Infrastructure as Compromise: Abusing Residual Trust in Infrastructure as Code Tools

Ruining Yang
Stony Brook University
New York, USA
ruiniyang@cs.stonybrook.edu

Narong Chaiwut
Stony Brook University
New York, USA
nchaiwut@cs.stonybrook.edu

Nick Nikiforakis
Stony Brook University
New York, USA
nick@cs.stonybrook.edu

Abstract

Infrastructure as Code (IaC) has transformed the way developers deploy and manage their infrastructure, enabling automated, reproducible, and version-controlled builds. At the same time, developer errors in IaC configurations can result in thousands of identically-vulnerable servers.

In this paper, we study the so-far ignored problem of residual trust in IaC tools. IaC configurations (also known as playbooks) can refer to remote code and data that will be fetched upon execution and be incorporated in the resulting infrastructure. As such, attackers can perform supply-chain attacks against IaC environments by taking over the third-party resources that are used by playbooks during the build process. To understand the attack surface of this threat, we focus on Ansible and Puppet and quantify all the ways that developers can introduce remote references in their code. We then perform a large-scale study of publicly-available IaC playbooks on GitHub to characterize the ways through which developers actually introduce remote references in their IaC scripts. By analyzing IaC playbooks in 247,131 repos, we identify 463 expired domains that can be immediately registered by attackers and 10,400 instances of misconfigurations in the addressed remote hosts. Our analysis also identified evidence of typosquatting errors across 30 Ansible repositories and 16 Puppet repositories, as well as a novel attack vector involving placeholder domain names. In recognition of the magnitude of the identified problems, we propose DependoScope, an extension for a popular code editor that steers developers away from erroneous remote references.

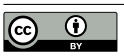
CCS Concepts

• **Security and privacy → Web application security**.

Keywords

Web applications, Infrastructure, Supply-chain attacks

ACM Reference Format:
Ruining Yang, Narong Chaiwut, and Nick Nikiforakis. 2026. Infrastructure as Compromise: Abusing Residual Trust in Infrastructure as Code Tools. In *Proceedings of the Sixteenth ACM Conference on Data and Application Security and Privacy (CODASPY '26), June 23–25, 2026, Frankfurt am Main, Germany.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3800506.3803500



CODASPY '26, Frankfurt am Main, Germany

ACM ISBN 979-8-4007-2562-3/2026/06
https://doi.org/10.1145/3800506.3803500

1 Introduction

Deploying and managing servers has become an increasingly complex task. The rise of diverse technologies, applications, and platforms has led to a surge in the number of servers that need to be managed and maintained. Moreover, virtualization and cloud computing allows one physical server to be turned into tens of virtual servers, further complicating the task of managing this ever-expanding infrastructure. Manual administration of these servers often leads to inefficiencies and errors, which can cause significant problems such as downtime, data losses, and security vulnerabilities. Specifically related to security vulnerabilities, numerous examples of high-profile break-ins involved the abuse of a forgotten server [16, 46], a piece of outdated software [39, 48], or a domain name that was forgotten and was left to expire [11, 27]. All of these issues arise because of bad inventory management and manual administration processes which do not scale to the levels required by large institutes and companies managing thousands of servers and software-related assets.

This need for better inventory management and automation is addressed by DevOps technologies where software development (Dev) and IT operations (Ops) are combined. DevOps promotes the continuous integration and continuous delivery of software, improves the process of software development, and monitors the performance of applications and infrastructure. Through these capabilities, DevOps technologies increase deployment frequency, reduce the failure rate of new releases, shorten the lead time between fixes, and improve mean time to recovery.

Within the broader topic of DevOps, Infrastructure as Code (IaC) tools have garnered attention since they tackle the specific pain points regarding the provisioning, deploying, and management of servers. IaC tools enable DevOps engineers to write code that describes not just their desired infrastructure (such as the location and operating system of virtual machines) but also the exact software that must be installed on that infrastructure and its configuration. This code enables the automated, reliable, and repeatable creation of new computing resources and can be version controlled along the rest of the software of that organization. Multiple popular tools exist that meet the demand for these IaC capabilities with Ansible and Puppet being two of the most popular ones.

Since IaC tools offer reliable and repeatable infrastructure and software environments, one properly written IaC configuration can yield thousands of securely configured servers. Yet the corollary of this is that *all* servers produced by a single insecure IaC configuration will be identically vulnerable to attacks. Prior work in IaC security has mostly focused on the detection of less-than-ideal code constructs (known as *code smells*) in configuration scripts [23, 32, 34, 38, 41].

In this paper, we draw attention to the overlooked aspect of *integrity* in IaC tools. IaC code is rarely self-contained since a significant fraction of build scripts rely on third-party servers to pull additional code and data. This includes the automated cloning of GitHub repositories, the pulling of Docker images and software packages from official and unofficial registries, as well as the downloading of software and configurations from arbitrary servers which the DevOps engineers consider trusted. Our key insight is that just because a specific third party is trusted *today*, it does not mean that it can be trusted forever. Domains expire and servers get compromised. As a result, an IaC playbook that once created the desired infrastructure can now create radically different infrastructure without the playbook itself ever changing.

