Shhh!: 12 Practices for Secret Management in Infrastructure as Code

Akond Rahman
Department of Computer Science
Tennessee Tech University
Cookeville, TN, USA
Email: arahman@tntech.edu

Farhat Lamia Barsha
Department of Computer Science
Tennessee Tech University
Cookeville, TN, USA
Email: fbarsha42@tntech.edu

Patrick Morrison
IBM
Research Triangle Park
Durham, NC, USA
Email: pjmorris@us.ibm.com

***Abstract*—Despite being beneficial in automated provisioning of computing infrastructure at scale, infrastructure as code (IaC) scripts are susceptible to containing secrets, such as hard-coded passwords. A derivation of practices related to secret management for IaC can help practitioners to secure their secrets, potentially aiding them to securely develop IaC scripts. *The goal of the paper is to help practitioners in secure development of infrastructure as code (IaC) scripts by identifying practices for secret management in IaC*. We conduct a grey literature review with 38 Internet artifacts to identify 12 practices. We identify practices that are applicable for all IaC languages, e.g., prioritized encryption, as well as language-specific practices, such as state separation for Terraform. Our findings can be beneficial for (i) practitioners who can apply the identified practices to secure secrets in IaC development, and (ii) researchers who can investigate how the secret management process can be improved to facilitate secure development of IaC scripts.**

***Index Terms*—configuration as code, configuration scripts, devops, devsecops, empirical study, grey literature, infrastructure as code, practices**

I. Introduction

Infrastructure as code (IaC) is the practice of using dedicated programming constructs to provision computing infrastructure at scale [14]. Practitioners use languages, such as Ansible [3], Puppet [25], and Terraform [37] to implement the practice of IaC. Adoption of IaC has resulted in benefits for practitioners. For example, Splunk [1], an information technology (IT) organization that provides cloud-based monitoring utilities, used Puppet to reduce its deployment time from several weeks to a couple of hours [26]. Swisscom [2], a Switzerland-based telecommunication provider, used Ansible to save 3,000 hours of IT administration work [5].

Despite reported benefits, IaC scripts are susceptible to include secrets [32], [31], such as hard-coded passwords and private SSH keys. Figure 1 shows an example of a secret, i.e., a hard-coded password. The code snippet presented in Figure 1 is obtained from an open source software (OSS) repository [3]. Presence of secrets is prevalent in the OSS domain: in recent work, Rahman and Williams [32] reported 9,175 hard-coded passwords to appear in a collection of 61,097 OSS IaC scripts. According to Common Weakness Enumeration (CWE) [24], "*if hard-coded passwords are used, it is almost certain that malicious users will gain access to the account in question*", which makes presence of secrets in IaC scripts to be detrimental to the security of provisioned computing infrastructures.

Prevalence of secrets in IaC scripts necessitates integration of adequate secret management in IaC development. Such integration, however can be challenging due to lack of practices that are pivotal to manage secrets. In the case of IaC, tools, such as Ansible Vault [4], Chef Vault [6], and Hiera [25] are available to store secrets respectively, for Ansible, Chef, and Puppet. Yet, a lack of practices related to secret management for IaC can limit the use of the aforementioned tools. Systematic derivation of practices for secret management can aid practitioners on how to use the aforementioned secret management tools adequately, so that secrets used in IaC development are secured. Furthermore, practitioners can use the derived list of practices as a benchmark against which to compare their current practices for secret management in IaC.

One approach to deriving practices for secret management is to analyze Internet artifacts, such as blog posts and video presentations. Instead of academic forums, such as research conferences practitioners are likely to expresses their perceptions in Internet artifacts [8]. In prior work, researchers have acknowledged the value of Internet artifacts in deriving practices, and analyzed Internet artifacts to summarize security practices [35], [22]. Analysis of Internet artifacts can be useful to synthesize practices for secret management in IaC.

The goal of the paper is to help practitioners in secure development of infrastructure as code (IaC) scripts by identifying practices for secret management in IaC.

We answer the following research question: ***RQ:* What practices can be used for secret management in infrastructure as code scripts?**

We use grey literature [13] review where we apply open coding [33] on 38 Internet artifacts, such as blog posts to identify practices related to secret management in IaC. We also identify which practices are specific to a secret management tool, such as Puppet Hiera [25], and practices that are applicable for secret management tools in general.

