

TROJANURLDETECTOR DESIGN

Our ultimate aim is to find Trojan URLs so that users can surf in a safe web environment. So, in this study, TrojanURLDetector is proposed, an innovative mechanism to detect and block Trojan URLs. TrojanURLDetector is a client/server structure. The client side, records browsing data and submits them to the server side periodically. The server side analyzes all submitted data to find Trojan URLs.

Trojan detecting mechanism: The downloading and execution of Trojan URLs must bring some changes to the computer system. Therefore, the mechanism of TrojanURLDetector records the changes and marks the Trojan URLs using those changes. Then, the browser can filter the Trojan URLs to protect the computer system. Present study mainly focuses on the Trojan URLs' detection. TrojanURLDetector system works as follows.

When numerous users simultaneously visit numerous URLs, the URLs information, as well as computer changes including the changes of processes, files and registries, are periodically sent into the server on which run several anti-virus engines. Combined with these data, each URL's suspicious degree is calculated according to Trojan-URL detection algorithm, the key part of TrojanURLDetector system. When the suspicious degree of a URL reaches a certain value, the URL is deemed as suspicious and is submitted to anti-virus engines. If the URL is marked as malicious by at least one anti-virus engine, then it is a Trojan URL.

Trojan-URL detection algorithm: The URLs visited by a user and the user's system states are recorded and stored in a 2-dimension array of sets Set(i, j) and a 2-dimension Boolean array State(i. j). Their definitions are as follows.

Set(i, j) is a set of URLs visited by user i during time period j. According to support 1, Trojan URLs only exist on the leaf of the visiting tree, so URLs on root or intermediate nodes make no contribution to Trojan-URL Detection Algorithm. The parent-child relationship among URLs is determined by Get header and Referer header of HTTP requests. State(i, j) holds user i's system state at the end of time period j and its value is either 0 or 1. State(i, j) = 0 means user i's system is normal during time period j, in other words, the URL set visited during period j by user i is safe; while State(i, j) = 1 means abnormal changes may occur during period j, i.e., the visited URL set is suspicious.

Based on Set(i, j), State(i, j) and support 2, it can be derived that if State(i, j) = 0, all URLs in Set(i, j) are benign and if State(i, j) = 1, there may be Trojan URLs existing in

```
Algorithm 2: Trojan-URL detection algorithm
Input: Set(i, j), State(i, j), n_User, n_Engine
Output: BlackList: a set of determined Trojan URLs
        Security_Set, susgree_list
Initial: Securirt_Set = ϕ, susgree_list = 0
         Time period = 0, BlackList = ϕ
1.  While (!TIMEOUT (TimePeriod))
2.      for i = 1 to n_User
3.          Suspicious_Set = ϕ
4.          if State(i, TimePeriod) = = 1
5.              Suspicious_Set (i, TimePeriod) = Set(i, TimePeriod)
6.          else
7.              Security_Set = Security_Set ∪ Set(i, TimePeriod)
8.          end_if
9.          Suspicious_Set(i, TimePeriod) - = Security_Set
10.           for each URL in Suspicious_Set
11.               susgree_list(URL) = susgree_list(URL)+1
12.         end for
13.         FREE (Set(i, TimePeriod))
14.         FREE (State(i, TimePeriod))
15.         FREE (Suspicious_Set(i, TimePeriod))
16.     end_for
17.     for j = 1to n_Engine
18.       if URL_CHECKING_ENGINE (susgree_list(j).URL) = = 0
19.             Security_Set = Security_Set ∪ {susgree_list (j).URL}
20.         else
21.             Blacklist = BlavckList ∪ {susgree_list (j).URL}
22.         end_if
23.         ERASE (susgree_list, j)
24.     end_for
25. end_while
```

Fig. 6: Torjan_URL detection algorithm

Set(i, j). Suppose that Security Set(i, j) is a set of all benign URLs and suspicious Set(i, j) is a set of all suspicious URLs visited by user i during time period j. It is easier to mark the Trojan URLs in suspicious Sete(i, j) after which is minimized by subtracting the security set from itself.

Suspicious degree represents the occurrence number of a URL in all Suspicious URL Sets. Suspicious degree is stored in a 1-dimension descending array susgree_list(i), where susgree_list(i) is the ith highest Trojan URL and susgree_list(i).degree is its corresponding suspicious degree. The upper boundary of suspicious degree is called the malicious threshold, demonstrating the efficiency of Trojan-URL Detection Algorithm and can be obtained via the experiment. If a URL's suspicious degree become larger than malicious threshold, it should be checked in URL checking engine, Algorithm shown in Fig. 6 is the Trojan-URL detection algorithm.