To understand the threat of remote references and residual trust in IaC tools, we first analyze the official documentation of Ansible and Puppet modules, in search of remote-reference support. We discover that 46.9% and 14.8% of all Ansible and Puppet modules respectively, allowed developers to reference remote code and data in their IaC playbooks. We then conduct a large-scale crawl of real-world IaC code that developers have made publicly available on Github. We download 247,131 repos with 3,580,699 IaC playbooks for Ansible and Puppet. By processing each playbook in search of remote references, we find that 31.0% of Ansible and 16.6% of Puppet repos contain at least one remote reference. We extract these remote references and look for outright vulnerabilities (such as expired domain names that attackers can immediately register) as well as misconfigurations (e.g. referenced hosts that are unresponsive). Overall, we find 463 expired domains and 10,400 out of 16,244 unique domains/IP addresses misconfigurations that could be weaponized to hijack IaC playbooks and compromise an organization's infrastructure. Among others, we identify an entirely new threat vector involving IaC developers using custom placeholder domains in their public scripts (such as `myexample.com`) instead of IANA-approved example domains (e.g. `example.com`). Other developers who download and execute these playbooks on their infrastructure *before* making any changes can therefore have their infrastructure hijacked from these custom domain names. We reach out to affected developers and all but one of them agreed with our assessment.

Our main contributions are as follows:

- We draw attention to the issue of integrity and remote references in IaC tools and show how attackers can produce malicious infrastructure by hijacking domains and hosts that were once trusted by developers.
- We quantify the attack surface of IaC tools regarding residual trust by automatically crawling and analyzing their documentation, in search for modules and plugins that accept remote references.
- We conduct a large scale analysis of how real-world IaC developers utilize remote references in their playbooks, analyzing 247,131 publicly available unique repos. We find vulnerabilities in popular and unpopular repos and reach out to the most popular repos affected.
- We show that state-of-the-art static analysis tools cannot detect all remote references and propose DependoScope, a pragmatic countermeasure that can assist IaC developers in choosing secure remote references as they are developing and reviewing their IaC playbooks.

```yaml
- name: Install and configure Apache with custom config
  hosts: all
  become: yes
  tasks:
    - name: Install Apache Package
      ansible.builtin.yum:
        name: httpd
        state: present
    - name: Download custom Apache config
      ansible.builtin.get_url:
        url: https://example.com/httpd.conf
        dest: /etc/httpd/conf/httpd.conf
    - name: Deploy a simple HTML File
      ansible.builtin.copy:
        dest: /var/www/html/index.html
        content: "<h1>Apache is working!</h1>"
    - name: Ensure Apache is running and enabled
      ansible.builtin.systemd:
        name: httpd
        enabled: yes
        state: restarted
```

a) Ansible Script for Apache

```
class { 'apache':
  ensure => present,
}
exec { 'download-httpd-conf':
  command => 'curl -o /etc/httpd/conf/httpd.conf https://example.com/httpd.conf',
  require => Package['httpd'],
}
file { [
      '/etc/httpd/conf/httpd.conf',
      '/var/www/html/index.html']:
  ensure => file,
  content => { '/var/www/html/index.html' => "<h1>Apache is working!</h1>" },
  require => Exec['download-httpd-conf'],
}
service { 'httpd':
  ensure   => running,
  enable   => true,
  subscribe => File['/etc/httpd/conf/httpd.conf', '/var/www/html/index.html'],
}
```

b) Puppet Script for Apache

Figure 1: Comparison of Ansible (a) and Puppet (b) scripts for installing and configuring the Apache web server.

To motivate additional work in this new area of Infrastructure as Code, we will be open-sourcing our data and countermeasure, upon publication of this paper.

2 Background

In this section, we provide an overview of Infrastructure as Code (IaC) and a comparison of Ansible and Puppet. We also describe our threat model showing how attackers can weaponize residual trust to hijack the infrastructure created through these IaC tools.

2.1 Infrastructure as Code (IaC), Ansible, and Puppet

Infrastructure as Code (IaC) automates the management and provisioning of computing infrastructure through machine readable definition files, replacing manual and error prone processes. This paradigm allows infrastructure configurations to be versioned, tested, and systematically deployed, much like application code.

Among the most prominent IaC tools are Ansible and Puppet. While both serve to automate system configuration, their architectures present different trust models. Ansible operates in an agentless fashion, typically executing tasks on remote nodes over SSH. This approach relies on the trust established via SSH credentials. In contrast, Puppet utilizes a master agent model, where a central master server compiles and distributes configuration "catalogs" to agent nodes, which then enforce the desired state. This model centralizes configuration logic but requires explicit trust between the agent and the master. Both tools define configurations in high level languages, using YAML for Ansible and a custom Domain Specific Language (DSL) for Puppet.

2.2 Remote References and Threat Model

A core promise of IaC tools is that they can be used to create identically configured environments by ensuring that the resulting

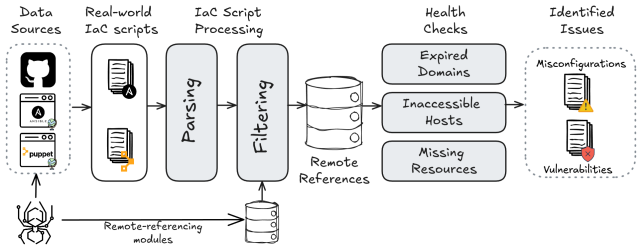


Figure 2: High-level view of our analysis pipeline for IaC configuration scripts written in Ansible and Puppet.

remote infrastructure complies with the model described in an IaC configuration script.

