A Generic Model for Trojan Horse Attacks Prevention and Handling

Dan Tang[1,2,a], Xiao-Hong Kuang[1,2,b]

[1]Hunan police academy, Changsha, China

[2] Key Laboratory of Network Crime Investigation ,Colleges of Hunan Province, Changsha, China

[a] tacom@sohu.com [b] tacom@163.com

Keywords: Trojan horse; Information security; Generic model;

Abstract. Malicious payloads and computer codes have conventionally strived to gain access to target systems for aims which the affected end user experiences as unwanted functions or loss of data. This paper will examine the major types of Trojan horses, their mode of operation, and consequently, propose a framework for attack prevention and handling. It will highlight the need for effective control based on the premise that since Trojan attacks pose as harmless software, they have the potential to cause damage of exceptional magnitude. Ultimately, the proposed prototype will employ functional modeling to illustrate its potential as a powerful approach to information security.

Introduction

Trojan horses are a sub-type of malicious code or malicious software, generally referred to as malware [1]. There are three major types of malware, namely: (1) worms, (2) viruses, and (3) Trojan horses. However, the latter distinguishes itself from the other malware types, because of its presentation as harmless software, in contrast to operating covertly. The National Institute of Standards and Technology (NIST), for instance, categorized Trojan horses into three models, according to their approach to malicious execution (see TABLE I.

Accordingly, the proposed model handles Trojan horse attacks through (1) detection of unwarranted activity, (2) identification of vulnerable network communication nodes, and (3) detection of the malicious operation mode. Consequently, the diagnosis would inform further development of effective information security (IS) routines.

TABLE I. TROJAN HORSE OPERATION MODES

Mode	Example
Stealthily performing unrelated malicious activity in addition to perceived original function	Computer games that collect unauthorized data, e.g. passwords
Adapting original function to execute malicious activity	Software code that 'piggy backs' on login routines to collect login credentials
Completely evolving original function into malicious activity	Computer game that deletes the host's data when executed

Background

In the recent years, IS has attracted attention based on the fact that the Internet (and in turn, computer networking) has become an effective means of sending/receiving data and that a substantial number of entities depend on the capability to transmit sensitive data. Similarly, malware has informed the efforts of antivirus programs' advancement to address the potential threats [2]. However, Trojan horses have continued to circumvent the security toolsets of antivirus programs because of their operation modes.

A Trojan code defeats the purpose of the conventional antivirus approach because it depends on the end user's perception of the code as genuine software for it to install. Furthermore, on installation, it reverses the client-server paradigm by turning the infected host into a server, while the antivirus toolsets continue to assume the end user's workstation as a client node [2].

In addition, while antivirus routines cast the operation modes of software as malicious when it exhibits the tendencies of self-replication – the Trojan code evades detection by desisting from creating its copies in favor of working as a standalone code that has the potential of creating backdoors for other malware [3].

Some studies have even labeled Trojans as apt examples of malicious code that perpetrates "non-targeted attacks" [7]. In other words, while malicious routines are known to resist antivirus' capabilities by comprising their effectiveness, the Trojan horse has the ability to feign its existence by hiding its presence in the system.

Installation and Categorization of Trojans

Trojans' classification

Trojan horses commonly derive their classification from (1) how they behave, and (2) how they carry out their attacks. Hence, the major Trojan types are:

- Remote Access Trojans (RAT);
- Destructive Trojans;
- Proxy Trojans;
- Security Software disabling Trojans;
- Denial-of-Service (DoS) attack Trojans;
- Data Sending Trojans; and
- File Transfer Protocol (FTP) Trojans [3].

On the other hand, when categorized in broader terms, Trojan forms typify the approach that the NIST proposed for Trojan identification (see TABLE I. , which is based on the Trojan attack modes, namely:

- Overt attacks – that is, through information alteration and DoS;
- Covert attacks – that is, through the injection of 'logic bombs', such as the *cookie monster* Trojan reported in 2001 by McAfee; or
- A combination of overt and covert attacks – that is, through information alteration, information leakage and unwarranted resource usage, such as the *Trojan.Download.Revird* and the *Backdoor.Gaster* Trojans reported by Symantec in 2003 [7].

How users install Trojans

1) Email attachments: End users who do not interrogate the relevance of the correspondence they receive through email are prone to accept offers to install unsanctioned software code [3]. The attachments are prevalent in spam mail that disguises itself as promotional material from casinos or pornography sites.

