POSTER: Automatic Generation of Vaccines for Malware Immunization

Zhaoyan Xu
Texas A&M University
z0x0427@cse.tamu.edu

Jialong Zhang
Texas A&M University
jialong@cse.tamu.edu

Guofei Gu
Texas A&M University
guofei@cse.tamu.edu

Zhiqiang Lin
University of Texas, Dallas
zhiqiang.lin@utdallas.edu

ABSTRACT

Inspired by the biological vaccines, we explore the possibility of developing similar vaccines for malware immunization. We provide the first systematic study towards this direction and present a prototype system, AGAMI, for automatic generation of vaccines for malware immunization. With a novel use of several dynamic malware analysis techniques, we show that it is possible to extract a lightweight vaccine from current malware, and after injecting such vaccine on clean machines, they can be immune from future infection from the same malware family. We evaluate AGAMI on a large set of real-world malware samples and successfully extract working vaccines for many families such as Conficker and Zeus. We believe it is an appealing complementary technique to existing malware defense solutions.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection

General Terms

Security

Keywords

Malware analysis, Malware detection

1. INTRODUCTION

Malware is a severe threat to our computer systems. As a matter of fact, they are the root-cause of most Internet attacks and illicit activities. According to a recent security report from Symantec [1], trends of malware are more apt to well-organized infections to make better profit for cyber criminals. That is, malware tends to infect a large number of victims and organize themselves into a controlled zombie army, known as a *botnet*. Such pandemic propagation can be easily fulfilled with only one malware family.

Currently most malware defense techniques on end hosts focus on the detection, and they typically fall into two different categories: signature-based detection and behavior-based detection. A signature-based approach typically attempts to extract some unique string patterns from malware binaries. However it is clear that this approach is hard to keep up with the fast increasing of unique malware samples each day in the wild due to the wide use of polymorphisms/packers in malware. Another more promising approach extracts the runtime behavior of malware (which supposed to be more stable within the same family of malware) for detection. This approach typically requires monitoring all kinds of system calls and it is typically very expensive and may cause a noticeable performance overhead on end hosts.

In addition to the detection of malware, another appealing solution is to prevent the malware from infecting the (clean) machine. To achieve this goal, a common approach is to patch the known system/software vulnerabilities within the machines. However, there is typically a long time window between the identification of vulnerabilities and the release of the corresponding patches, not mention that there are probably many zero-day (unknown) vulnerabilities inside our software. In addition, many recent successful social engineering attacks (e.g., through Email, IM, web) can exploit the vulnerability inside humans instead of software. As a result, our machines are still extremely prone to pandemic malware infections. The need for new lightweight and complementary techniques for effective malware prevention is pressing.

Interestingly, if we look at the case of pandemic diseases that can infect a large-scale human bodies, an effective and successful defense against known notorious diseases is vaccine. Once our body is immunized with a certain vaccine, we can prevent the further infection from the same disease. Then can we apply the similar idea to generate vaccines for malware immunization? At the first glance, this idea of malware vaccine might not be very appealing because it is only effective for a specific known malware infection. However, we observe that at one time, typically there are very few high-profile malware families that widely propagate across the Internet. If we can successfully generate lightweight vaccines and quickly deployed them on a wide range of machines, we can successfully prevent the infection from the specific malware we target. This complementary approach thus becomes appealing. At least it wins us time before we are able to obtain other better defense capabilities, e.g., getting some patches for the vulnerabilities from the vendors.

Our paper is motivated by the aforementioned question. We find that indeed there exist similar vaccines for certain malware that can be used to immunize machines from future infections. For example, many fast-spreading malware programs (e.g., Conficker worm) will clearly mark an infected machine as *infected* so that they can avoid wasting time and effort in re-infecting the machine again. In this case, this kind of *marks* can be considered as an effective and safe vaccine to immunize a clean machine from the same infection. We will discuss a representative example in detail in §2.1.

In this paper, we propose new techniques to automatically generate vaccines for effective and efficient malware immunization.

CCS'12, October 16–18, 2012, Raleigh, North Carolina, USA.
ACM 978-1-4503-1651-4/12/10.

More specifically, we concentrate on malware's targeted resources and design automatic analysis techniques to determine whether the manipulation of these resources can successfully prevent (or at least affect) malware's infection/execution. We treat such resources as our malware vaccines and derive concrete information needed for generating vaccines. Furthermore, we discuss how we can practically deploy our generated vaccines onto the end-host. Previous work [2] has discussed the possible existence of of virus vaccines. To the best of our knowledge, this is the first systematic work to perform program analysis to automatically generate vaccines for real-world malware immunization.

In summary, our paper makes the following contributions.