In this paper, we draw attention to the fact that these configuration scripts do not have to be self-contained. That is, they can include one or more remote references which point to code and data that will be used during the provisioning of new infrastructure. These remote references can include, e.g., remote GitHub repositories that need to be cloned on the newly-provisioned servers, Docker images to be pulled and executed, and specific configurations for newly installed software. Our previously highlighted Ansible/Puppet playbooks both pull a remote configuration file for the newly deployed web server (Line 11 for Ansible, and Line 6 for Puppet in Figure 1).

These references point to arbitrary remote servers which DevOps developers trust at the time when they are writing their IaC programs. Our key insight is that just because a specific third party is trusted *today*, it does not mean that it can be trusted forever. Domain expirations [15, 19, 21, 43, 51], server compromises [8, 9, 49], stale DNS records [7, 26], and network-level attacks [6, 12, 13, 50] can be abused by attackers to take control of the referenced hosts. In turn, this can cause IaC tools to produce *radically* different servers between runs, without these configuration scripts ever changing. Just because an IaC configuration script produces the desired output today, this does not mean that it will continue to do so in the weeks, months, and years *after* it was written.

The threat model that we therefore explore is where attackers analyze IaC configuration scripts to locate hijackable remote references. Depending on how exactly a remote reference is used in an IaC configuration script, the damage from such a hijacked domain name could range from a Denial of Service attacks all the way to a complete takeover of newly-provisioned computing infrastructure.

It is worth pointing out that even experts in DevOps (e.g. Microsoft) underestimate the threat of integrity in IaC, making statements such as the following in their educational resources [17]: "Just as the same source code always generates the same binary, an IaC model generates the same environment every time it deploys." The goal of this paper is to show that this statement only holds true if the IaC model is entirely self-contained and does not depend on any remote code/data.

3 System Design

As we established in the previous section, developers of IaC configuration scripts are able to include remote references in their code which can be ultimately weaponized by attackers to hijack the provisioned infrastructure. At the same time, the extent to which developers use remote references in real-world code is not known. Similarly, even though IaC languages offer integrity mechanisms for some of their remote references, it is unclear not just how often these mechanisms are available but also whether developers take advantage of them.

In this section, we first establish the overall the attack surface of Ansible and Puppet by crawling their documentation to identify all the ways that remote references *can* can be introduced. We then characterize how developers *do* use them in practice by curating a large dataset of real world IaC scripts from GitHub, and describe how we evaluate the "health" of all identified remote references to assess their hijackability.

3.1 Identifying Remote References in IaC Documentation

In this section, we describe the process we followed to systematically discover how remote references *can* be included in Ansible and Puppet scripts. By identifying all the tasks and modules that support such references, we not only establish the overall attack surface of these tools related to remote dependencies, but also obtain the specific module and parameter names of these tasks which can be used as filters when parsing real-world IaC scripts (Section 3.2).

Ansible documentation. In November 2023, we crawled the official Ansible documentation page to obtain the names of all available Ansible modules. By automatically visiting the web page of each module, our crawlers extracted the name, type, and description of each parameter. In addition, we downloaded all the usage examples provided on the official website for these modules whenever they were available. In total, we collected information on 8,187 modules, which included details on 177,201 parameters.

To identify as many parameters related to remote references as possible, we followed a snowball sampling approach. We start by looking for module examples that include values starting with `http(s)://`. We make note of the parameter names corresponding to these values and then search all crawled parameters for the same names. Whenever we discover composite parameter names (e.g. `cpm_url`) we split those names into their components and repeat the process. We do multiple passes of parameter matching across the entire crawled documentation, employing filters as necessary to remove sources of false positives. Ultimately, we were able to build a list of 7,087 parameter names (including parameter names such as `url`, `uri`, `host`, and `server`) that all indicate the use of a remote reference in an Ansible module.

Puppet documentation. In our analysis of Puppet, we began by manually examining the standard core resource types that can reference remote locations, as documented on the official Puppet documentation [28]. We observed that the standard module lacks explicit attributes designed specifically for this purpose. Instead, Puppet provides the flexibility to use OS-level commands to achieve similar functionality. For example, the `exec` resource type can utilize the `command` attribute to execute OS-standard tools for file retrieval (see Figure 1). In this case, the `exec` resource type employs the `command` attribute to invoke the `curl` application, which is used to reference remote files.

To enable more complicated use cases, Puppet allows developers to publish their own Puppet scripts as modules through Puppet Forge [29], their official module repository. For instance, the `nginx` module manages NGINX configurations and introduces specific attributes that enable users to explicitly define remote references.

Overall, given the laconic nature of Puppet's syntax and the popularity of their Puppet Forge platform, our crawler considers both sources of documentation for discovering how developers can introduce remote references in their Puppet scripts.

3.2 GitHub Crawling and Parsing

Like prior analyses of IaC issues [30, 32, 37], we obtain our real-world IaC configuration scripts from GitHub. By searching for the keywords "ansible" and "puppet", we downloaded 220,531 Ansible repositories published between March 2012 and December 2023, as well as 26,600 Puppet repositories published between January 2010 and April 2024. As we can already see from these statistics, even though Ansible is a more recent IaC tool than Puppet, it has enjoyed significantly larger adoption.

For the Ansible part, since not all files in the downloaded repositories are Ansible playbooks, we first focus only on files ending with `.yaml` or `.yml`. We then categorize Ansible files into two types: playbooks and variable files. We use Ansible's built-in syntax-checking tools to determine whether each analyzed file is an Ansible playbook. Since Puppet uses its own file extension, we simply focus on files written in Ruby that have the `.pp` file extension.