Contributions: We list our contributions as follows:

[1] https://www.splunk.com/en_us
[2] https://www.swisscom.ch/en/residential.html
[3] https://github.com/alphagov/puppet-graphite

```
class graphite::params {
  $admin_password =                                                    Hard-coded password
    ↪ 'sha1$1b11b$edeb0a67a9622f1f2cfeabf9188a711f5ac7d236' <-----'
  $bind_address = '127.0.0.1'
  $port = 8000
  $root_dir = '/opt/graphite'
  $version = '0.9.12'
  $worker_processes = 2
  $group = 'www-data'
  $time_zone = 'UTC'
  $memcache_hosts = []
}
```

Fig. 1: Example of an IaC script including a secret, i.e, a hard-coded password.

- A list of practices that practitioners can follow while managing secrets for IaC scripts; and
- A mapping between identified practices and secret management tools for IaC.

We organize the rest of the paper as follows: in Section II we discuss background and related work. We provide the methodology of our paper in Section III. We provide and discuss our findings respectively, in Sections IV and Section V. We conclude our paper in Section VI.

II. Background and Related Work

In this section, we discuss background information and related work necessary for the paper.

A. Background

IaC is the process of automatically managing configurations and provisioning computing environments at scale using source code [14]. Along with automation IaC also advocates the application of recommended software engineering practices, such as linting, testing, and use of version control. A dedicated programming language allows specifying the necessary environment settings, such as required libraries or the amount of RAM for computing environments.

IaC languages, such as Ansible [3], Chef [6], and Puppet [25] have dedicated syntax that can be used to accomplish IT administration tasks. We present an example in Figure 2, where a user called 'testuser' is created in three IaC languages, namely, Ansible, Chef, and Puppet. As shown in Figures 2a, 2b, and 2c respectively, the syntax for creating the user 'testuser' is different for Ansible, Chef, and Puppet. For Ansible, creation of the user occurs using a play, e.g., `Create user 'testuser'` is a play. In the case of Chef and Puppet `resources` are used to create the user even though the syntax of specifying resources are different between Chef and Puppet. Specifying two properties of the file namely 'uid' and 'comment' are also different across Ansible, Chef, and Puppet.

B. Related Work

Our paper is closely related to prior research on IaC scripts. In prior work researchers have focused on code maintainability aspects, e.g., Schwarz [34] and Bent et al. [39], in separate research studies investigated code maintainability aspects of IaC scripts. Quality issues of IaC scripts have also garnered interest. Rahman et al. [27] constructed a defect taxonomy for IaC scripts that included eight defect categories. In another work, Rahman et al. in separate studies have also quantified security weaknesses that appear in Ansible [31], Chef [31], and Puppet scripts [29]. Testing issues in IaC scripts have also been investigated by researchers. Hanappi et al. [10] investigated how the convergence of IaC scripts can be automatically detected, and proposed an automated model-based detection framework for convergence. Ikeshita et al [17] reported testing of IaC scripts can be time-consuming, and proposed a test suite reduction technique for IaC scripts. Hummer et al. [15] observed that testing in IaC is different to that of GPLs, as testing in IaC necessitates testing of remote production environments. Rahman et al. [28] identified five development anti-patterns for IaC scripts. In recent work, Kumara et al. [19] conducted a grey literature review to identify IaC development practices. The difference between our paper and Kumara et al. [19]'s work is that Kumara et al. [19] did not investigate practices for IaC-related secret management, whereas our paper solely focuses on identifying practices for secret management in IaC.

III. Methodology

We conduct a grey literature review to identify practices for secret management in IaC. A grey literature review is the process of reviewing and synthesizing content included in Internet artifacts, such as blog posts and video presentations [13]. A grey literature review is different from a systematic mapping study (SMS) or a systematic literature review (SLR), as in the case of a SMS or a SLR, researchers use peer-reviewed articles indexed in scholar databases. In the case of grey literature review, researchers use artifacts that are available on the Internet, and not peer-reviewed. We take motivation from prior research related to security practices where researchers have used Internet artifacts to derive security practices that can be integrated into the software development process [38], [35]. Our hypothesis is that by systematically collecting and analyzing we also can identify practices for secret management in IaC.

An overview of our methodology is presented in Figure 3. We describe each step of our methodology as follows:

A. Search for Internet Artifacts

We collect Internet artifact using a set of search strings. We start with the search string 'secret management practices for infrastructure as code', as our research study focuses on identifying secret management practices related to IaC. We collect the top 100 relevant search results as determined by the Google search engine. From manual inspection of the 100 search results, we observe practitioners to mention the following secret management tools: 'ansible vault', 'chef vault', 'puppet hiera', and 'hashicorp vault'. As these tools are used to manage secrets, we include the names of these tools as part of search string construction process. Furthermore, the phrase 'configuration as code' is also used as a synonym to refer to IaC [30]. Based on the above-mentioned observations, we altogether use six search strings to conduct our search process, which are listed below:

```
- name: Create user 'testuser'
  user:
    name: testuser
    comment: "testuser is a temporary user"
    uid: 1234
```

(a)

```
user 'testuser' do
  comment 'testuser is a temporary user'
  uid 1234
end
```

(b)

```
user {'testuser':
  comment => 'testuser is a temporary user',
  uid => '1234',
}
```

(c)

Fig. 2: Creation of a user with user name 'testuser'. Figures 2a, 2b, and 2c presents the syntax of creating a user respectively, in Ansible , Chef, and Puppet.

