Do Configuration Management Tools Make Systems More Secure? An Empirical Research Plan

Md Rayhanur Rahman
mrahman@ncsu.edu
North Carolina State University
Raleigh, NC, USA

William Enck
whenck@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Laurie Williams
lawilli3@ncsu.edu
North Carolina State University
Raleigh, NC, USA

ABSTRACT

Configuration Management Tools (CMT) help developers manage the system and installed application in an automated and efficient manner. However, misconfiguration in these tools can make a system vulnerable to compromises. Whether the usage of these tools makes the systems secure - this question can only be answered through empirical evidence. Hence, we propose a empirical research plan on the impact of CMT on systems where these tools have been applied. As a case, we will investigate the case of Endpoint Linux Management System managed by Puppet, a popular configuration management tool.

CCS CONCEPTS

• **Configuration Management Tools**; • **Static Analysis**; • **Security Smells**; • **Puppet**;

KEYWORDS

Configuration Management Tools, Puppet, Security Smells, Static Analysis

ACM Reference Format:

Md Rayhanur Rahman, William Enck, and Laurie Williams. 2020. Do Configuration Management Tools Make Systems More Secure? An Empirical Research Plan. In *Hot Topics in the Science of Security Symposium (HotSoS '20), April 7–8, 2020, Lawrence, KS, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3384217.3384223

1 INTRODUCTION

Configuration Management Tools (CMT) have gained popularity in recent years, mainly in cloud computing domains [4]. A large number of computing nodes are being managed by these tools such as Puppet [7], Chef [3] or Ansible [10]. These languages facilitate software practitioners and IT staff in the provisioning of the Cloud infrastructure, deploying the application, and orchestrating the overall system configuration behavior. These tools are also being used to manage and automate software configurations, to enforce their characteristics in computing nodes, and to provision the node activities in distributed computing infrastructures. Without the help of these tools, human intervention is needed for managing every configuration changes in software, hosts, and overall systems which is exhaustive, inefficient, and error-prone. CMT vendors also claim that usage of their tools brings about more efficiency, reliability, and security in systems that are previously managed by human intervention [6].

Consequently, IT industries have switched to the use of CMT for software development, deployment, and provisioning. However, switching to CMTs from a human operator-based workflow requires rigorous documentation, training, and business process changes which incur costs in terms of money and time. However, these tools do not guarantee that the systems managed by these tools will be secure. For example, NordVPN has suffered from a leak of its cryptokeys due to a misconfiguration in their configuration scripts [2]. Such occurrences necessitates the need for more empirical evidence regarding the impact of CMT. As a case, we will consider the Endpoint Linux Management System of North Carolina State University (NCSU) where Puppet has been used as the CMT for the past two years. We propose to run an empirical analysis on how the usage of Puppet impacts the secureness of Endpoint Linux Management System and other systems where the subsequent effects can be propagated at scale.

Below we provide the goal of our study, the research questions, and the tasks we need to perform to answer our research question.

The goal of this research is *to aid information technology (IT) professionals understand whether the usage of CMT leads to more secure systems through an empirical study of a CMT managed system and repository.*

We empirically evaluate our goal statement by answering the following research questions:

(1) Does the usage of CMT make the Endpoint Linux Management System more secure?
(2) Does misconfiguration in CMT affect the security of the system?
(3) What are the characteristics of the evolution of the CMT repository in the case of breaches?
(4) Are the CMT scripts written according to the best practices from a security point of view?

2 RESEARCH PLAN

We will perform an empirical study on the NC State Endpoint Linux Management System [5] which is being managed by Puppet, one of the popular CMT. Prior to the use of Puppet, the Linux hosts were managed through a plethora of custom shell and bash scripts along with human activities when system configuration needed to change in response to some situations, for example, updating the Linux kernel or installing patches to a vulnerable module. The endpoint systems is being managed by Puppet for last two years and hence, this scenario is ideal for evaluating the impact of CMT from the

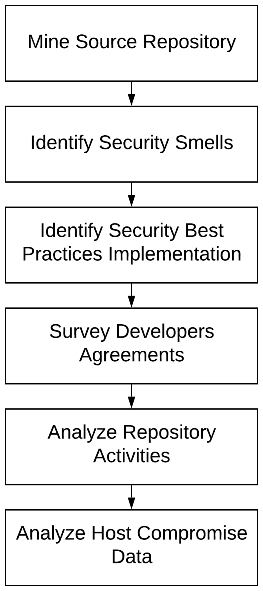
Mine Source Repository
Identify Security Smells
Identify Security Best Practices Implementation
Survey Developers Agreements
Analyze Repository Activities
Analyze Host Compromise Data

Figure 1: Research Steps

aspect of overall security in the system. We will perform the following analysis as shown in Figure 1 to answer the aforementioned research questions.