Ansible's variable files contain data in the form of key-value pairs that can be used in playbooks or tasks. We refer to the Ansible official documentation [4] for a list of paths that may contain variable files (e.g. group_vars/*, host_vars/*). As long as these files are found in the appropriate paths, we treat them as variable files. In contrast to Ansible, Puppet uses the => syntax to separate attributes and values (e.g., `remote => https://www.example.com`).

After identifying the Ansible configuration scripts, we parsed each file to locate the URLs, domain names, and IP addresses involved in networking. Since some configuration scripts use variables (defined in a separate variable file) instead of hard-coding values in their playbooks, we use the prototype of the tool GASEL, developed by Opdebeeck et al. [23]. GASEL employs a program dependence graph to analyze and determine variable usage in playbooks, which we use to scan each repository to find the names of all variables and their corresponding values. In this way, we can replace variable names in YAML files with actual values. For Puppet, we used the parser developed by Saavedra et al. [38] to parse all identified Puppet scripts. We then applied a depth-first search to extract the attribute names that contain remote references.

When resolving IP addresses, we filter out private IP addresses and multicast IP addresses. For domain names found, we use the full list of top-level domains (TLDs) to identify as many valid domain names as possible. We also specifically filtered out certain entries that are more likely to represent files rather than actual domain names, such as filenames ending in `.py`, `.sh`, or `.md`. While these happen to be valid TLDs, their inclusion would incur a prohibitively large number of false positives.

In the case of Ansible's script analysis, we implemented an additional filtering step to minimize false positives given the large number of scripts involved. Parameters containing keywords such as "tag" or "version", as well as values that match our IP address regular expression but have a first-octet value of less than five (e.g., 2.1.1.3), were flagged as likely to be versioning/tagging metadata and subsequently excluded from further consideration.

Using Ansible's official documentation (crawled and catalogued as described in Section 3.1), we determined that tasks containing keywords such as "absent," "disabled," or "deleted" generally do not trigger external network operations involving remote references, so these tasks were systematically filtered out. In addition, any remote references that did not include a protocol scheme were removed, unless we could match the tasks in which they were used in our crawled Ansible documentation.

3.3 Health of Remote References

Through the previous two steps we are able to quantify not just all the ways that IaC developers can introduce remote references in their configuration scripts, but also all the actual remote hosts that they have chosen to trust, according to their publicly accessible GitHub code.

Given the discovered remote references, we can therefore assess the health of these references and to what extent attackers could compromise IaC configuration scripts by hijacking the remote references on which they depend. As we noted in Section 2.2, there are numerous ways to hijack remote infrastructure. Yet not all ways can be ethically quantified in a generic context. To this end, in this paper, we focus on remote references that are clearly misconfigured (pointing to high likelihood that they can be hijacked) or are readily exploitable because the domains that they reference were left to expire and are available for registration. Given our focus on just these types of healthchecks, we argue that all of our findings in this paper can be considered as lower bounds of the true vulnerability of IaC scripts to supply-chain attacks via hijacked remote references.

Expired domain names. Given the domain of a remote reference extracted from a Puppet or Ansible playbook, we attempt to retrieve its DNS records using Google's open resolvers. In addition to A and AAAA records, we query a domain's DNS servers for multiple types of DNS records including SOA, TXT, and CNAME. If any one of these records is returned successfully, we exclude that domain from further analysis.

The domains that are left after our first filtering are then queried against three popular domain-name registrars namely Dynadot [3], GoDaddy [2], and Namecheap [1]. We use their publicly available APIs to check whether a domain name is available for registration. Using three registrars (as opposed to just one) provides us with a more accurate picture of the expired-domains landscape since not all registrars support the same TLDs.

Any domain that is marked as available for registration by one or more registrars is one that attackers can register and immediately seed the affected IaC configuration scripts with malicious code and data. In Section 4, we present a detailed analysis of how many domains we discovered and common patterns behind them.

Missing resources and inaccessible servers. Prior research has shown that the ease of provisioning infrastructure on public clouds often results in stale DNS records and links [7, 26]. For example, the

Table 1: Summary of remote references from the documentation and GitHub analysis for both Ansible and Puppet.

Details	Ansible	Puppet
Document crawling		
No. of modules	8,187 (100%)	7,500 (100%)
No. of scripts with at least one remote reference	3,839 (46.9%)	1,109 (14.8%)
GitHub repositories		
No. of repos	220,531 (100%)	26,600 (100%)
No. of repos with at least one remote reference	68,424 (31.0%)	4,413 (16.6%)
No. of scripts	3,174,234 (100%)	406,465 (100%)
No. of scripts with at least one remote reference	194,511 (6.1%)	18,641 (4.6%)

owners of `example.com` can add a new DNS records to their zone (`test.example.com`) and point that record to a public IP address (e.g. `203.0.113.2`) of computing infrastructure (e.g. a virtual machine) on a public cloud. At a later time, when they are done with development, they can let go of that infrastructure while forgetting to remove the corresponding DNS record from their zone files. In this way, there remains a dangling/stale DNS record `test.example.com` pointing to the `203.0.113.2` IP address that has now returned to the pool of available addresses of that public cloud. Attackers can then request IP addresses from that public cloud, until they are given the one corresponding to `test.example.com`. Once they get access to that IP address, they can abuse that access in a variety of ways including the issuance of TLS certificates for `test.example.com` as well as potential bypasses of the Same-Origin Policy in a user's browser in the context of an XSS attack.