In algorithm shown in Fig. 6, variable time period in line 1 depends on real time. Set(i, j) and State(i, j) are submitted to TrojanURLDetector for calculating per time period. From line 2 to 16, the suspicious degree of every suspicious URL is updated. After updating, URL checking engines fetch the highest n_Engine suspicious URLs for testing (line 18). If a URL is recognized as benign, it will be added to security set (line 19), or if malicious it will be added to BlackList. User computers request the latest BlackList to update their own. Whatever it is suspicious or not, the URL will be erased from susgree_list (line 23) after checking. Line 17 to 24 can be compiled into parallel computation for efficiency. One point should be noticed is that from line 13 to 15, Set, State and Suspicious_set are free to save the runtime memory. In addition, it demonstrates that the sequence of users to submit the data does not affect the correctness of the algorithm, so user's privacy can be protected.

Subtraction is the key operation in the algorithm. After the subtraction operation, the contents in Suspicious Set are only single files, for example, bmp, msic, exe and so on, these are later downloaded into URL checking engines and rendering of such files is much faster than that of a whole web page.

Design optimization: According to the analysis above, the efficiency of Trojan URL detecting is improved by clustering internet users into groups. We design an ideal routine as follows: IPs of websites visited by each user are sampled. Then cluster analysis is applied to the similarities among users and to partition all users into different clusters. The partition will be adjusted according to the changes in statistical data of users as time goes by. But considering the time complexity of cluster analysis and the data size of internet users, it is very unrealistic for TrojanURLDetector to adopt this routine in the real world. So, we cluster all users by the IP address range of each country. A faster clustering method is expected in the future.

EVALUATION

Based on given data and statistical characteristic of the internet, we deployed a virtual internet environment, where we deployed TrojanURLDetector system and evaluated its efficiency. How soon a Trojan can be detected after its outbreak and how many users would be infected before a Trojan is detected are the two most important aspects to evaluate the efficiency of TrojanURLDetector.

Simulation system: There are three components in this simulation system, internet environment, client and TrojanURLDetector. The configuration options of our simulation system include n_URL (total number of URLs), P(x)(traffic of each URL), n_user (number of internet users), avg_URL (average of URLs visited per user per day), TrojanSet (set of Trojans URL), n_Engine (number of Trojans Checking Engine), shown in Table 2. The following are the working process of each component during one time period.

Table 2: Configuration options of our simulation system

Configuration options	Term
Total number of URLs	n_URL
Traffic of each URL	P(x)
Number of Internet users	n_user
Average of URLs visited per user per day	avg_URL
Set of Trojans URL	TrojanSet
No. of Trojans checking engine	n_Engine

Internet environment: In every time period, the traffic of each URL under the internet environment is generated by the following two steps.

Step 1: For each website i, generate a Poisson random number t_i with the mean P(i)

Step 2: Find the traffic of each website i by Eq. 5.

$$\text{Traffic}(i) = t_i / \sum_{j=1}^{5000} t_j \tag{5}$$

Client: Client generates the URL sets of each user visited and the state of each system.

Step 1: Clear all system states

Step 2: Generate a normal random number n_disturb with the mean n_user×disturb_percent and standard deviation 1; Generate n_disturb discrete uniform random numbers with maximum n_user, representing users whose systems are judged to be suspicious because of their own behaviors such as run a new process, create a new file and so on

Step 3: For each user, generate a normal random number n_URL with the mean avg_URL and standard deviation 1; Generate n_URL numbers with maximum n_URL satisfy the distributive law of traffic(i) in Eq. 5 as the set of URLs visited by this user

Step 4: For each user, if any Trojan URL appears in his visited URL sets or his system state is disturbed, set the state of his system to be suspicious

TrojanURLDetector

Step 1: Read data from users and update the suspicious degree list using algorithm shown in Fig. 3

Step 2: Verify suspicious URLs which reach their threshold values

Step 3: If a Trojan URL is identified, record it suspicious degree and the current time period

Table 3: Configuration of simulation system

Term	Values
n_URL	5000
n_Engine	5
P(x)	see (6)
n_user	50000
TrojanSet	1, 5, 8, 14, 28, 48, 64, 94, 128, 150, 162, 174, 255, 283, 319, 348, 373, 404, 431, 475

Experiments: In the experiments, traffic distribution of 5000 URLs is constructed using Eq. 1 which is the traffic distribution of the world's top 1200 websites.

