

Project Proposal

Peter Wrench

13th March 2013

1 Principal Investigator

Peter Mark Wrench

0834437934

g10w1139@campus.ru.ac.za

Supervisor: Prof. Barry Irwin

2 Project Title

The project is proposed under the title of:

A PHP Sandbox for the Dissection of Web Malcode and Remote Access Trojans

3 Statement of the Problem

The overwhelming popularity of PHP as a hosting platform [2, 28] has made it the language of choice for developers of Remote Access Trojans (or web shells) and other malicious software [6, 26]. Web shells are typically used to compromise web platforms by providing the attacker with basic remote access to the system, including file transfer, command execution, network reconnaissance and database connectivity [16]. Once infected, compromised systems can be used to defraud users by hosting phishing sites, perform Distributed Denial of Service (DDOS) attacks, or serve as anonymous platforms for sending spam or other malfeasance [19].

The proliferation of such malware has become increasingly aggressive in recent years, with some monitoring institutes registering over 70 000 new threats every day [4, 15]. The sheer volume of software and the rate at which it is able to spread make traditional, static signature-matching infeasible as a method of

detection [8, 24]. Previous research has found that automated and dynamic approaches capable of identifying malware based on its semantic behaviour in a sandbox environment fare much better against the many variations that are constantly being created [11, 8, 22, 5]. Furthermore, many malware tools disguise themselves by making extensive use of obfuscation techniques designed to frustrate any efforts to dissect or reverse engineer the code [20, 7]. Advanced code engineering can even cause malware to behave differently if it detects that it is not running on the system for which it was originally targeted [25]. To combat these defensive techniques, this project intends to create a sandbox environment that accurately mimics a vulnerable host and is capable of semi-automatic semantic dissection and syntactic deobfuscation of PHP code.

4 Research Objectives and Potential Extensions

In response to the problem detailed above, three primary objectives have been identified. An additional two objectives are also presented as potential extensions to the core project.

- Primary Objectives:
 - The development of a sandbox-based system capable of safely executing and dissecting potentially malicious PHP code. The sandbox will mimic a vulnerable host and will allow the code to run as it usually would, but without negatively affecting the machine on which it is being run. The purpose of such a system would be to identify potentially malicious actions undertaken by the code (semantic analysis).
 - An auxiliary system for performing normalisation and deobfuscation of input code prior to execution. This system would analyse the code for syntactic structures and functions that are typically associated with code obfuscation (such as *eval*) and replace them with equivalent, deobfuscated alternatives using PHP’s function overriding mechanisms (such as *__call*) [3].
 - A basic reporting mechanism for feedback on any backdoors or other potentially offensive features detected by the system. This would have to interact with the sandbox system to determine which features have been discovered.
- Secondary Objectives
 - An exploration into possible methods of code classification based on similarity to previously analysed samples. This would draw on existing work in the field of similarity analysis [30, 10].

- The construction of a taxonomy tracing the evolution of popular web shells such as c99, r57, b374k and barc0de [21] and their derivatives. This would involve the implementation of several tree-based structures that have the aforementioned shells as their roots and are able to show the mutation of the shells over time. Such a task would build on research into the evolutionary similarity of malware already undertaken by Li et al. [20].

5 Background and History

Malware can be described as software that harmfully attacks other software [18]. The first piece of malware for the PC, a virus called Brain, was developed in Pakistan in 1986 [27]. Able to spread only via media such as floppy disks, these early malware samples failed to make a large impact of any kind. The advent of the Internet and related technologies allowed malware to propagate more efficiently and led to a drastic increase in the development of malicious tools [4].

Malware can manifest itself in different ways and is made up of several subcategories, the most common of which are listed below [18]:

- Computer viruses
- Computer worms
- Trojan horses (the focus of this project)
- Rootkits
- Keyloggers
- Spyware and adware

Malware analysis is the process of extracting behaviours and capabilities from malware samples and was developed in response to the malware explosion catalysed by the advent of the Internet [4, 15, 25]. These techniques are often signature-based - they identify malware based on known patterns or signatures [9]. As a result of these increasingly sophisticated methods of detection, malware developers began to disguise their code using a process known as obfuscation [4, 15]. This involves creating code that is difficult to analyse while still being functionally equivalent to the original [25]. Code obfuscation was successful at avoiding detection by most signature-based systems until updated signatures were released [22], during which time considerable damage could be caused [14]. Research into techniques for dealing with obfuscation has therefore focused on dynamic systems that are capable of classifying code according to behaviour rather than structure [24, 20, 7].

The evolution of malware has largely been driven by a need to stay ahead of the developers of malware detection signatures [30]. As malware detectors became adept at recognising common web shells, malware coders were forced to create variations of sufficient complexity to avoid detection by these signatures [31, 10]. Genetic algorithms that synthesise new code from a pool of existing samples are often used for this purpose [23]. In addition to avoiding detection, these variations are often endowed with new features through the evolution process [30, 23].