In the context of this paper, we focus on identifying hosts that are seemingly misconfigured, either because they are not hosting the resource at the location that an IaC script expects them to, or because they are altogether inaccessible. Given the popularity of web protocols we first try to connect to a host over HTTP(S). If a specific filepath was used in a playbook (e.g. https://example.com/data.bin) we request the same resource from the remote host. We make note of any requests that result in an HTTP(S) 4XX (client-side error) and 5XX (server-side error) status codes. When the protocol of a remote reference is not available in the IaC configuration script, we rely upon a network scanner that checks the top 1,000 most popular network ports. We conservatively mark a host as inaccessible only when none of these ports are open.

4 Results

In this section, we present our findings on remote reference parameters and attributes, drawing from the official documentation of Ansible and Puppet. We then extend our analysis to GitHub repositories, where we examine multiple aspects of the identified remote references, including their number, expired and re-registerable domains, and inaccessible remote hosts. Furthermore, we conduct a dynamic analysis of these re-registerable domains, which could be potentially abused to hijack newly-provisioned infrastructure. Finally, we compare our findings with a state-of-the-art static analyzer for IaC scripts [38] to understand why many of our discovered remote references could not be detected by prior work.

4.1 Identified Support for Remote References

We crawled the official documentation sites of Ansible and Puppet on November 2023 and January 2024 respectively. As described in Section 3.1, our crawlers searched for all the different ways that developers could introduce remote references to their IaC configuration scripts.

The top portion of Table 1 summarizes our findings from that crawling process. Out of 8,187 Ansible modules, we discovered that 3,839 modules—or 46.9% of the total—contain at least one parameter that can refer to remote resources. From the 7,500 Puppet modules that were available on Puppet Forge, our crawlers identified 1,109 (14.8%) providing support for one or more remote references. While both IaC languages extensively support remote references, we observe that Ansible's support is significantly more widespread with almost half of its modules being able to reference remote resources.

Further investigation into the names of parameters and attributes in Ansible and Puppet that are likely to reference remote locations revealed common names such as `scopes`, `hostname`, `gateway` and `baseurl`. As we describe throughout this paper, a single remotely-referenced resource that falls in the hands of attackers can be abused to potentially compromise all provisioned infrastructure from the affected IaC scripts.

4.2 GitHub Repositories Analyzed

The bottom portion of Table 1 lists the number of Ansible and Puppet repositories we were able to identify and download from GitHub. Even though we keep track of the popularity statistics of each repo (e.g. the number of stars) we decided not to use any popularity-based filters. We reasoned that even relatively unpopular snippets of IaC code can land in highly-critical environments (either directly or via copy-pasting) which cannot be adequately represented just based on GitHub statistics.

For Ansible, we were able to download 220,531 publicly-accessible repositories, identifying 3,174,234 Ansible script files for analysis. After the process of filtering (as described in Section 3.2), our findings revealed that 194,511 Ansible scripts contain at least one remote reference, representing 6.1% of the total Ansible scripts.

Similarly, we downloaded 26,600 publicly-accessible Puppet repos, hosting a total of 406,465 Puppet script files. Upon analysis, we found that 18,641 files contained at least one remote reference, representing 4.6% of the total Puppet scripts. Even though all Puppet statistics are an order of magnitude smaller than Ansible, we observe that the relative reliance on remote references is similar across both IaC tools.

4.3 Dynamic Remote References Analysis

Prior work in the space of IaC security has proposed static analyzers that can parse the configuration scripts written by developers in search for vulnerabilities and "code smells" (code constructs that are indicative of vulnerabilities) [32, 34, 37]. Yet no static analysis can reason about the status of domain names and remote hosts. As described in Section 3, our main contribution lies in identifying the problem of remote references in IaC tools and dynamically assessing their health status.

4.3.1 DNS analysis. From the 194,511 Ansible scripts that included at least one remote reference, we extracted a total of 387,101 remote references (consisting of full URLs, domain names, and hardcoded IP addresses). Similarly, we extracted 43,110 remote-references from

Table 2: Results of DNS and IP address analysis for all parameters of Ansible and attributes of Puppet

Details	Ansible	Puppet
DNS Analysis		
No. of remote references with DNS records	338,113 (87.3%)	38,610 (89.6%)
No. of remote references with no DNS records	3,013 (0.8%)	1,303 (3.0%)
No. of IP addresses	45,975 (11.9%)	3,197 (7.4%)
Total	387,101 (100%)	43,110 (100%)

Table 3: Types of attacks for domain re-registration

Categories of domain attacking	Ansible	Puppet
Remote Code Execution (RCE)	81 (47.1%)	85 (72.0%)
Data Theft	55 (32.0%)	17 (14.4%)
Denial of Service (DoS)	35 (20.3%)	16 (13.6%)
Total	172 (100%)	118 (100%)

18,641 remote-reference-including Puppet scripts. In total, we identified 7,622 unique domains and 8,622 unique IP addresses across both Ansible and Puppet scripts.

We focused on the remote-references that included domain names and collected DNS resolution data for a number of different resource records. If a domain name had at least one record of any type, we considered it registered and thus not vulnerable for domain re-registration attacks. Even though the vast majority of domain-utilizing remote references did respond to DNS requests, we still discovered that thousands (as shown in Table 2) did not. For example, the domain names involved in 3,013 Ansible remote references did not respond with any DNS records.