- **Step One: Mining Source Repository:** We will gather source code, commit logs, and other source control activities such as pull requests, merge conflicts, feature branches from the Github repository of NC State Endpoint Linux Management Systems.
- **Step Two: Identify Security Smells:** We will apply static analysis techniques to detect the presence of security smells in the Puppet scripts. Security Smells are the occurrence of insecure coding patterns in the sources which introduce security weakness into the software and might lead to software breaches [9]. Most prominent security smells that can be found in Puppet scripts are hardcoded secrets, use of HTTP without TLS, suspicious comments, admin by default etc [1, 8].
- **Step 3: Identify Security Best Practices Implementation:** We will also apply static analysis techniques to identify whether the best security practices have been thoroughly followed. Puppet, like other CMT, can be complicated at times. Hence developers should follow good, recommendable and high-level best practices such as use of inventory, private modules, dry runs before deployments [**?**], otherwise scripts will become difficult to extend and maintain, consequently the system will be more prone to misconfiguration which will lead to less secured system.
- **Step 4: Survey Developers Agreement:** After the identification of security smells and security best practices, we will conduct a survey with the developers of the scripts. We will study whether developers agree with the identified instances of security smells and good practices or whether the particular instance is a false alarm or whether the instance is contextually irrelevant.
- **Step 5: Analyze Repository Activities:** We will analyze the repository activities to study the behaviour of developer activities in response to the event of a compromise. We will analyze the reported bugs, issues, patches, pull requests, and commit logs to understand the development activities and nature of fixing the issues regarding the compromises in hosts. We will also look for any testing activities that has been performed to evaluate the effectiveness of fixes.
- **Step 6: Analyze Host Compromise Data:** We will gather the data of host compromises and whether they have been caused by misconfiguration or misuse in Puppet scripts or security smells or vulnerable modules from Puppet or operating systems or other applications.

3 CONCLUSION

CMT have made developers more productive and efficient in managing and provisioning the computing nodes and installed applications. However, these tools might make the systems susceptible to software weakness through misconfiguration or vulnerable modules. We propose a research plan to empirically investigate the impact of the usage of CMTs on systems and application in terms of security of the systems. For the case study, we have chosen Endpoint Linux Management system for NC State University. From the preliminary findings of our work, we found out a hundred of security smell occurrences which could pose a threat to the overall security and rough edges of their tools.

REFERENCES

[1] [n.d.]. CWE: Common Security Weakness. https://cwe.mitre.org/. [accessed:1/6/19].
[2] Ars Technica. [n.d.]. Hackers steal secret crypto keys for NordVPN. Here's what we know so far. https://arstechnica.com/information-technology/2019/10/hackers-steal-secret-crypto-keys-for-nordvpn-heres-what-we-know-so-far/. [accessed:1/24/20].
[3] Chef. [n.d.]. Chef. https://www.chef.io/. [accessed:1/24/20].
[4] FLEX ERA Blog. [n.d.]. Cloud Computing Trends: 2016 State of the Cloud Survey. https://www.flexera.com/blog/cloud/2016/02/cloud-computing-trends-2016-state-of-the-cloud-survey/. [accessed:1/24/20].
[5] NC State University. [n.d.]. github:ncstate-linux/infrastructure. https://github.ncsu.edu/ncstate-linux/infrastructure. [accessed:1/24/20].
[6] Puppet. [n.d.]. Maximizing IT Security with Configuration Management. https://puppet.com/resources/whitepaper/maximizing-it-security-configuration-management/. [accessed:1/24/20].
[7] Puppet Labs. [n.d.]. Puppet. https://puppet.com/. [accessed:1/24/20].
[8] Akond Rahman, Chris Parnin, and Laurie Williams. 2019. The seven sins: Security smells in infrastructure as code scripts. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 164–175.
[9] Rayhanur Rahman, Akond Rahman, and Laurie Williams. 2019. Share but Beware: Security Smells in Python Gists. In *Proceedings of the International Conference on Software, Maintenance and Evolution* (Ohio, USA) *(ICSME '19)*.
[10] Redhat. [n.d.]. Ansible. https://www.ansible.com/. [accessed:1/24/20].