Spyware-based menaces against web applications

Sergio Castillo-Perez, Joaquin Garcia Alfaro

Spyware-based Menaces Against Web Applications

Sergio Castillo-Perez[†], Joaquin Garcia-Alfaro[‡]

[†]Autonomous University of Barcelona,
Dept. of Inf. and Comm. Engineering,
Edifici Q, 08193 Bellaterra, Spain

[‡] Open University of Catalonia,
Computer Science and Multimedia Studies
Rambla Poble Nou 156, 08018 Barcelona, Spain

E-mail: *scastillo@deic.uab.es, joaquin.garcia-alfaro@acm.org*

Abstract

In the last decade, substantial progress has been made in Internet and web-based technologies. Applications related to education, health care, banking, or even social actions between individuals and groups, can highly benefit with the use of these technologies. However, computer attacks can drastically compromise web users' privacy. The spreading of Spyware in Internet applications is a proper example. We analyze in this paper the Spyware menace for compromising the security and privacy of web browser resources.

Key words: Network Security, Privacy, Data Protection, Countermeasures.

1 Introduction

The use of the web paradigm in all kinds of business models and organizations is becoming truly pervasive. Indeed, its use is becoming an emerging strategy in all kinds of application software companies [1]. It allows the design of fully interactive applications which can potentially be used by thousands of users around the world. The existence of new technologies for the improvement of traditional web features allows software engineers to conceive new services and spaces which are not longer restricted to specific operating systems. Traditional information systems related to education, health care, banking, or even emergency response, can highly benefit with the use of this technology.

The current complexity of the web paradigm has however a direct impact on the security of web browsers and more precisely in the treatment of its resources. Attacks against browsers can compromise the security and privacy of web users. This can have serious consequences given the pervasive presence of malicious software, such as Spyware. Spyware can indeed be secretly installed on Web browsers to steal sensitive data, such as user identities, passwords, and financial data [7]. Web browsers must therefore include, in addition to the expected value offered to their users, reliable mechanisms to ensure security and privacy of its users. We survey in this paper some Spyware-based techniques that can be used by malicious entities to violate web users' privacy. We present a sample scenario that shows how the privacy of a user accessing web services, may be violated by Spyware associated to the Web browser. We then discuss some defense mechanisms that might be used to reduce the risk posed by the Spyware threat.

Paper organization — The remainder of this paper has been organized as follows. Section 2 presents the Spyware threat and develops our motivation scenario. Section 3 overviews some defense mechanisms to reduce the risk of the Spyware threat. Section 4 closes the paper.

2 Spyware

Spyware is a type of malicious software (*malware*) [11] which is installed secretly on the client side to monitor users' behaviors and/or steal sensitive information, such as passwords and credit card numbers. Spyware uses Internet communications to send the stolen information towards malicious databases without the user knowledge or consent. This captured information can eventually be sold with malicious purposes, such as electronic fraud [6], identity theft, spamming, etc. Moreover, Spyware is responsible of system misbehavior, as the appearance of navigation windows with unwanted advertising, web browser hijacking, slow Internet connections speed and backdoor installation to partially control the users' system. Spyware is often complemented by evasion mechanisms, often referred in the literature as

rootkits, to ease the process of evading its presence from network and system operators, as well as to avoid being eliminated from the infected system. These mechanisms include the use of third party techniques for hiding processes, files and network connections; and the use of encryption and anti-debugging techniques. The use of these evasion mechanisms, and the corresponding change of basic system functionality features and applications, is the main reason of the system misbehavior aforementioned.

The term Spyware must not be confused with the term virus or worm. Though all three terms refer to malware, they are quite different regarding their operation modes, especially their infection mode. The main difference is that Spyware does not usually self-replicate. In other words, a system affected by Spyware does not spread the infection to other computers. Spyware is generally spread onto a system disguised as a different application that the user intends to install. In this case, the infection depends on a direct interaction required by the user. The use of non-certified applications, system updates, or plug-ins, often downloaded from malicious web sites or from malicious peer-to-peer systems, is the most common way of Spyware infection. Alternative ways of Spyware infection also include the exploitation of system application vulnerabilities. For instance, the exploitation of web browsers vulnerabilities during user browsing to force the system to download and install the Spyware. We analyze more in detail the infection process in the sequel.