By isolating the domain names from the failed resolutions, removing duplicates, and querying different registrars (as described in Section 3.3), we ultimately discovered that 345 Ansible domain names and 118 Puppet domains were available for registration, at the time of our experiments.

To analyze the potential impact of attackers re-registering these domain names, we randomly sampled 290 domains available (roughly half of the 345 Ansible domains and all 118 Puppet domains) for registration and traced them back to their repos and specific IaC modules/tasks in which they were used. By observing the exact context in which these domain names are used and consulting official IaC documentation for Ansible and Puppet, we were able to reason about the capabilities that attackers could gain via these domain names. Table 3 shows the results of this analysis.

We identified 81 (47.1%) Ansible and 85 (72.0%) Puppet domains as potentially able to lead to Remote Code Execution (RCE) attacks if re-registered, since an attacker could distribute malicious code via these URLs. For example, the domain `apachegroup.org` from the `infrastructure-puppet` repository is used to set up Apache for Puppet.

Additionally, we identified 55 (32.0%) Ansible and 17 (14.4%) Puppet domains as those where an attacker could potentially steal sensitive data. For instance, `sqrawler.com` from the `sysadmin-puppetmasterfiles` repository is used for database configuration. Moreover, 35 (20.3%) Ansible and 16 (13.6%) Puppet domains were identified as those that could lead to DoS attacks if exploited. An example is `wartenserver.net` from the `openvpnbox` repository, which is used to set up a VPN server.

Table 4: Results of liveliness analysis with deduplication for Ansible and Puppet remote references. Top half of this table analyzes references that were just domain names. Bottom half analyzes references that were full URLs.

Details	Ansible	Puppet
Domains		
Categories		
Unique domains with HTTP	17,128 (10.7%)	4,346 (20.1%)
Unique domains with HTTPS	143,356 (89.3%)	17,291 (79.9%)
Responses		
Unique domains with 2xx responses	132,866 (82.8%)	17,285 (79.9%)
Unique domains with 4xx responses	5,037 (3.1%)	442 (2.0%)
Unique domains with 5xx responses	327 (0.2%)	176 (0.8%)
Unique domains with connection/timeout errors	21,646 (13.5%)	2,499 (11.5%)
Unique domains with TLS errors	534 (0.3%)	1,235 (5.7%)
Unique domains with other codes	74 (0.0%)	0 (0.0%)
URLs		
Categories		
Unique URLs with HTTP	47,639 (19.3%)	10,001 (34.0%)
Unique URLs with HTTPS	199,250 (80.7%)	19,436 (66.0%)
Responses		
Unique URLs with 2xx responses	173,270 (70.2%)	11,792 (40.1%)
Unique URLs with 4xx responses	34,191 (13.8%)	5,243 (17.8%)
Unique URLs with 5xx responses	1,367 (0.6%)	153 (0.5%)
Unique URLs with connection/timeout errors	36,858 (14.9%)	11,086 (37.7%)
Unique URLs with TLS errors	1,108 (0.4%)	1,161 (3.9%)
Unique URLs with other codes	95 (0.0%)	2 (0.0%)

The most popular Ansible repository that hosted code vulnerable to this attack has 1,108 stars and 451 forks, whereas the most popular Puppet has 30 stars and 30 forks. These results show not just the popularity of repos affected by expired remote references but all the other repos that could be potentially affected by the same issues.

4.3.2 Accessibility checking. In the previous section, the identified issues were outright vulnerabilities since attackers can just register the expired domains and start serving malicious code and data. However, just because a domain-name resolves to an IP address, this does not necessarily mean that there is service listening at the network port where the IaC client is expecting or even that the host itself is "alive." In this section, we analyze this latter set of problems and identify misconfigurations many of which are presumably hiding actual vulnerabilities (such as stale and hijackable DNS records pointing to public clouds). Table 4 lists the categories and responses of these misconfiguration-related health checks. We check the remote references in two formats: domain names, where we extract and analyze the TLD+1 or include subdomains when present, and full URLs, where we use the entire remote reference for analysis. This ensures comprehensive coverage of the liveliness of remote references.

In our analysis, we considered that the same developer might include multiple entries with the same domain or identical full URLs within the same GitHub repository. To avoid repetition which could artificially inflate our findings, we filtered out duplicate content based on repositories. For example, assume that Alice has `repo1` and `repo2`, and Bob has `repo3`, where all three repos have multiple remote references to `example.com` and `mail.example.com`. In our analysis, these domains will only be counted once per repository, regardless of how many times they appear, thereby resulting in a total of 6 counts (2 for each repository). The same approach is applied to full URLs—if a full URL appears multiple times within a single repository, we only count it once.

First, we classified the protocols used in the remote references by extracting the "scheme" portion from them. Regarding just domain names, in Ansible, we found that 143,356 (89.3%) remote references used HTTPS, while 17,128 (10.7%) still relied on HTTP. For Puppet, the situation was similar but slightly less secure: 17,291 (79.9%) references used HTTPS, while 4,346 (20.1%) did not. The threat of plaintext HTTP communications is well known and hence the fact that between 10.7% and 20.1% of remote-referencing IaC scripts use them is cause for concern. In terms of remote references that were full URLs (bottom part of Table 4), 199,250 (80.7%) Ansible references and 19,436 (66.0%) Puppet references used HTTPS. In contrast, 47,639 Ansible references and 10,001 Puppet references still used HTTP. The percentage of HTTP usage in Ansible and Puppet for URLs was 19.3% and 34.0%, respectively, leaving them straightforwardly vulnerable to man-in-the-middle attacks.