2) File downloads: File downloads present themselves as links to genuine code, which the inattentive end user assumes as the required resource [1]. However, on keener inspection, the user can recognize that the file downloads only purport to offer functions, which ape a well-known provider. The phenomenon is common in the illegal downloads ecosystem, such as torrent downloads and pirated software or computer games.

3) Software development environments: In software development setups, such as Integrated Development Environments (IDEs), malicious agents exploit the open-source nature of popular IDEs like the NetBeans IDE to inject Trojans in the compiler routines [3]. As a result, the compiled programs become transmitting carriers for Trojan horses in their target systems.

Detection and Detactability of Trojans

Studies have concluded that Trojans do not interact directly with the hardware of their host systems, but rather, they depend on the unauthorized access to Application Programming Interfaces (APIs) [6]. The argument simplifies the *detection* of the Trojan actions because API calls provide an abstraction layer removed from the operating system of the host, and in addition, API calls are comparatively easier to analyze. To prove the *detectability* of Trojans, the Thimbleby, Anderson and Cairns study generated a relation equation between two programs (ρ and P), a Trojan routine (α) and a function (*r*) that "distinguishes their behavior; informally" [6] (see equation 1).

$$\rho \; \alpha \; P \Leftrightarrow \exists r \in R: [\rho] \; r \; \square \; [P] \; r \qquad (1)$$

The Trojan detectability function, however, assumes that the environment acts as a variable constant—such that, Trojan manifestations in one operating system like UNIX would not be similar to the manifestation in a different system; like Windows [2]. On the other hand, detectability does not mitigate the intrusion of systems, in the first place [7]. Thus, while detection of Trojans borrows from the basis of analyzing unusual system functions (hence; successful intrusions), the prevention of attacks still depends on high-level IS policies in place of "intrusion detection mechanisms" [1],[7]. The gap in the IS approach to Trojan attacks handling, as a result, informs the development of a prototype that acknowledges the intrusion-detection deficiencies in order to produce a powerful Trojan deterrent.

Trojan Attack Prevention and Handling Prototype

In the majority of cases, Trojan infections occur in computer networking environments. The hybrid nature of contemporary networks, such as the Internet, means that in addition to the simplistic client-server or star connection paradigms, network nodes also communicate in peer-to-peer (P2P) setups (see Fig. 1) [8]. The proposed model, in this case, therefore assumes that the potential Trojan infection would occur through a TCP port, since the conventional network setup would rely on the TCP/IP applications for communication [8]. For functional purposes, the model would also assume a unique port as the vulnerable gateway to the target system.

Attack technique

On installation, a conventional Trojan code would change the setting of a registry key, assumed as β. On connection to the network, the target system would allow 'backdoor entry' through port χ. The installation process for the Trojan should, however, be straightforward enough to convince the targeted user to install through, for instance, a single click of an unassuming resource, such as an attached graphic from email.

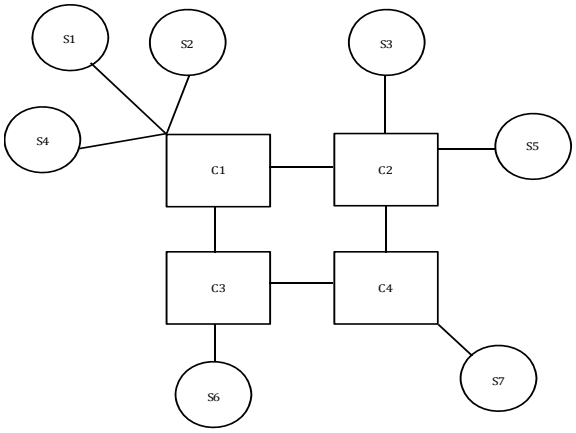

Fig1. Hybrid model networking featuring star (C→S) and P2P (C→C) communication topologies

Through port χ, the attacker achieves a TCP port connection with the target network. Henceforth, a network scan of the open ports allows the attacker to note the IP addresses of the vulnerable workstations. Through the IP addresses of the systems whose port χs are open, the attacker could exploit the Trojan horses to (1) steal login credentials, (2) spy, (3) conduct screen-watching, or (4) assume larger degrees of the victim systems' control [1],[10].

The significance of the Trojan attack paradigm is evident through the casting of the target workstation as the new server for the attackers' 'observing' signals [6]. Thus, by turning the vulnerable system into a server for the attack routine, the Trojan routine could offer up the connection details of the compromised system to the malicious agents.

Generative model of attack prevention and handling

By assuming the detectability relation (see equation 1), and a network connection N, it is feasible that the detection algorithm (outlined below) would generate the 'α' Trojan detection.