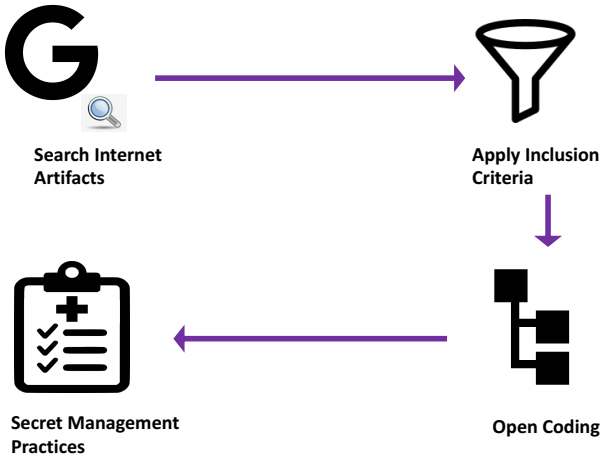

Fig. 3: An overview of our methodology.

- 'secret management practices for infrastructure as code'
- 'secret management practices for configuration as code'
- 'practices for ansible vault'
- 'practices for chef vault'
- 'practices for hashicorp vault'
- 'practices for puppet hiera'

For each of the 6 search strings we collect the top 100 relevant search results returned by the Google search engine. We use the Google Chrome web browser in incognito mode to collect the search results.

B. Apply Inclusion Criteria

We collect 600 search results collected using the 6 search strings listed in Section III-A. We apply an inclusion criteria to identify Internet artifacts that are relevant for our research study:

- The artifact is written in English;
- The artifact is available for reading;
- The artifact explicitly discusses secret management for IaC;
- The artifact is not a duplicate of another Internet artifact; and
- The artifact mentions at least one practice for secret management related to IaC.

C. Open Coding to Determine Practices

We use a qualitative analysis technique called open coding [33] with the collected Internet artifacts to answer the research question. Open coding is a technique for qualitative analysis that can summarize the latent theme from unstructured text data [33]. Our hypothesis is that by analyzing Internet artifacts we will have an understanding of secret management practices related to IaC. We use open coding as it can be used to extract patterns from Internet artifacts, as done in prior work [11], [19], [35]. The first author of the paper, who has 5 years of experience in IaC development conduct open coding. As part of the open coding process, the first author reads each Internet artifact, and extracts any mentioned practices. If multiple practices yield similar meaning, they are merged together as one practice.

IV. Results

From our set of 6 search strings we obtain 600 results sorted based on relevance as determined by the Google search engine. Upon applying an inclusion criteria stated in Section III-B we obtain 38 Internet artifacts. A complete breakdown of our filtering process is shown in Figure 4. Our set of 38 Internet artifacts consists of 2 YouTube videos, 2 question and answer posts, 1 Slideshare presentation, and 33 blog posts.

Practices for Secret Management in Infrastructure as Code: From our analysis of Internet artifacts we identify 12 practices, which we present below based on how many Internet artifacts mention the practice. Below, each practice name is followed by the count of Internet artifacts that mention the practice. For example, the practice of data organization is mentioned in 11 of the 38 Internet artifacts.

I. ***Data Organization*** (11): This category of practice is related to the organization process of secrets when IaC scripts are interfaced with secret management tools. We identify three practices that belong to this category:

- *Adequate directory structure*: The practice of using adequate directory structure when managing secrets with secret management tools. Adequate directory organization helps in better maintenance. According to one practitioner: "*Following a proper directory structure for Ansible variables, vaults, main tasks will help in easy understanding and incur less time consumption*" [2]. One practitioner strongly discourage storing all secret-related artifacts in one single directory saying "*DO MAKE USE OF FOLDERS! For the love of god, [do] this. Putting all files in a single folder both makes that a BIG folder, but also introduces a namespace collision*" [9]. For Ansible, we observe practitioners to recommend specific structures, for example, creating separate sub-directories called 'group_vars', 'vars', and 'vault'.
- *Hiera data hierarchy*: The practice of placing secret data at the appropriate hierarchy when using Hiera. The practice