6 Approach

The first preparatory phase of the project is divided into three parts, the first of which will involve gaining an understanding of the PHP language with particular emphasis on the function calls that are commonly employed in code obfuscation (such as *eval*). In the second part, a study of current PHP obfuscation practises will be undertaken to better understand how to reverse them [12, 29]. During the third part, several common PHP shells will be examined to become familiar with their overall structure and to identify any behaviour that they might share. These shells have already been sourced [1].

The second phase is the collection of web shells and phishing kits that can be used as inputs to the system. Samples will be collected from the Internet and the archives of the Security and Networks research group at Rhodes University.

The third phase will be concerned with the examination of related work, particularly in the field of semantic code analysis [8, 24]. Research into PHP deobfuscation conducted by Christodorescu et al. and Hyung et al. will also be studied to provide a base from which to construct the syntactic deobfuscator section of the system [11, 8].

The fourth phase of the project will be concerned with the construction of the PHP sandbox environment. Since the project involves storing and executing malicious code, care will have to be taken to ensure that the environment is isolated from the host machine to prevent any malicious actions from affecting it. This will be achieved by making use of a virtual machine running an Apache web server. If possible, a virtual file system could also be implemented as an added precaution [13, 17]. Ideally the machine should have controlled access to an Internet connection, as some malware will simply not execute if it detects that it is running in a simulated environment with no external connections [25]. Existing malware analysis tools such as Anubis and CWSandbox will be studied to discover how to achieve such a state of accurate emulation.

The main phase of the project involves the development of a system to examine PHP code. It is intended that the core system be written in Python for ease of coding. Since the system is not built for real-time dissection of code, the speed advantages offered by other languages (such as C or C++) are not an important

consideration. It is envisaged that the system will consist of three major parts: a deobfuscation segment that will attempt to strip away any unnecessary function calls, a sandbox environment that will identify any malicious features, and a reporting mechanism that will produce both the deobfuscated code and a list of its major features. An analysis of several existing implementations of PHP decoders will also be studied to gain an understanding of the main principles of the decoding process.

The final phase of the project will involve a thorough testing of the completed system. This will be achieved by comparing the results with those produced by other PHP decoding systems, most notably the system developed by Ballast Security [1]. The accuracy of the features identified by the system will also need to be determined. This will be done by acquiring a common malware tool with a known feature set and testing whether the system is able to reproduce it.

7 Requirements

The hardware requirements for this project are as follows:

- Access to a PC capable of running multiple virtual machines

The software requirements for this project are as follows:

- *Windows 7*
- *Ubuntu* and an *Ubuntu* image for use as a virtual machine
- Virtual machine software (*VirtualBox*, *VMPlayer*)
- *Python 3.3*
- A Python IDE (*PyScripter*, *PyDev*, *Eric*)

The project will also require the collection of live malware samples obtained from the Internet and the Security and Networks research group at Rhodes University.

8 Project Time Line

Activity	Proposed Deadline
Submit project proposal	<i>01 March 2013</i>
Create project website	<i>15 March 2013</i>
First Seminar: Present project outline to staff and peers	<i>19 March 2013</i>
Complete setup of virtual machine environment	<i>22 March 2013</i>
Become familiar with PHP	<i>25 March 2013</i>
Submit literature review	<i>27 May 2013</i>
Complete first draft of dissection system	<i>20 June 2013</i>
Perform first full test of draft system	<i>01 July 2013</i>
Complete fully functional system	<i>21 July 2013</i>
Second Seminar: Report on project progress	<i>06 August 2013</i>
Complete first draft of thesis	<i>20 August 2013</i>
Submit short paper	<i>16 September 2013</i>
Third Seminar Series: Present project to staff and peers	<i>29 October 2013</i>
Submit thesis	<i>01 November 2013</i>
Complete website	<i>04 November 2013</i>
Oral research exam	<i>19 November 2013</i>

References

- [1] PHP Decoder. Online, 2007. Available from: <https://defense.ballastsecurity.net/decoding/index.php?hash=d5c0c0123e7bd4c4fc7002cdb588e92d>.
- [2] Usage Stats for April 2007. Online, 2007. Available from: <http://www.php.net/usage.php>.
- [3] What’s new in php 6. In *Pro PHP*. Apress, 2008, pp. 41–52.
- [4] Malware Statistics. Online, 2009. Available from: <http://www.av-test.org/en/statistics/malware/>.
- [5] BURGUERA, I., ZURUTUZA, U., AND NADJM-TEHRANI, S. Crowdroid: behavior-based malware detection system for android. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices* (New York, NY, USA, 2011), SPSM ’11, ACM, pp. 15–26.
- [6] CHOLAKOV, N. On some drawbacks of the php platform. In *Proceedings of the 9th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing* (New York, NY, USA, 2008), CompSysTech ’08, ACM, pp. 12:II.7–12:2.
- [7] CHRISTODORESCU, M., AND JHA, S. Testing malware detectors. *SIG-SOFT Softw. Eng. Notes* 29, 4 (July 2004), 34–44.