2.1 Infection Process

Traditional Spyware infection mechanisms include the following two strategies: (1) infection processes started by victim users and (2) self-infection processes automatically started through existing system vulnerabilities. In the first case, malicious code contained inside a trusted piece of software gets installed into the system under user's consent. The camouflage of malicious code is often associated with illicit software, such as cracking of commercial applications or altered versions of peer-to-peer (P2P) applications and/or web browser add-ons. The use of social engineering is a perfect example of how manipulating security-unaware user into performing actions, such as opening infected emails or visiting websites associated with the installation of the illicit software. In any case, we consider that in this first category, the infection process requires user's tacit approval.

In the second case, the infection mechanism exploits existing vulnerabilities in system applications. We include in this category the subsequent infection after a user agrees to visit a malicious website or opens a document which exploits either the web browser or the visualization application. The difference now is that the infection does not require user's approval. The existence of a vulnerability here is enough to trigger the infection process by executing the malicious code hidden in the malicious web site or document. In this case, the installation of the Spyware into the system goes unnoticed by the user. This strategy is often combined with more sophisticated attacks, such as Cross-site scripting attacks (XSS for short) and/or DNS Spoofing redirections. Once the infection process has been successfully performed, the Spyware is ready to start capturing and profiling both user and system data. Different strategies can be then used by the Spyware to implement the subsequent gathering and evasion processes. We describe in the sequel the use of user space vs. kernel space strategies.

User Space Execution — In this case, the malware only has regular user privileges. Today's most common Spyware belongs to this category, since its development is straightforward — even unskilled attackers can do it based on automatic tools. Due its simplicity and the fact that such Spyware acts under user privileges, the detection and disinfection of this category of Spyware are fairly trivial. The strategies employed by this class of malware, in order to intercept information and to evade detection/removal, is mainly based on diverting the execution flow of the affected application, ceding control to a certain code of the malware. Two different techniques might be used: (1) modification of the memory addresses that are mapped to the shared libraries functions used by the affected applications; and (2) exploitation of the add-on/plug-in mechanisms provided by the affected applications. Both techniques would allow the Spyware to get complete control of the application flow, and so to implement the interception and evasion routines.

Kernel Space Execution — The previous technique, although it allows to infect user space memory areas, does not enable the Spyware to further propagate the infection throughout the system. Given that most of the affected applications running in user space are almost always stored in system space areas, the Spyware would need administration privileges to keep spreading the infection everywhere the system. The execution of the Spyware in Kernel space provides with such a functionality. Specifically, the Spyware code is now loaded and executed within the memory space assigned for operating system routines. The execution within this memory space area provides to the malware higher resistance to be detected and/or removed from the system. Indeed, the Spyware code is now running with administration privileges. This implies higher complexity in the Spyware code, which makes it less common. The common way to manage the execution of malware at kernel space relies on the modification of memory addresses associated to the operating system calls (syscalls for short).

2.2 Web Browser Infection Scenario

We show in this subsection a hypothetical scenario where the privacy of a given user, who is accessing web services, may be violated by Spyware associated to the browser. The

technique used by the Spyware of our scenario to intercept user's data is to exploit the add-on mechanisms of certain programs — in our case, the web browser. Most of today's web browsers allow this option in order to ease the adding of new enhancements by third party programmers. To do so, the main application's routine delegates the execution flow to the new functions and grants them the permission to control certain events. For instance, the main routine may grant the access to both read and modify the Document Object Model (DOM). If so, these complements may control the web browser application during the loading of a web page, the processing of uploaded/downloaded files, etc. Hence, the way that this pluggable technology works can be exploited to put in place the necessary Spyware routines.

Let us suppose that a malicious user wants to obtain valid credit card numbers with the aim of carrying out electronic fraud. Let us also assume that the malicious user finds out a given vulnerability associated to a particular version of a web browser that allows the execution of arbitrary code. Analyzing the vulnerability, our malicious user would prepare various web servers containing the necessary code that exploits the vulnerability and would spread the necessary links (e.g., by using social engineering) [6]. The visit of the victims to those malicious web sites leads to the Spyware installation by exploiting the vulnerability. The objective of the Spyware programmed by the malicious user is to capture and redirect towards a secret place those credit card numbers introduced by the victims during their electronic web-based sessions. Hence, the aforementioned pluggable techniques will be used by the installed Spyware.