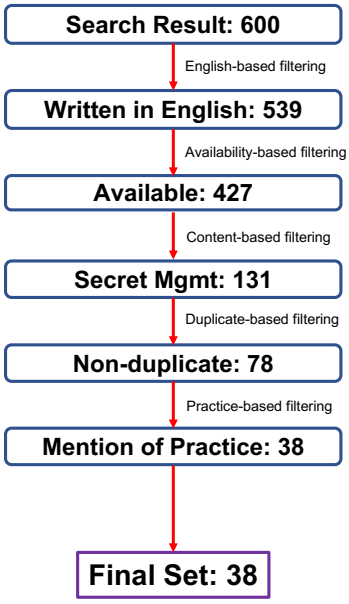

Fig. 4: Application of our inclusion criteria to collect our set of 38 Internet artifacts that we use in our paper.

is specific for Hiera, a secret management tool that can be used to manage secrets for Puppet. Even though Hiera helps practitioners to avoid hard-coding secrets in Puppet scripts, use of Hiera can negatively impact readability. To mitigate the problems with readability, practitioners advocate to store secrets that are used by few Puppet scripts at the top of the hierarchy, whereas the secrets used by many Puppet scripts should be stored at the bottom. In this manner, readability can be achieved as the practitioner will know the secrets that are used by many Puppet scripts are available at the bottom of the Hiera data hierarchy, whereas the secrets listed at the top of the hierarchy have limited use cases. We provide an example in Figure 5 that demonstrates how secrets shared across a few scripts can be placed at the top level (line#2-3), whereas secrets used by all Puppet scripts can be placed at the bottom level (line#11-12).

- *Naming conventions*: For better readability and maintenance, practitioners state the use of dedicated naming conventions for variables for managing secrets. According to one practitioner, naming conventions should be applied from the beginning of IaC secret management: "*Keep a convention to use secrets! You will not be able to review changes to the secrets and you will not be able to grep for ansible variables in secrets files! So be thorough since the beginning*" [40]. One practitioner recommended use of a prefix or a suffix so that variables that are used in secret management are easily recognizable. For Ansible, another practitioner recommend the use of the prefix `vault_` stating "*The vault variables should be written starting with 'vault_'. This will help in differentiating easily the vault variables and normal variables*".

```
1 hierarchy:
2   - name: "Secret Data for Script#1"
3     path: "secret_dir/script1.yaml" # Path to YAML file
        ↪ with secrets for Script#1
4
5   - name: "Secret Data for Script#2-10"
6     path: "secret_dir/script10.yaml" # Path to YAML
        ↪ file with secrets for Script#2-10
7
8   - name: "Secret Data for Script#11-25"
9     path: "secret_dir/script25.yaml" # Path to YAML
        ↪ file with secrets for Script#11-25
10
11  - name: "Secret Data Used by All Scripts"
12    path: "common.yaml" # Path to YAML file with
        ↪ secrets used by all scripts
```

Fig. 5: An example to demonstrate how secret data can be organized adequately in Hiera.

II. ***Password Management for Ansible Vault*** (9) : This category refers to practices unique to managing passwords for Ansible Vault. In Ansible Vault, a password is used to encrypt necessary secrets with a file called '*vault*'. The password is later used to decrypt the encrypted secrets. We observe practitioners to suggest two practices on how to manage the *vault* files:

- *Separation of vaults from the version control system*: The *vault* file must not be committed to a version control system (VCS). One practitioner recommended the use of automation scripts that can check if a *vault* file is being committed to a VCS. The practitioner observes that the `ansible-vault [encrypt|decrypt]` command, which is typically used to add/edit vault files, is susceptible of committing a *vault* file in the version history [1]. According to the practitioner, a Git Hook can be used to check if *vault* files are being committed to the VCS, potentially preventing the *vault* file from being included in the VCS [1]. A Git hook is a custom script that executes a set of actions when an event, such as a commit or a push occurs in a Git-based VCS [21].
- *Use of command line utilities*: We observe use of command line utilities, such as `--vault-id` to be a recommended approach to provide Ansible Vault passwords. One practitioner stated: "*Since Ansible 2.4, the recommended way to provide a vault password from the cli is to use the* `--vault-id` *cli option*" [4]. Without the use of command line utilities, passwords for Ansible Vault need to be provided using a file, which can be accidentally pushed into a VCS.

III. ***Access Control*** (6) : The practice of applying access control policies on secrets as well as artifacts that are used to store secrets. Practitioners recommend restriction on who stores and reads secrets so that a person or a machine with unauthorized access is incapable of using these secrets. Practitioners also recommend application of access control for artifacts, such as *vault* files used in Ansible Vault to mitigate unauthorized access.