$$P(x) = 2.703e^{-0.02825x} + 3.139e^{-0.000974x} \tag{6}$$

The average of URLs visited per user per day represented by avg_URL can be obtained using Eq. 6, which is shown in Eq. 7.

$$\text{avg\_URL} = \left\lceil \sum_{j=1}^{5000} P(j) \right\rceil = 43 \tag{7}$$

In order to evaluate the efficiency of TrojanURLDetector, we designed 69 groups of experiments numbered 1 to 69. Set the length of time period of group i to be (24/i)h, n_URL to be 5000, n_Engine to be 5, P(x) to be Eq. 6, n_user to be 50000 and TrojanSet to be {1, 5, 8, 14, 28, 48, 64, 94, 128, 150, 162, 174, 255, 283, 319, 348, 373, 404, 431, 475}, shown in Table 3. All the 69 groups have the same configuration. In each group of experiments, we run the simulation system five times.

Results of the experiments show that the efficiency of TrojanURLDetector is in proportion to the length of system's working period and the stability of its performance is determined by the distribution of all URLs traffics.

The performance stability can be depicted in Fig. 7a and b where the working period is set as 0.5 h. Figure 7a shows the length of time before a Trojan is detected. Figure 7b shows the number of victims. It shows that the higher a Trojan URL's traffic, the sooner it will be detected and the lower a Trojan URL's traffic, the smaller the number of victims.

In Fig. 7b, we can see the performance of TrojanURLDetector is not stable especially with Trojans whose URL traffics are between 2 and 5%. That is because the differentiation degrees of these Trojan URLs and their neighbor safe URLs are low.

Previous studies on browser-based security are divided into two main areas. One focused on VM-based isolation and detection technique (Wang *et al*., 2006; Moshchuk *et al*., 2006, 2007; Anagnostakis *et al*., 2005) of which the main purpose is to execute the web content in

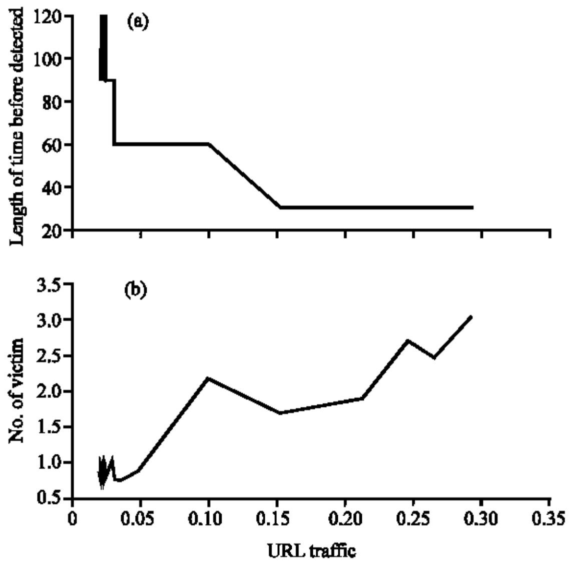

Fig. 7: Relationship between performance stability and URL traffic distribution, (a) Detecting time and (b) No. of victims

a clean VM to look for evidence of malicious side-effects, this is easy to understand, but a problem comes, since web content will be executed in a clean VM, the efficiency may be reduced, while in TrojanURLDetector, every user will install the client-side, but the efficiency won't be reduced obviously. Researchers in the other area design new concept web browsers or add security function to available browser architecture. Present study is devoted to discovering the objective relationship between the numerous static behavior characteristics and Trojan URL. Using the static algorithm, the TrojanURLDetector detects Trojan URLs, provides security warnings to users and urges web station administrators to improve the quality of their station. The following is an introduction of related research work.

Tahoma is a safety-oriented platform for web applications (Chen *et al*., 2007). Browser in this architecture runs on a VMMs-based layer named browser operating system which provides isolation between different web applications, but there also exists a problem, users' permanent data can not be saved, if saving data by users is allowed, malware can also exploit it. The OP web browser (Grier *et al*., 2008) is a secure browser whose overall design approach is to combine operating system design principles with formal methods. Vogt *et al*. (2007) enhanced the javascript engine with dynamic data tainting ability to track the flow of sensitive information in the web browser. This enhanced web browser has additional protection layer for users against XSS attacks, in this mechanism, efficiency is also a problem. SpyProxy (Moshchuk *et al*., 2007) is implemented as an extended web proxy: it intercepts user's web requests, downloads content on their behalf and evaluates its safety before returning it to users, this mechanism is faster than traditional way, but since the webpage must be loaded in Spy server, the efficiency must be reduced obviously.