Second, to assess the accessibility of remote references, we sent requests in both domain and URL formats and analyzed the responses for Ansible and Puppet references. We found that the majority of responses were `2xx` status codes for both domain and URL formats in Ansible and Puppet, indicating successful access. However, there were errors such as `4xx`, where the remote references were not found: 5,037 (3.1%) and 34,191 (13.8%) in domain and URL formats for Ansible, and 442 (2.0%) and 5,243 (17.8%) in domain and URL formats for Puppet. Additionally, there were other errors, such as `5xx` errors, which indicate server-side issues. TLS errors, which occur when establishing a secure connection, were also observed. Moreover, we also identified several other response codes, such as `300`, `301`, and `999`, which reflect various states of redirection and unusual client requests, respectively. Interestingly, connection or timeout errors were particularly noteworthy, as they were the most frequent error responses in both domain and URL formats for Ansible and Puppet.

4.3.3 Inaccessible hosts. Table 4 also shows a non-negligible number of connection errors associated with specific remote IaC references. For example, at the level of domain names, we were unable to connect to 21,646 domains used in Ansible scripts. To differentiate between protocol-level errors vs. general host errors, we used a popular network scanner and scanned the 1K most popular ports on these hosts. If any single one of these ports was responsive, we conservatively mark the host as accessible for this test. The result of this process was that 5,002 unique hosts referenced in Ansible scripts and 710 referenced in Puppet ones appear to be complete inaccessible, thereby increasing the likelihood that these could be weaponized by dedicated attackers who can keep requesting new IP addresses from public cloud providers until they are given the one they are looking for.

Overall, all the aforementioned error cases (4xx/5xx HTTP messages, TLS errors, timeout errors, and entirely inaccessible hosts) indicate that the server referenced by an IaC script does not offer what the IaC script expects it to, pointing to misconfigurations and therefore likely vulnerabilities.

4.3.4 Popularity Analysis. The relationship between the accessibility of remote references and GitHub star counts is shown in Figure 4. The data reveals a clear trend: Puppet code is consistently more problematic compared to Ansible, across all star ranges. Among repositories with zero stars, 68.31% of Puppet repositories have accessibility issues, while only 35.18% of Ansible repositories face the same problem. As the popularity increases, the percentage of problematic repositories for both tools rises, but Puppet consistently remains higher. In the 100+ star range, 88.24% of Puppet repositories are affected, compared to 53.03% for Ansible. This is a counterintuitive result that likely relates to functionality. The more capable an IaC script is, the more opportunity for remote references in its code and the more likely it is to be popular, compared to low-utility code.

4.3.5 Placeholder analysis. When manually reviewing the domain names that our pipeline identified and extracted from Ansible and Puppet scripts, we realized that many developers used domain names as placeholders in their scripts. For instance, developers would use `example.com` in a specific remote reference, with the expectation that other developers, when downloading their code, would replace these with their own domain names. While `example.com` and `example.org` are appropriate IANA-approved example domains and can be safely used, other developers would come up with their own placeholders, such as, `test.com` and `foobar.com`. The issue with these custom placeholder domain names is that they can be abused by attackers to exploit the infrastructure of developers who download and execute these scripts, before attempting to replace the placeholders.

To better understand the magnitude of this domain placeholder issue, we used ChatGPT-4o to flag instances of placeholder keywords in our list of domain names. In total, ChatGPT-4o identified a total of 182 domains that could be placeholders, with a total of 9,808 occurrences, accounting for 5.4% of all domains in Ansible and Puppet after removing duplicate domains as we mentioned in Section 4.3.2. Some of the most frequent domains were the IANA-approved `example.com` and `example.org`, which appeared 6,568 and 994 times in Ansible and Puppet, respectively. In addition, we also observed some domain names that are likely typos, such as `teste.br`, `exmaple.com` and `exampple.org`. In Table 6 (in the Appendix), we list all domains that were used at least 100 times in Ansible and Puppet scripts.

In terms of words embedded in domain names, ChatGPT-4o flagged a total of 65 keywords as placeholders. The most frequent keywords are "example" and "test." The frequency distribution of these keywords is visualized in Figure 3.

To the best of our knowledge, this is the first time that placeholder domain names have been discovered as a general threat vector in supply-chain attacks. This has to do with the nature of IaC scripts in the wild, some of which are meant to be used as templates for other developers. Developers who do not replace these placeholder domain names *before* executing these scripts, could end up downloading malware and infecting their infrastructure. We revisit this issue of placeholder domain names in Section 5 where we responsibly disclose vulnerabilities to affected developers and also propose a developer-centric countermeasure against remote-reference attacks.

4.3.6 Typosquatting and domain parking. Detecting typos in domain references is crucial, as trivial mistakes can lead to vulnerabilities exploitable through typosquatting attacks. In our study, we downloaded the most recent Tranco ranking and subsequently isolated the top 100,000 domains to serve as a foundational dataset.