IV. ***Prioritized Encryption*** (6): The category of practices related to prioritizing what secrets need to be encrypted. Practitioners have recommended against encrypting all data provided in IaC scripts. Encryption of all data can lead to maintainability issues as suggested by one practitioner: "*Do not take all the variables in vault encryption, otherwise it will be difficult for reviewing in case of errors, if occurred*" [2]. While managing secrets, practitioners need to identify what secrets can be consequential and prioritize those secrets for encryption. For example, a private SSH key to interface with an AWS instance can be prioritized over a user name called 'temp', that is used to create a temporary user in the AWS instance.

V. ***Separation*** (4) : The practice is related to separation of concerns, which asserts that the software artifacts should be separated based on the computing task that is being accomplished [16]. In the case of secret management for IaC, practitioners recommend separating secret management environments so that secrets are grouped based on a certain characteristic. We observe practitioners to recommend two practices related to separation:

- *Artifact separation*: The practice of using separate artifacts when managing secrets. For example, while describing secret management practices for Ansible, a practitioner advocated for using separate $vaults$ for separate tasks stating "*Although it's convenient to have a single vault with all of the encrypted secrets, a better security practice is to separate the secure credentials into separate relevant vaults*" [4]. In this manner, with separate $vaults$ practitioners are aware of what $vault$ file is used for what purpose, potentially mitigating maintainability issues.
- *State separation*: This practice is applicable for Terraform states. States refer to the snapshots managed by Terraform, where each snapshot refers to the provisioned infrastructure and corresponding configuration at a certain point in time [37]. States are stored as '$tfstate$' files. While managing secrets for Terraform it is recommended to keep $tfstate$ files separate from secrets [5].

VI. ***Logging*** (4): The practice of applying logging to track all secret-related operations. Logging is helpful to understand what actors are accessing secrets at what time, providing better visibility in the secret management process [18].

VII. ***Unsealing*** (4) : This practice is unique to Hashicorp Vault [12]. In Hashicorp Vault unsealing refers to the process of decrypting the data encryption key that Hashicorp Vault uses to encrypt all necessary data [18]. We identify two practices to adequately accomplish unsealing:

- *Automation*: For unsealing, Hashicorp Vault uses Shamir's Secret Sharing principle [36], where the master key is split into multiple pieces, each of which must be provided manually by the developer, and later combined to regenerate the master key. The manual process is tedious and error-prone, which can be mitigated using an automated feature provided by Hashicorp Vault called 'auto-unseal'. In this manner, automation can be used when unsealing the master key, potentially mitigating possible errors that can occur due to manual unsealing.
- *Replication*: Practitioners recommend use of multiple vault servers to perform unsealing so that the unsealing process as well as the secrets stored in Hashicorp Vault are not susceptible to single point of failures. Practitioners recommend use of at least two Hashicorp Vault servers, where one server will be the active server, and the other server will be standby. Another practitioner recommends to use at least three Hashicorp Vault servers, where one server will remain active, and two servers will act as standby servers [6].

VIII. ***Secret Rotation*** (3) : This practice refers to the use of rotation, where the value of a secret is periodically retired and replaced with new values. We observe practitioners to recommend rotation for secrets, such as passwords, tokens, and private SSH keys. Rotation can prevent secrets to be exposed to unauthorized users, as mentioned by one practitioner: "*In the case of password reuse or leaks, it is best to regularly re-key the passwords in use to limit exposure*". A lack of secret rotation can be consequential, as it happened for Microsoft Azure, where inadequate secret rotation used for authentication resulted in a 14-hour outage [23].

IX. ***Transport Layer Security*** (2) : This practice refers to the use of transport layer security (TLS) in an end-to-end fashion so that all data transmitted between the IaC compiler and the secret management tool is secured. If a tool, such as Hashicorp Vault is used to manage secrets then practitioners recommend use of end-to-end TLS for both production and development environments [7].

X. ***Search*** (2) : The practice of making the search process efficient for secrets that are stored on secret management tools. We observe the search process of secrets to be dependent on the utilities that secret management tools provide. For example, for Hiera practitioners recommend the use of `confine_to_keys` to efficiently search for a secret.

XI. ***Speedup*** (2): This category refers to the practice of increasing the speed while managing secrets. For example, to speed up the secret management process in Ansible Vault, use of the `cryptography` package is recommended instead of the `PyCrypto` package that is the default package used by Ansible [4].