Strider Honey Monkey is a pipeline of active client-side VM-based honeypots (Wang *et al*., 2006). It performs large scale, systematic and automated web patrol to detect browser based vulnerability exploit. Provos *et al*. (2008) provided a detailed study of the pervasiveness of so-called drive-by downloads on the Inter- net. They studied the relationship between the user browsing habits and exposure to malware, the different techniques used to lure the user into the malware distribution networks and the different properties of those networks, the work they have done make it clear that detecting malware with mass data in Internet is a new domain, which is also mainly mentioned in our study. In addition, they used a large scale data collection infrastructure that continuously detects and monitors the behavior of websites that perpetrate drive-by downloads.

CONCLUSIONS

This study conducted an extensive study of Trojans and described some statistical characteristics of website traffic, then proposed an effective Trojan URL detection mechanism, TrojanURLDetector, to improve the security level of the Internet. TrojanURLDetector is a set of well organized servers where, at a fast speed. Users visit URLs, at the same time it records all URLs accessed and its computer system's state changes, sends these data to servers. By comparing URLs each user visited and the health states of each user, the Trojan URL can be marked. In present experiment, an Internet simulation system was built to deployment and evaluates TrojanURLDetector. Results of the experiment show that the efficiency of TrojanURLDetector is determined by the website traffic, but it is affected by the differentiation degree of website traffic. If the differentiation degree of a Trojan URL's traffic is not sufficient, the efficiency of TrojanURLDetector will be affected. Some clustering methods may help to enhance the differentiation degree, such as cluster websites according to regions and this will be studied in future work.

ACKNOWLEDGMENT

This study was supported by National Science Foundation of China with Grant No. 60903108.

APPENDIX

Maximum of differentiation degree
Definition: Sequence differentiation degree:

$$\text{diff\_degree}(S_n) = \min\{\frac{S_{i-1}}{S_i}, i = 2,3,...n\}$$

Target: Get the maximum value of diff_degree(S_n)
Solution:
Let

$$t = \min\{\frac{S_{i-1}}{S_i}, i = 2,3,...n\}$$

Given

$$\frac{S_1}{S_2}\cdot\frac{S_2}{S_3}\cdot...\cdot\frac{S_{n-1}}{S_n} = \frac{S_1}{S_n} \quad (1)$$

$$\frac{S_{i-1}}{S_i} \geq t, i = 2,3,...n$$

$$\frac{S_1}{S_2}\cdot\frac{S_2}{S_3}\cdot....\cdot\frac{S_{n-1}}{S_n} \geq t^{n-1} \quad (2)$$

Combine Eq. 1 and 2, we get $t \leq \sqrt[n-1]{\frac{S_1}{S_n}}$. When $t = \sqrt[n-1]{\frac{S_1}{S_n}}$, $\min\{\frac{S_{i-1}}{S_i}, i = 2,3,...n\}$ gets its maximum value.

Then $t = \sqrt[n-1]{\frac{S_1}{S_n}}$ is substituted into Eq. 1,

$$\frac{S_1}{S_2}\cdot\frac{S_2}{S_3}\cdot....\cdot\frac{S_{n-1}}{S_n} = t^{n-1}$$

$$\frac{S_{i-1}}{S_i} \geq t$$

$$\frac{S_1}{S_2} = \frac{S_2}{S_3} = ... = \frac{S_{n-1}}{S_n} = t$$

That is to say, S_n is a geometric progression with a common ratio t.
Thus $S^n = at^n$, $a = S_1/t$.

Comparison between differentiation degrees: For a given discrete function $S^n = ae^{bn}+ce^{dn}$.

Its differentiation degree is:

$$\text{diff\_degree}(S_n) = \frac{ae^{b(n-1)} + ce^{d(n-1)}}{ae^{bn} + ce^{dn}}$$
$$= \frac{ae^{b(n-1)} + ce^{d(n-1)}}{e^b(ae^{b(n-1)} + ce^{d(n-1)}) + ce^{d(n-1)}(e^d - e^b)}$$

Suppose $An = ce^{d(n-1)}\,(e^d - e^b)$ (1)

Table 4: A_n of China, USA and the global

Country	$A_n(b,c,d)$
China	$0.8686e^{-0.03549(n-1)}\,(e^{-0.03549}-e^{-0.3949})$
USA	$2.557e^{-0.05346(n-1)}\,(e^{-0.05346}-e^{-0.4848})$
Global	$3.168e^{-0.009354(n-1)}\,(e^{-0.009354}-e^{-0.2768})$