- We introduce the concept of generating malware vaccines to prevent current malware infection. We also present a simple taxonomy of malware vaccines. We believe this is a promising complementary approach for malware defense.
- We design and implement the prototype system, AGAMI, which can automatically generate malware vaccines. With a novel use of dynamic malware analysis techniques, we show that it is possible to automatically extract effective vaccines from current malware.
- We evaluate our system with a large set of real-world malware samples. Our empirical evaluation shows that it is practical to generate working vaccines for many real-world malware families, such as Conficker and Zeus.

2. PROBLEM STATEMENT

2.1 A Motivating Example

We first motivate our research with a real-world high-profile malware example, Conficker worm [5] (our conficker acsac paper), which has infected millions of computers during 2008 - 2011. There is an interesting logic inside of Conficker, the pseudo-code of which is shown in Figure 1. As we can see clearly that Conficker needs to check the existence of some mutexes in the system before it actually starts the infection. And if such mutexes already exist, Conficker will unconditionally terminate itself. This is mainly because it does not want to re-infect the machine again if it is already infected (thus a waste of time and effort, and may also cause undesired problems with double infections). More specifically, the mutex names in Conficker follow some specific patterns, e.g., "`Global \ <string>-99`", where the `<string>` is derived from the CRC32 hash of the host computer name. All Conficker malware samples within the same family follow such patterns.

```
1  …
2  name = get_computer_name();
3  mutex_string = calc_mutex_name(name, 'Global\');
4  result = open_mutex(mutex_string);
5  if (result==ERROR_FILE_NOT_FOUND){
6    create_mutex(mutex_string);
7    continue_infection();
8   }
9   else {
10   exitprocess();
11  }
```

Figure 1: Conficker Mutex Checking Logic

While this mutex checking helps Conficker to avoid double infections, we can also use this fact to against it. That is, if we can deliberately craft the same mutex names on a clean machine, then this machine can be immune from future Conficker infection. This is a perfect example of malware vaccines. We start from this motivating example and ask further questions: are there other types of malware vaccines? are vaccines prevalent in malware? how can we automatically and efficiently extract vaccines for malware immunization? The desire to answer these questions forms the basis of this paper.

2.2 Malware Vaccine Background

What is a malware vaccine? The concept of vaccine is originally from biology, which refers to a biological preparation that improves immunity to a particular disease, which is done by injecting certain agent that resembles a disease-causing microorganism. Similarly, a malware vaccine is a computational preparation that improves immunity to a particular malware program. In essence, malware, like any generic program, conducts a series of operations on system resources and outputs the computing result. These system resources in the computer system is analogue to the microorganisms in our body. In this sense, malware-targeted resources (computer organisms) are similar to those disease-causing microorganisms, the essential components of a vaccine. More precisely, we define a malware vaccine as a specific system resource (or a collection of them) that is created or used by malware in order for its normal infection and execution.

How a malware vaccine works? Based on how a malware vaccine works, we can classify it into two categories:

- It simulates the existence of certain computer organism (system resource) such that malware will exit upon the awareness of such existence, e.g., because it does not want to re-infect the victim again.
- It prevents malware from creating/accessing certain critical computer organism such that malware cannot obtain its essential resources to fulfill the functions, thus it will terminate or its functionalities will be significantly impacted/weakened.

A taxonomy of malware vaccine. Besides the aforementioned mentioned categories of malware vaccines, we can further define different types based on different perspectives.

Just like biological vaccines may not guarantee complete protection from a disease, the effectiveness of a malware vaccine can vary. Based on the effectiveness, we can classify malware vaccines into two types: full immunization (e.g., it can completely cease the malware execution) and partial immunization (e.g., it significantly affects the execution of some major functions in malware).

In terms of vaccine delivery and deployment, there could be two types: direct injection and vaccine daemon. The first type is very lightweight, e.g., a specific mutex name or file name. They can be simply injected into the target computer once and it will be effective afterwards. The second type of malware vaccines requires to run a vaccine daemon on the machine. For example, it will prevent the creation (or other access types) of certain specific files, registries, library, system services, windows, processes thus to prevent malware from obtaining critical resources or information to fulfill its functionalities.

It is worth noting that an ideal malware vaccine is those with full immunization and one-time injection. However, other types of vaccines are also useful, as shown in our evaluation.