XII. ***Limiting Authentication*** (1): This practice refers to limiting authentication attempts and duration while using secret management tools. We observe practitioners to recommend limiting the number of attempts to log into a secret management tool. Furthermore, limiting the authentication duration is also recommended so that existing authentication sessions cannot be used by an unauthorized user. Limited authentication can help minimize the total impact of an unauthenticated access.

[4]https://www.cloudsavvyit.com/3902/how-to-use-ansible-vault-to-store-secret-keys/

[5]https://caylent.com/managing-secrets-in-terraform

[6]https://www.youtube.com/watch?v=fOybhcbuxJ0

[7]https://expel.io/blog/production-readiness-hashicorp-vault-kubernetes/

Rater Verification: Our open coding approach is conducted by the first author, which makes the practice derivation process susceptible to bias. We mitigate this bias by allocating another rater, who is the second author of the paper. The second author is a graduate student in the department, with one year of professional experience in software engineering. The second author is provided the set of 38 Internet artifacts, and asked to map each Internet artifact to one or multiple practices identified by the first author. The second author independently conduct the process. Upon completion, the agreement rate between the first and second author is computed using Cohen's Kappa [7]. For the 38 Internet artifacts the Cohen's Kappa between the two raters is 0.8, which is 'substantial' according to Landis and Koch [20].

Mapping of Practices to Tools: We provide a mapping between each identified practice and applicable tools in Table I. The 'Practice' column lists each of the 12 practices, whereas the 'Tool' and 'Language' columns respectively, represent the applicable secret management tool and the applicable IaC language. For example, the practice access control is applicable for all tools and all IaC-related languages. Table I provides a summarized overview of what practices are generic and what practices are tool-specific.

TABLE I: Mapping of Practices and Secret Management Tools for IaC. 'All' indicates generic practices.

Practice	Tool	Language
Access control	All	All
Authentication limits	All	All
Data organization	All	All
Environment separation	Hashicorp Vault	All
Logging	All	All
Password management for Ansible Vault	ansible-vault	Ansible
Prioritized encryption	All	All
Rotation	All	All
Secret Search	Hiera	Puppet
Speedup	ansible-vault	Ansible
TLS	All	All
Unsealing	Hashicorp Vault	All

V. Discussion

We discuss the implications and limitations of our paper in this section:

Implications for Practitioners: Our derived practices can be useful for practitioners when managing secrets for IaC scripts. For example, our derivation shows how secrets should be organized for Puppet, if Puppet Hiera is used to manage secrets for Puppet development. Inadequate application of secret management can lead undesirable consequences, which can be mitigated using our derived list of practices. Certain practices namely, logging, secret rotation, and use of transport layer security are generic practices, which highlight the applicability of generic security best practices to be applicable for secret management tools for IaC as well.

Implications for Toolsmiths: Our findings lay the groundwork for toolsmiths on how to design and develop tools that will make secret management easier for practitioners. For example, as discussed in Section IV, establishing a naming convention is a recommended practice to manage secrets. Toolsmiths can develop tools that can automatically rename variables in IaC scripts that are used to store secrets. Furthermore, with sophisticated program analysis that tracks the flow of secrets across a Puppet module, toolsmiths can develop tools that automatically construct a data hierarchy for Hiera where commonly shared secrets are placed in the low-level.

Implications for Researchers: Our findings can be leveraged to conduct further research in the domain of IaC security. Researchers can investigate to what extent the listed practices in Section IV are adopted in open source and proprietary IaC scripts. Based on prevalence of secrets in IaC scripts, we conjecture that systematic use of secret management for IaC is not commonplace. If empirical research substantiates our conjecture, then researchers can investigate factors that contribute to the lack of use of secret management. Researchers can investigate to what extent our identified practices for IaC are also applicable for other related domains, such as automated container orchestration. Furthermore, empirical research that substantiates the effectiveness of automated secret management tools for secure development of IaC scripts, could also be of interest.

Limitations: Our findings are susceptible to external validity as the reported practices are limited to the Internet artifacts that we used in our paper and may not generalize for another set of Internet artifacts. We mitigate this limitation by systematically collecting and filtering Internet artifacts using 6 search strings. The derivation of practices are also biased by rater knowledge: the first author may have missed practices listed in the set of 38 Internet artifacts. We mitigate this limitation by using assigning another rater who independently inspected the set of 38 Internet artifacts. We observe 'substantial' agreement between the two raters for the set of 38 Internet artifacts.