- $\alpha = \{N \cap E, N^c\}$
- $\forall\{N, E \in N^c\}\exists N^c$ Which indicates control and where $\alpha \rightarrow$ -1 on N^c
- Check the registry entry corresponding to the registry key β
- Then, if $\alpha == -1$, revert $\beta =$ 'default', else $\beta = \beta$

Accordingly, mapping the algorithm into pseudo code results in the routine shown in Fig. 2

```
Input—port ranges
Output—the current_status or required Action
Declare-- I, s, t, p as integers
Trojan_node = 0  /* the current_status of affected
node α(b)*/

Start
  for each (i=1 to n-1) /* the  number of testable
  nodes*/
      for each (N in Nc)
        for each (α(b) to Nc)
            if (α(b)= '-1' ) then Nc(i) = '-1'
              if Nc(i) = ' -1' then
              While p= 'open'
                If t= χ then
                Close t;
                Close p;
                Endif
              Endwhile
            Endif
        Endfor
      Endfor
  Endfor
```

The generative model illustrates that through the approach of detectability, an IS countermeasure could reduce the veracity of a Trojan attack. It employs mathematical set models to allow the security routines to scan a set of network nodes for vulnerability before, or on suspicion of a Trojan horse intrusion. The applicability addresses the vulnerability of workstation in a hybrid network topology and abstracts the system scans to the constituent client-server or star topology connections. However, the model's approach bases its argument on a conceptual assumption, which would not work in a production environment unless subjected to scientific testing.

Conclusion

The paper has proposed a tentative prototype for the detection and handling of a generic Trojan attack. It has assumed that the Trojan horse functions through the covert operation mode, where the malware poses as a desirable software resource, only to exploit the access privileges to create 'backdoors' for further system attacks. Through review of established Trojan horse detection routines, the paper has generated a framework, which includes an algorithm for Trojan attack detection. However, the prototype exhibits its limitations through the assumption of a networking environment where the TCP ports are fixed. Thus, its application in dynamic port assignment network topologies would challenge the framework's feasibility. Nevertheless, the paper has highlighted the significance of an IS approach, which embraces the need for proactive security measures through early detection and Trojan attack deterrence.

References

[1] P. Mell, K. Kent and J. Nusbaum, 'Guide to malware incident prevention and handling', National Institute of Standards and Technology, Gaithersburg, MD, 2005.

[2] G. Al-Saadoon and H. Al-Bayatti, 'A comparison of trojan virus behavior in Linux and Windows operating systems', World of Computer Science and Information Technology Journal, vol. 1, no. 3, pp. 56-62, 2011.

[3] H. Saini, S. Mishra and P. Sahoo, 'Defense against trojans using honeypots', IUP Journal of Science & Technology, vol. 7, no. 3, pp. 49-61, 2011.

[4] S. Sergeevich and T. Vladimirovich, 'Analysis of modern attacks on antiviruses', Journal of Theoretical and Applied Information Technology, vol. 76, no. 1, pp. 59-63, 2015.

[5] A. L. Young, 'Trojan horse programs', in Handbook of Information Security: Threats, Vulnerabilities, Prevention, Detection and Management, 3rd ed., H. Bidgoli, Ed. Hoboken, NJ: John Wiley & Sons, 2006, pp. 107-118.

[6] P. Liu, B. Niu and Z. Zhu, 'A kind of improved detection and prevention of Trojan horse based on attack tree', in Pervasive Computing and the Networked World, 1st ed., Q. Zu, M. Vargas-Vera and B. Hu, Ed. New York: Springer, 2013, pp. 374-384.

[7] H. Thimbleby, S. Anderson and P. Cairns, 'A framework for modelling trojans and computer virus infection', The Computer Journal, vol. 41, no. 7, pp. 444-458, 1998.

[8] H. Holm, 'A large-scale study of the time required to compromise a computer system', IEEE Transactions on Dependable & Secure Computing, vol. 11, no. 1, pp. 2-15, 2014.

[9] Q. Sun, X. Sun, N. Wang and Q. Wang, 'A large-scale trojans control model based on layered and P2P structure', Journal of Software, vol. 9, no. 6, pp. 1423-1427, 2014.

[10] Y. Zhang, L. Yang, Y. Zhou and W. Kuang, 'Information security underlying transparent computing: Impacts, visions and challenges', Web Intelligence & Agent Systems, vol. 8, no. 2, pp. 203-217, 2010.