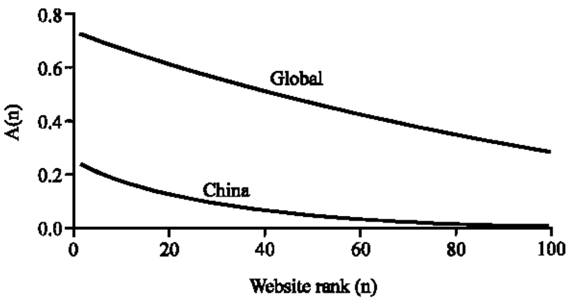

Fig. 8: The differentiation degree: China vs. global

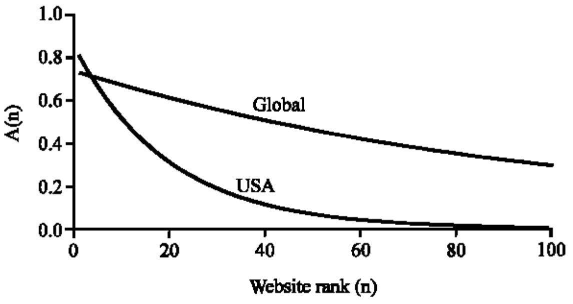

Fig. 9: The differentiation degree: USA vs. global

It can be seen that the smaller the An, the stabler the diff degree(S_n). Therefore, we can compare the differentiation degrees of two discrete functions by finding their A_n.

Taking China and USA as Examples, we compare fitting results of clustered users and that of the global Internet users. The fitting curve of Chinese websites is $P(n) = 4.744e^{-0.3949n}+0.8686e^{-0.03549n}$ and that of USA is $P(n) = 15.6e^{-0.4848n}+2.557e^{-0.05346n}$.

Substitute fitted coefficients into Eq. 1, we get Table 4 and the results are shown more intuitively in Fig. 8 and 9. From the Fig. 8 and 9, we get that the differentiation degree will be greatly increased using a suitable cluster approach.

REFERENCES

Anagnostakis, K.G., S. Sidiroglou, P. Akritidis, K. Xinidis, E. Markatos and A.D. Keromytis, 2005. Detecting targeted attacks using shadow honeypots. Proceedings of the 14th Conference on USENIX Security Symposium, August 2005 USENIX Association, Berkeley, CA, USA., pp: 9-9.

Chen, S., D. Ross and Y.M. Wang, 2007. An analysis of browser domain-isolation bugs and A light-weight transparent defense mechanism. Proceedings of ACM Conference on Computer and Communications Security (CCS), Oct. 29-Nov. 2, Alexandria, VA, USA., pp: 2-11.

Grier, C., S. Tang and S.T. King, 2008. Secure web browsing with the OP web browser. Proceedings of IEEE Symposium on Security and Privacy (SP), May 2008, IEEE Computer Society, Oakland, California, USA., pp: 402-416.

Moshchuk, A., T. Bragin, G. Steven and L. Henry, 2006. A crawler-based study of spyware in the web. Proceedings of Annual Network and Distributed System Security Symposium (NDSS), Feb. 2006, Internet Society, San Diego, California, USA., pp: 1-17.

Moshchuk, A., T. Bragin, D. Deville, S.D. Gribble and H.M. Levy, 2007. SpyProxy: Execution-based detection of malicious web content. Proceedings of USENIX Security Symposium, August 2007, USENIX Association, Boston, MA, USA., pp: 27-42.

Provos, N., P. Mavrommatis, M.A. Rajab and F. Monrose, 2008. All your iFRAMEs point to us. Proceedings of USENIX Security Symposium, July 2008, USENIX Association, San Jose, USA., pp: 1-16.

Vogt, P., F. Nentwich, N. Jovanovic, C. Kruegel, E. Kirda and G. Vigna, 2007. Cross- site scripting prevention with dynamic data tainting and static analysis. Proceedings of Annual Network and Distributed System Security Symposium (NDSS), Feb. 2007, Internet Society, San Diego, CA, pp: 1-12.

Wang, Y.M., D. Beck, X.X. Jiang, R. Roussev, C. Verbowski, S. Chen and S. King, 2006. Automated web patrol with strider honeymonkeys. Proceedings of Network and Distributed Systems Security Symposium (NDSS), Feb. 2006, Internet Society, San Diego, California, USA., pp: 1-15.