VI. Conclusion

IaC scripts automates the process of provisioning computing infrastructure at scale using a dedicated programming language. Despite yielding benefits for IT organizations, IaC scripts contain secrets, such as hard-coded passwords. The prevalence of secrets in IaC scripts underlines the importance of adequately managing secrets in IaC scripts. A list of practices related to secret management can aid practitioners in managing secrets, deterring them from hard-coding secrets into IaC scripts. We have conducted a grey literature review with 38 Internet artifacts to identify practices for secret management in IaC. From our analysis, we have identified 12 practices for secret management related to IaC. Our identified practices include tool-specific practices, e.g., unsealing for Hashicorp Vault, as well as, tool-agnostic practices, such as access control. The most frequently mentioned practice is data organization. Our findings also reveal that for secret management in IaC, practitioners can use language-specific tools, e.g., Hiera for Puppet, as well as tools, such as Hashicorp Vault that work with all IaC languages. Our findings lay the

groundwork for future research related to secure development of IaC scripts.

ACKNOWLEDGMENT

We thank the PASER group at Tennessee Tech University for their valuable feedback. This research was partially funded by the U.S. National Science Foundation (NSF) award # 2026869.

REFERENCES

[1] Raphael Campardou, "Ansible (Real Life) Good Practices," https://reinteractive.com/posts/167-ansible-real-life-good-practices, 2014, [Online; accessed 09-May-2021].

[2] 4hathacker, "Ansible Vault - Lets encrypt sensitive data while automation," https://www.4hathacker.in/2018/01/ansible-vault-lets-encrypt-sensitive.html?m=1, 2018, [Online; accessed 21-April-2021].

[3] Ansible, "Ansible Documentation," https://docs.ansible.com/, 2021, [Online; accessed 28-May-2021].

[4] ——, "Ansible Vault — Documentation," https://acozine.github.io/html/user_guide/vault.html, 2021, [Online; accessed 21-May-2021].

[5] Ansible, "Swisscom Automates IT Management WITH RedHat Ansible Tower," https://www.ansible.com/hubfs/pdfs/RH-Ansible-Tower-swisscom-case-study.pdf?hsLang=en-us, 2021, [Online; accessed 13-Feb-2021].

[6] Chef, "About Chef Workstation," https://docs.chef.io/workstation/, 2021, [Online; accessed 26-May-2021].

[7] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960. [Online]. Available: http://dx.doi.org/10.1177/001316446002000104

[8] V. Garousi, M. Felderer, M. V. Mäntylä, and A. Rainer, *Benefitting from the Grey Literature in Software Engineering Research*. Cham: Springer International Publishing, 2020, pp. 385–413. [Online]. Available: https://doi.org/10.1007/978-3-030-32489-6_14

[9] Gary Larizza, "Puppet Workflows 4: Using Hiera in Anger," http://garylarizza.com/blog/2014/10/24/puppet-workflows-4-using-hiera-in-anger/, 2021, [Online; accessed 12-May-2021].

[10] O. Hanappi, W. Hummer, and S. Dustdar, "Asserting reliable convergence for configuration management scripts," *SIGPLAN Not.*, vol. 51, no. 10, pp. 328–343, Oct. 2016. [Online]. Available: http://doi.acm.org/10.1145/3022671.2984000

[11] M. M. Hasan, F. A. Bhuiyan, and A. Rahman, "Testing practices for infrastructure as code," in *Proceedings of the 1st ACM SIGSOFT International Workshop on Languages and Tools for Next-Generation Testing*, ser. LANGETI 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 7–12. [Online]. Available: https://doi.org/10.1145/3416504.3424334

[12] Hashicorp Vault, "Hashicorp Vault," https://www.vaultproject.io/, 2021, [Online; accessed 23-May-2021].

[13] S. Hopewell, M. Clarke, and S. Mallett, "Grey literature and systematic reviews," *Publication bias in meta-analysis: Prevention, assessment and adjustments*, pp. 49–72, 2005.

[14] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation*, 1st ed. Addison-Wesley Professional, 2010.

[15] W. Hummer, F. Rosenberg, F. Oliveira, and T. Eilam, "Automated testing of chef automation scripts," in *Proceedings Demo & Poster Track of ACM/IFIP/USENIX International Middleware Conference*, 2013, pp. 1–2.

[16] W. L. Hursch and C. V. Lopes, "Separation of concerns," 1995.

[17] K. Ikeshita, F. Ishikawa, and S. Honiden, "Test suite reduction in idempotence testing of infrastructure as code," in *International Conference on Tests and Proofs*. Springer, 2017, pp. 98–115.

[18] Jon Benson, "5 best practices for secrets management," https://www.hashicorp.com/resources/5-best-practices-for-secrets-management, 2021, [Online; accessed 03-May-2021].

