Topic 2 – "Cloud and DevOps"
Detecting malware during delivery and production

John Stegink
john@stegink.net

***Abstract*— When new versions of software products are released at a higher pace, questions arise as: what are the implications when a product is infected with malware, how can malware in a product be detected before it is distributed and what available monitoring tools can be used to detect malicious behavior. This document describes research into these questions for a Content Management System (CMS) running on an isolated webserver. Further research could be conducted into the enhancement of the detection of malware before distribution, systems containing personal data, and organizational structures of software companies.**

I. Introduction

In December 2020 it became known that several large companies and government organizations in the US have been the victim of a data breach as part of an attack [1]. This attack was carried out by by inserting malware in widely used security software created by Solarwinds. The malware was distributed when over 18,000 clients of Solarwinds updated their Orion[2] software through an automatic update. The malware was detected after it had been active for more than half a year. By then the malware had been used as a backdoor to access highly classified information.

Frequent updates as were used in the case of SolarWinds are made possible by using DevOps. Users and customers do expect frequent and timely updates to address faults or issues that emerge when using a software application, according to Fuggetta and Di Nitto [1]. DevOps integrates the two worlds of development and operations, using automated development, deployment and infrastructure monitoring, as described by Ebert et al. [2]. This software process enables the Continuous Integration / Continuous Delivery (CI/CD) of software, which is highly automated by using automatic testing, automatic deployment, and distribution via the internet.

Security is considered a major concern for using DevOps processes, as explained by Mohan and Othmane [3] however, not much research was conducted into monitoring and product security in the delivery phase. SecDevOps was created to make DevOps more secure. SecDevOps is the process of integrating secure development best practices and methodologies into development and deployment processes which that are made possible by DevOps. SecDevOps mainly focuses on the process itself: definition, security best practices, compliance, process automation, software configuration, team collaboration, etc. Some research has been done into how to verify the software used for DevOps, as analyzed by Paule, Dullmann, and Hoorn [4]. When the security of the process has been compromised it is very important to verify the final end product and monitor the behavior of the product.

The implications of the existence of malware in a software system depend very much on the application of the software and the that environment the system is operating in. Pai Kasturi et al. [5], state that, although over 55% the worlds websites run on Content Management Systems (CMS), little research has been done into the investigation and remediation of CMS-targeting cyber attacks. According to this paper most of the CMS's run on internet-facing web servers, which makes them vulnerable for attacks.

Malware inserted into software during the CI/CD process is different from malware that makes use of bugs or vulnerabilities in the software product. When the malware is not present in the software product when it is installed, it has to be inserted separately for every environment. The malware is inserted by taking advantage of bugs and vulnerabilities. On the contrary, when the malware is inserted by using an infected CI/CD process, the malware is present in every environment, either activated or (temporarily) deactivated. In this case the source code of the product does not contain the malware, vulnerabilities or bugs, so the developers are not aware of the presence of malware. This invisibility makes it impossible to detect malware with tools or procedures that analyse the source code.

If the malware can not be detected in the source code, it has to be detected in the final product. The check has to be done as late in the delivery process as is possible, because it is not known in advance what part of the CI/CD has been infected.

II. Ideas

The context described in the introduction gives rise to questions concerning the existence of malware in CMS systems. Therefore, a research answering the following research questions can be conducted.

A. Implications of malware in a CMS

An important question to be answered is, why it is harmful when malware is inserted in a CMS. The main purpose of a CMS is to edit and manage content for a website. Almost no personal information is stored in the system. As mentioned

[1]Lily Hay Newman, "No One Knows How Deep Russia's Hacking Rampage Goes", Wired, December 12, 2020, https://www.wired.com/story/russia-solarwinds-supply-chain-hack-commerce-treasury/

[2]https://www.solarwinds.com/solutions/orion

in the introduction, a CMS is often running on an isolated webserver. These facts could lead to the conclusion that the implications of a malware attack on a CMS is not damaging. Thus it is not important to put money or effort into preventing malware from being inserted into a CMS. This makes it important to answer the following question: *What are the implications when a CMS, infected with malware is running on an isolated webserver?*

To answer this question, a study on the implications of past attacks on CMS systems has been done. The study concentrated on 10 attacks that have been carried out within the last 5 years. These attacks have been analyzed on their most severe implications. Before the study is conducted a few implications can be summed up:

- The malware can create an efficient way to automate and execute a mass-scale level attack. The malware will be run on multiple instances of the CMS. By publishing content, it can insert malware to the connected website. This damage can be severe.
- The malware can put false information on the website. This could lead to damage to the reputation of the organization running the website. The spread of this false information can be made easier by the use of the authority of the website involved.
- The malware can be used to do a Distributed Denial of Service (DDoS) attack [3] that could lead to a fallout of websites.