2.3 Approach Overview

The goal of this paper is to provide a systematic framework to automatically and efficiently generate an effective vaccine for a

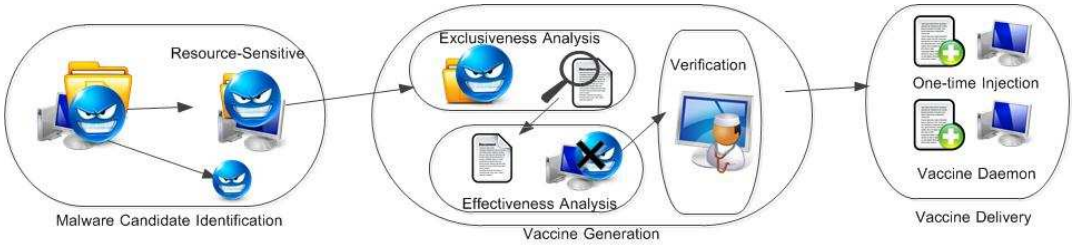

Figure 2: System Architecture

given malware sample, similar to the production of human body vaccines in biology. This certainly requires us to well understand the running behavior of malware. Thus, a basic assumption of our approach is that we can obtain the malware sample and conduct dynamic analysis. This is a reasonable assumption for any malware analysis research. Furthermore, our current analysis is based on Windows because it is the most targeted platform by malware. Our techniques could be applied to other platforms though.

Our approach is illustrated in Figure 2. At the high level, it consists of three phases: *Candidate Selection*,*Vaccine Generation* and *Vaccine Deployment*.

In the first phase, we will filter out malware samples that are unlikely to contain vaccines, at the same time profile the normal execution of the malware to obtain an overview of the malware's access to system resources. This provides us with detailed information about the sources that are accessed during the malware's infection, such as what types of resources and the corresponding resource-identifier names, what operations (e.g., create, read/write) on the resources and the corresponding results (e.g., success, fail). During our profiling, we will also apply a variant of dynamic taint analysis technique [3] to determine whether the malware's execution will be affected by certain resources it has accessed. If no program branches depend on any system resource, then we filter this malware because it does not contain vaccines that we can extract. At the end of this phase, we obtain a list of candidate resources that can affect the control flow of the malware execution.

In the second phase, our task is to generate vaccines by testing their exclusiveness, impact on malware execution, and determinism. It contains three sub-steps.

- This first step is the exclusiveness analysis, mainly used to filter these resource identifiers that have been used in other benign software, thus not exclusive to malware itself.
- The second step is to measure the potential impact of a certain system resource. We start a second-round execution monitoring by manipulating the result of the specific malware's resource operation, which will generate a manipulated trace. We apply program alignment techniques [4] to compare the concrete execution difference between the manipulated trace with the normal trace, and determine if the system resource can significantly impact the malware functions, e.g., cause malware to stop the execution. At the end of this step, we generate a list of resources that can effectively stop the malware's infection (full immunization), or significantly affect the malware's certain functions (partial immunization).
- The third step is to measure the determinism of the specific system resource identifier, e.g., *filename* or mutex name. An effective malware vaccine should be *deterministic*. A deterministic value could be a fixed/static value, or a value that is generated from a deterministic algorithm (from deterministic resources). The mutex names in Fig. 1 is a good example. To tell if a specific resource identifier is deterministic, we perform *backward taint analysis and program slicing* techniques to fully understand the identifier generation logic and the parameters it depends on. Based on that, we can further analyze the root-cause of the identifier generation, and generate a program slice responsible for such generation logic.

In the last phase, we need to deploy the malware vaccine for an end host. There are also two situations: direct injection and vaccine daemon, as mentioned previously.

3. PRELIMINARY RESULTS AND CONCLUSION

In this paper, we imitate the process of biological vaccine production and develop an automatic system, AGAMI, to extract possible malware vaccines from given malware samples with a novel use of multiple dynamic program analysis techniques. Such vaccines can be used to build a immune system at an end host to defend against the specific malware's infection.

To demonstrate the real-world practicability, we have conducted experiments on a large set of real-world malware samples (consisting of over $1,600$ samples collected from various sources). We show that many malware samples logic is commonly sensitive to several types of resources such as mutex, file operation, Windows, process, and library.

We can generate over 500 vaccines that belong to 210 malware samples. Among them, we find over 300 vaccines have static values which can be very easily deployed onto the end-host system. Meanwhile, based on the effectivness of vaccine, there are over 50 vaccines which can fully stop malware's execution.

Our results are very encouraging as a proof-of-concept study of malware vaccine generation. We believe it is an appealing complementary technique in the malware battle.

4. REFERENCES

[1] Symantec. `http://www.symantec.com/threatreport/`.
[2] *A Pathology of Computer Viruses*. Springer-Verlag, 1992.
[3] Thanassis Avgerinos, Edward Schwartz, and David Brumley. All you ever wanted to know about dynamic taint analysis and forward symbolic execution (but might have been afraid to ask). In *Proc. of IEEE S&P'10*, 2010.
[4] A.Zeller. Isolating cause-effect chains from computer programs. In *Proc. of the 10th ACM SIGSOFT symposium on Foundations of Software Engineering*, 2002.
[5] Phillip Porras, Hassen Saidi, and Vinod Yegneswaran. An Analysis of Conficker's Logic and Rendezvous Points. `http://mtc.sri.com/Conficker/`, 2009.