[19] I. Kumara, M. Garriga, A. U. Romeu, D. Di Nucci, F. Palomba, D. A. Tamburri, and W.-J. van den Heuvel, "The do's and don'ts of infrastructure code: A systematic gray literature review," *Information and Software Technology*, vol. 137, p. 106593, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584921000720

[20] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *Biometrics*, vol. 33, no. 1, pp. 159–174, 1977. [Online]. Available: http://www.jstor.org/stable/2529310

[21] J. Loeliger and M. McCullough, *Version Control with Git: Powerful tools and techniques for collaborative software development*. " O'Reilly Media, Inc.", 2012.

[22] R. Mao, H. Zhang, Q. Dai, H. Huang, G. Rong, H. Shen, L. Chen, and K. Lu, "Preliminary findings about devsecops from grey literature," in *2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 2020, pp. 450–457.

[23] Mary Jo Foley, "Microsoft's latest cloud authentication outage: What went wrong," https://www.zdnet.com/article/microsofts-latest-cloud-authentication-outage-what-went-wrong/, 2021, [Online; accessed 13-May-2021].

[24] MITRE, "CWE-Common Weakness Enumeration," https://cwe.mitre.org/index.html, 2021, [Online; accessed 02-May-2021].

[25] Puppet, "Product documentation," https://puppet.com/docs/, 2021, [Online; accessed 27-May-2021].

[26] ——, "Splunk Case Study," https://puppet.com/resources/customer-story/splunk-case-study/, 2021, [Online; accessed 14-Feb-2021].

[27] A. Rahman, E. Farhana, C. Parnin, and L. Williams, "Gang of eight: A defect taxonomy for infrastructure as code scripts," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ser. ICSE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 752–764. [Online]. Available: https://doi.org/10.1145/3377811.3380409

[28] A. Rahman, E. Farhana, and L. Williams, "The 'as code'activities: development anti-patterns for infrastructure as code," *Empirical Software Engineering*, vol. 25, no. 5, pp. 3430–3467, 2020.

[29] A. Rahman, C. Parnin, and L. Williams, "The seven sins: security smells in infrastructure as code scripts," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 164–175.

[30] A. Rahman, A. Partho, P. Morrison, and L. Williams, "What questions do programmers ask about configuration as code?" in *Proceedings of the 4th International Workshop on Rapid Continuous Software Engineering*, ser. RCoSE '18. New York, NY, USA: ACM, 2018, pp. 16–22. [Online]. Available: http://doi.acm.org/10.1145/3194760.3194769

[31] A. Rahman, M. R. Rahman, C. Parnin, and L. Williams, "Security smells in ansible and chef scripts: A replication study," *ACM Trans. Softw. Eng. Methodol.*, vol. 30, no. 1, Jan. 2021. [Online]. Available: https://doi.org/10.1145/3408897

[32] A. A. U. Rahman and L. Williams, "Different kind of smells: Security smells in infrastructure as code scripts," *IEEE Security Privacy*, pp. 2–10, 2021.

[33] J. Saldaña, *The coding manual for qualitative researchers*. Sage, 2015.

[34] J. Schwarz, "Code Smell Detection in Infrastructure as Code," https://www.swc.rwth-aachen.de/thesis/code-smell-detection-infrastructure-code/, 2017, [Online; accessed 02-July-2019].

[35] M. I. Shamim, F. A. Bhuiyan, and A. Rahman, "Xi commandments of kubernetes security: A systematization of knowledge related to kubernetes security practices," in *2020 IEEE Secure Development (SecDev)*. Los Alamitos, CA, USA: IEEE Computer Society, sep 2020, pp. 58–64. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SecDev45635.2020.00025

[36] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, p. 612–613, Nov. 1979. [Online]. Available: https://doi.org/10.1145/359168.359176

[37] Terraform, "Terraform Language Documentation," https://www.terraform.io/docs/language/state/index.html, 2021, [Online; accessed 25-May-2021].

[38] A. A. Ur Rahman and L. Williams, "Software security in devops: Synthesizing practitioners' perceptions and practices," in *Proceedings of the International Workshop on Continuous Software Evolution and Delivery*, ser. CSED '16. New York, NY, USA: ACM, 2016, pp. 70–76. [Online]. Available: http://doi.acm.org/10.1145/2896941.2896946

[39] E. van der Bent, J. Hage, J. Visser, and G. Gousios, "How good is your puppet? an empirically defined and validated quality model for puppet," in *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, March 2018, pp. 164–174.

[40] Vincenzo Pii, "Where to put ansible-vault password," https://devops.stackexchange.com/questions/3282/, 2018, [Online; accessed 29-May-2021].