To guarantee the success of the previous scenario, the attacker must persuade a large number of victims (i.e., users of the vulnerable web browser) to visit the malicious web sites. The more victims the attacker would successfully fool, the higher the attacker's profit. The time factor would also be decisive in this process since, as soon as the browser vulnerability is reported and fixed, the likelihood of success for the attack will drastically be reduced. With the objective of speeding up the infection process, several other strategies such as DNS poisoning, Spamming, and cross-site scripting attacks, are going to be used by the attacker to lead as much victims as possible towards the malicious web sites.

After the infection, the Spyware — now seen as a browser add-on — remains active, but silent, while waiting for the matching of data associated to e-commerce transactions (i.e., to capture data such as credit card numbers, expiration numbers, passwords, etc.). All the information captured by the Spyware is eventually sent, once postprocessed and protected, to the attacker's database. Let us notice that although the web browser can use cryptographic operations to ensure the authenticity, integrity and confidentiality of the information exchanged with e-commerce servers (e.g., use of SSL/TLS communications), this does not protect the data from being stolen by the Spyware which is locally executed at the client side.

If we analyze the use of SSL/TLS in the TCP/IP model, we can see that the information remains protected between the application layer (HTTPS) and the transport layer (TCP). However, since the Spyware that infected the browser is executed at the application layer (i.e., HTTPS), it intercepts the information before being protected by SSL/TLS. Therefore, it can store and modify such information before it is sent to the following layers of the TCP/IP model. We should also notice that the Spyware does not raise either any suspicious alert associated to the use of SSL/TLS certificates, since the verification process is not affected by the infection. Similarly, the use of verification codes, such as CVC2, CVV or CID, does not help either to detect or prevent the malicious activities associated to the Spyware. These codes are also provided by the users and certainly captured by the Spyware without any difficulties.

3 Prevention Techniques

We can consider three main forms of prevention to avoid the infection of a system by Spyware. The first one, more ambitious, is to encourage users to improve their computer hygiene. This includes the enhancement of their behavioral patterns, such as keeping their operating systems and associated components updated, to avoid downloading software from untrusted sources, and to ignore emails or attachments coming from unknown senders.

The second form of prevention, much more technical, can be defined as a protection design pattern. The objective is to avoid the infection of a system or to reduce the damage of the infection. This includes hierarchical protection domains, often called protection rings. The result is a layered structure of trust in the operating system. This structure of trust is generally complemented by specific hardware that allows a protected separation between trusted and untrusted processes. It provides, moreover, several levels of access to the resources of the system. Rings, or domains, are arranged in a hierarchical manner: from most privileged (most trusted) to least privileged (least trusted). This protection mechanism is specially relevant to reduce the risk of Spyware targeting the kernel level of an operating system. It increments the technical difficulties to solve to get control of the system services. Attacks would target now some kind of low-level hardware vulnerabilities [4].

The third defense consists on using automatic applications to increase the odds of detecting and isolating incoming Spyware. Depending on the detection strategy, this automatic software can be classified in two main categories: *syntactic signatures based analysis* and *semantic-aware analysis*. *Syntactic signatures based analysis* intends to collect as much information about suspicious applications as possible. It avoids moreover the execution of suspicious files. Then, using this characterization, it determines whether the file can be considered as Spyware. The main detection mechanism basically compares the suspicious file towards

a database of characterizing signatures, generally seen as a sequence of low-level instructions. This pattern-based signature database is generated by analyzing previously known Spyware samples and, as a consequence, it must be regularly updated by the users. As Moser et al. points in [10], there are some evasion schemes based on some kind of obfuscation, such as polymorphism or metamorphism, that can be used by the Spyware authors to remain undetectable. To do so, some kind of code mutation alter the files without changing its behavior. So, every infection can be performed by a different image, defeating pattern-based signatures detection.