- [8] CHRISTODORESCU, M., JHA, S., SESHIA, S., SONG, D., AND BRYANT, R. Semantics-aware malware detection. In *Security and Privacy, 2005 IEEE Symposium on* (May), pp. 32–46.
- [9] EGELE, M., SCHOLTE, T., KIRDA, E., AND KRUEGEL, C. A survey on automated dynamic malware-analysis techniques and tools. *ACM Comput. Surv.* 44, 2 (Mar. 2008), 6:1–6:42.
- [10] GUPTA, A., KUPPILI, P., AKELLA, A., AND BARFORD, P. An empirical study of malware evolution. In *Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International* (Jan.), pp. 1–10.
- [11] HYUNG CHAN KIM, DAISUKE INOUE, M. E. Y. T. K. N. Toward Generic Unpacking Techniques for Malware Analysis with Quantification of Code Revelation. Online, 2009. Available from: <http://jwis2009.nsysu.edu.tw/location/paper/Toward%20Generic%20Unpacking%20Techniques%20for%20Malware%20Analysis%20with%20Quantification%20of%20Code%20Revelation.pdf>.
- [12] JAIN, S. Malware obfuscator for malicious executables. In *Global Trends in Information Systems and Software Applications*, P. Krishna, M. Babu, and E. Ariwa, Eds., vol. 270 of *Communications in Computer and Information Science*. Springer Berlin Heidelberg, 2012, pp. 461–469.
- [13] JIANG, X., WANG, X., AND XU, D. Stealthy malware detection through vmm-based "out-of-the-box" semantic view reconstruction. In *Proceedings of the 14th ACM conference on Computer and communications security* (New York, NY, USA, 2007), CCS '07, ACM, pp. 128–138.
- [14] KALAFUT, A., ACHARYA, A., AND GUPTA, M. A study of malware in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement* (New York, NY, USA, 2006), IMC '06, ACM, pp. 327–332.
- [15] KASPERSKY, E. Number of the Month: 70K per day. Online, October 2011. Available from: <http://eugene.kaspersky.com/2011/10/28/number-of-the-month-70k-per-day/>.
- [16] KAZANCIYAN, R. Old Web Shells, New Tricks. Online, December 2012. Available from: https://www.owasp.org/images/c/c3/ASDC12-Old_Webshells_New_Tricks_How_Persistent_Threats_haverevived_an_old_idea_and_how_you_can_detect_them.pdf.
- [17] KING, S., AND CHEN, P. Subvirt: implementing malware with virtual machines. In *Security and Privacy, 2006 IEEE Symposium on* (May), pp. 14 pp.–327.
- [18] KRAMER, S., AND BRADFELD, J. A general definition of malware. *Journal in Computer Virology* 6 (2010), 105–114.

- [19] LANDESMAN, M. Malware Revolution: A Change in Target. Online, March 2007. Available from: <http://technet.microsoft.com/en-us/library/cc512596.aspx>.
- [20] LI, J., XU, J., XU, M., ZHAO, H., AND ZHENG, N. Malware obfuscation measuring via evolutionary similarity. In *Future Information Networks, 2009. ICFIN 2009. First International Conference on* (Oct.), pp. 197–200.
- [21] MOORE, T., AND CLAYTON, R. Evil searching: Compromise and recompromise of internet hosts for phishing. In *Financial Cryptography and Data Security*, R. Dingledine and P. Golle, Eds., vol. 5628 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2009, pp. 256–272.
- [22] MOSER, A., KRUEGEL, C., AND KIRDA, E. Limits of static analysis for malware detection. In *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual* (Dec.), pp. 421–430.
- [23] NOREEN, S., MURTAZA, S., SHAFIQ, M. Z., AND FAROOQ, M. Evolvable malware. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation* (New York, NY, USA, 2009), GECCO '09, ACM, pp. 1569–1576.
- [24] PREDA, M. D., CHRISTODORESCU, M., JHA, S., AND DEBRAY, S. A semantics-based approach to malware detection. *SIGPLAN Not.* 42, 1 (Jan. 2007), 377–388.
- [25] SHARIF, M., LANZI, A., GIFFIN, J., AND LEE, W. Impeding malware analysis using conditional code obfuscation, 2009.
- [26] SUNNER, M. The rise of targeted trojans. *Network Security 2007*, 12 (2007), 4 – 7.
- [27] SZOR, P. *The Art of Computer Virus Research and Defense*. Addison-Wesley Professional, 2005.
- [28] TATROE, K. *Programming Php*. Oreilly & Associates Inc, 2005.
- [29] ?URFINA, L., AND KOLÁ?, D. C source code obfuscator. In *Book of Abstracts ISCAMI 2011* (2011), Ostrava University, p. 1.
- [30] WALENSTEIN, A., AND LAKHOTIA, A. The software similarity problem in malware analysis. In *Duplication, Redundancy, and Similarity in Software* (Dagstuhl, Germany, 2007), R. Koschke, E. Merlo, and A. Walenstein, Eds., no. 06301 in Dagstuhl Seminar Proceedings, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.
- [31] WITTEN, B., AND NACHENBERG, C. Malware evolution: A snapshot of threats and countermeasures in 2005. In *Malware Detection*, M. Christodorescu, S. Jha, D. Maughan, D. Song, and C. Wang, Eds., vol. 27 of *Advances in Information Security*. Springer US, 2007, pp. 3–15.