B. Detection of malware before distribution

When malware has been injected in a CMS during the CI/CD process it is important to discover the injection of the malware as soon as possible. This way the damage is kept to a minimum. Because the CI/CD process can not be trusted, the software must be verified before it is distributed. At the distribution stage the software is finished and will not be modified anymore. This leads to the following research question: *How can malicious code be detected automatically when it has been inserted in a CI/CD process in a CMS before it is distributed?*

Preconditions have to be defined before the above question can be answered. First, because the main reason for using DevOps is to deliver frequent and timely updates, the detection should be done automatically. Second, the automatic detecting of malware should not become part of the CI/CD process because that would make it part of the process it is designed to check.

With these preconditions in mind, a lab experiment has been conducted that evaluated whether building a tool for detecting malware is feasible. For this experiment, a tool was developed that logs suspicious changes of a product when it is compared to the previous version of the product. The algorithm for this tool is simple. First, the tool searches for the differences in the binary code between the current and the previous version. Then, for each difference the location in the source code will be determined by the tool. Finally, the source code repository will be used to identify whether a source code change has been made that corresponds to this location. If this is not the case it will mark the change as suspicious and add it in to a logfile.

The lab experiment used infected CI/CD processes that infected a CMS with extra software. The log file was analyzed on two criteria. First, a check was done of how much of the inserted malicious code was marked as suspicious. Second, a check was done of how many false indications on possible malware were logged.

C. Monitoring malware after distribution

Unfortunately it is not possible to detect all malware before distribution, so malware could be running on a server unnoticed. It is still important to detect this malware so it can be stopped. Monitoring systems, as described by Ebert et al. [2], monitor system aspects such as the use of system resources and network traffic. To detect malicious code, an administrator will have to be notified when an anomaly is detected. This anomaly could indicate a security threat. The question is whether monitoring tools that are currently on the market can be used to detect malware in a CMS system. Or to be more specific: *When a CMS is running on an isolated webserver, what available monitoring tools can be used to detect malicious behavior?*

To answer this question the most popular monitoring tools have been compared to recent attacks. In section II-A a list of 10 analyzed recent attacks was compiled. A list of the most popular monitoring tools with their features has also been constructed. All features of the monitoring tool where enumerated by studying the documentation of the tool. This way, a table was constructed that contains a combination of the sort of attacks and whether each of the monitoring tools could have detected the presence of malware.

III. Future Work

Particularly the study on the detection of malware before distribution is very promising. Only one algorithm was considered, it would be very interesting to do a study on the use of AI in the detection of anomalies that indicate the presence of malware.

Apart from CMS systems, every software product could potentially be infected with malware. It would be very interesting to research systems that contain personal data, because the implications could be even larger.

With time, more and more information about the Solarwinds hack will probably be available. This information could be used as a subject for a case study. This study could be used to learn about how to prevent such attacks in the future.

Research, that is not technical-oriented, could also be conducted into this subject. Examples of research could be, the correlation organizational structure, or the level of stock market orientation of the companies developing website systems and the vulnerability of the produced software by these organizations.

[3]Denial-of-service attack: https://en.wikipedia.org/wiki/Denial-of-service_attack

REFERENCES

[1] A. Fuggetta and E. Di Nitto, "Software process," in *Proceedings of the on Future of Software Engineering*. ACM, 2014, pp. 1–12.

[2] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "Devops," *IEEE Software*, vol. 33, no. 03, pp. 94–100, may 2016.

[3] V. Mohan and L. Othmane, "Secdevops: Is it a marketing buzzword? - mapping research on security in devops," in *2016 11th International Conference on Availability, Reliability and Security (ARES)*. Los Alamitos, CA, USA: IEEE Computer Society, sep 2016, pp. 542–547. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ARES.2016.92

[4] C. Paule, T. F. Dullmann, and A. V. Hoorn, "Vulnerabilities in continuous delivery pipelines? a case study," in *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*. Los Alamitos, CA, USA: IEEE Computer Society, mar 2019, pp. 102–108. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICSA-C.2019.00026

[5] R. Pai Kasturi, Y. Sun, R. Duan, O. Alrawi, E. Asdar, V. Zhu, Y. Kwon, and B. Saltaformaggio, "Tardis: Rolling back the clock on cms-targeting cyber attacks," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 1156–1171.