Semantic-aware analysis [3], on the other hand, can be implemented by modeling the interaction of the suspicious files and the system environment. This strategy can be seen as a way to overcome the deficiencies of the previous strategy, since any mutation of the executable image does not affect the final behaviour of Spyware. This can be understood if we consider that syntactic signatures does not take into account the semantic of the instructions. Thus, this detection mechanism is also sometimes known as behavior-based detection. According to [5], we can classify these strategies in two categories depending how data collection to model the behaviour is extracted, that is *static* or *dynamic*. In the last years, several behavioural detection strategies has been developed based on different algorithms, such as graph isomorphism, model checking or heuristics. We encourage the reader to consult [5] for a taxonomy details. The combination of *static* and *dynamic* strategies can be merged to enhance the detection process [8]. However, these techniques are not completely effective for the detection of today's and future generations of Spyware. First, because it is quixotic to believe that we can build sets of signatures that are complete enough to detect all kinds of Spyware [9]; and second, because the strategy of determining whether a program exhibits a specific malicious behavior independent of a suspicious file is a undecidable problem [2].

4 Conclusions

We introduced in this paper the Spyware menace for compromising the security and privacy of web browser resources. This menace can especially harm web-based applications, such as online education, health care, and electronic banking. We surveyed some Spyware-based techniques that can be used by malicious entities to infect a system; and briefly introduced some prevention techniques. We plan to analyze more in deep these techniques in a forthcoming version of this paper.

Acknowledgments — We acknowledge the financial support received from the Spanish Ministry of Science and Innovation and the FEDER funds (grants TSI2006-03481, TSI2007-65406-C03-03 E-AEGIS, and CONSOLIDER-INGENIO 2010 CSD2007-00004 ARES).

References

[1] Cary, C., Wen, H. J., and Mahatanankoon, P. A viable solution to enterprise development and systems integration: a case study of web services implementation. International Journal of Management and Enterprise Development, 1(2):164–175, Inderscience, 2004.

[2] Christodorescu, M., Jha, S., Seshia, S. A., Song, D., and Bryant, R. E., Semantics-Aware Malware Detection. IEEE Symposium on Security and Privacy (S&P'05), pp.32-46, 2005.

[3] Egele M., Kruegel C., Kirda, E., Yin, H., and Song, H., Dynamic Spyware Analysis. *USENIX Annual Technical Conference*, Santa Clara, CA, June 2007.

[4] Embleton, S., Sparks, S., and Zou, C. SMM Rootkits: a New Breed of OS Independent Malware. In *SecureComm '08: Proceedings of the 4th international conference on Security and privacy in communication netowrks*, pages 1–12, New York, NY, USA, 2008. ACM.

[5] Jacob, G., Debar, H., and Filiol, E. Behavioral Detection of Malware: From a Survey Towards an Established Taxonomy. *Journal in Computer Virology*, 4(3):251–266, 2008.

[6] Garcia-Alfaro, J., Cuppens, F., Autrel, F., Castella-Roca, J., Borrell, J., Navarro, G., and Ortega-Ruiz, J. Protecting On-line Casinos against Fraudulent Player Drop-out. IEEE International Conference on Information Technology. IEEE Computer Society, April 2005.

[7] Hu, Q. and Dinev, T. Is Spyware an Internet Nuisance or Public Menace?. *Communications of the ACM*, SPECIAL ISSUE: Spyware, 48(8):61-66, 2005.

[8] Kirda, E., Kruegel, C., Banks, G., Vigna, G., and Kemmerer, R. A., Behavior-Based Spyware Detection. *USENIX Security '06*, Vancouver, Canada, August 2006.

[9] Lee, Y. and Kozar, K.A. Investigating Factors Affecting the Adoption of Anti-spyware Systems.. *Communications of the ACM*, SPECIAL ISSUE: Spyware, 48(8):72-77, 2005.

[10] Moser, A., Kruegel C., and Kirda E. Limits of Static Analysis for Malware Detection. *Computer Security Applications Conference*, 2007. ACSAC 2007. Twenty-Third Annual In Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual (2007), pp. 421-430.

[11] Skoudis, E. and Zeltser, L. Malware: Fighting Malicious Code. *Prentice Hall PTR*, 2004.