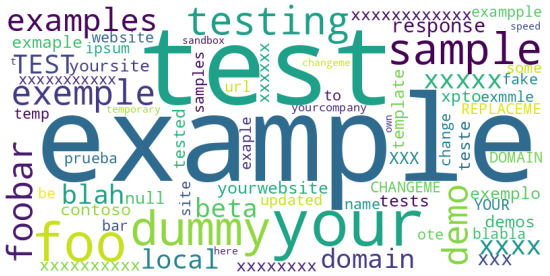


Figure 3: Word cloud showing the frequency distribution of keywords in domain names

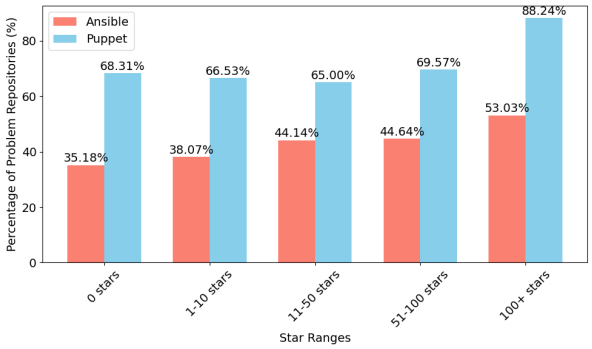


Figure 4: Percentage of Repositories with Inaccessible Remote References by GitHub

We then generated domain variants in these 100,000 domains using four distinct typographical error methodologies that are commonly used in prior work on typosquatting [20, 47, 53]: character omission, permutation, duplication, and substitution. The generated typo variants were then cross-referenced against the domains identified within Ansible and Puppet code. Our analysis identified that 21 Ansible repositories and 11 Puppet repositories contained at least one occurrence of these artificially generated typo variants. Moreover, when applying the same typo generation techniques to every domain within individual repositories, we observed vulnerabilities manifesting in 12 Ansible repositories and 5 Puppet repositories.

In addition to typosquatting attacks, domain parking represents another significant security concern within IaC. Domain parking refers to the practice of registering domain names without actively using them, often displaying placeholder or advertisement-filled pages [52, 54]. Domains that are left to expire and re-registered are often parked by their new owners. As a result, IaC domains that are currently parked are likely to have switched owners between the time a domain was referenced in an IaC script and the present. Parked domains can serve malware and are also typically for sale, allowing dedicated attackers to obtain control of high-value domain names. To investigate this issue, we conducted an analysis of domain parking in the same set of Ansible and Puppet repositories. We find that domain parking affected 208 Ansible repositories involving 98 unique domains and 91 Puppet repositories involving 33 unique domains. This underscores the critical need for continuous monitoring and assessment of domain references in IaC to preempt potential security risks associated with a domain's loss of ownership.

4.4 Code Smells and Content Integrity

For our last analysis of remote references included in IaC scripts, we want to understand how our findings compare to security smells proposed in prior work on IaC security [32, 34, 37]. Namely, we evaluate our results using GLITCH [37], a language-agnostic framework designed to detect code smells in IaC scripts, including an "integrity smell" which flags cases where developers rely on remote resources but do not check their integrity (e.g. they do not embed a cryptographic hash of a remote resource that they are downloading). For this analysis, we used GLITCH with its default settings and focused on scripts that contain at least one remote reference, totaling 194,511 for Ansible and 18,641 for Puppet, respectively.

Figure 5 and Figure 6 show the results of this analysis for both Ansible and Puppet IaC scripts. The top branches of both trees (labeled as "download and not safe") are cases where GLITCH correctly identifies a download *and* the fact that developers should have checked for that file's integrity but either did not or explicitly disabled integrity checks. In all these cases, GLITCH behaved as expected since, given how GLITCH is designed, there is no straightforward way for the tool to produce false positives.

For the remainder of cases (the bottom two branches of each Sankey diagram) which represent the vast majority of analyzed scripts, GLITCH concludes that either no download is involved or that whatever is being downloaded is safe because the developers are making use of built-in integrity mechanisms. Since all of these scripts that we used as input to GLITCH include remote references, there is, by definition, the opportunity to download something from a remote host. To understand why GLITCH concludes that no downloads are involved and whether it correctly identifies integrity mechanisms (i.e. its susceptibility to false negatives) we randomly selected 500 samples from each of these two branches and analyze their remote references. We present our findings below:

• ❶ **Not download:** In the majority of scripts that GLITCH marked as "safe" regarding integrity-related code smells, it did so because it concluded that no download was involved (53.9% for Ansible and 59.8% for Puppet). Through our sampling process, we identified the following reasons for the disagreement between our analysis and GLITCH's results:

• **Deployment environment setup - server identity:** This refers to scripts that are used during server setup which declare a server's identity. For example, a Puppet Nginx setup script includes a virtual host with the domain `othalland.xyz`, informing the web server that it is responsible for that domain name. While technically this is not a remote reference in and of itself, we argue that it will become a remote reference when developers start uploading HTML content to the server. That is, all references that are intended to be local (e.g. `othalland.xyz/script.js`) will in fact be requesting that resource from another server, since the domain `othalland.xyz` was available for registration at the time of our analysis. We discovered that 37.3% and 26.2% are cases of server identity configurations in Ansible and Puppet, respectively.

• **Deployment environment setup - repository pulling:** This refers to scripts that pull dependencies from third-party sources. For example, the script adds a remote reference (`http://yum.spacewalkproject.org`) in the list of trusted software repos. Even though a key for the downloaded software is also specified, an attacker who

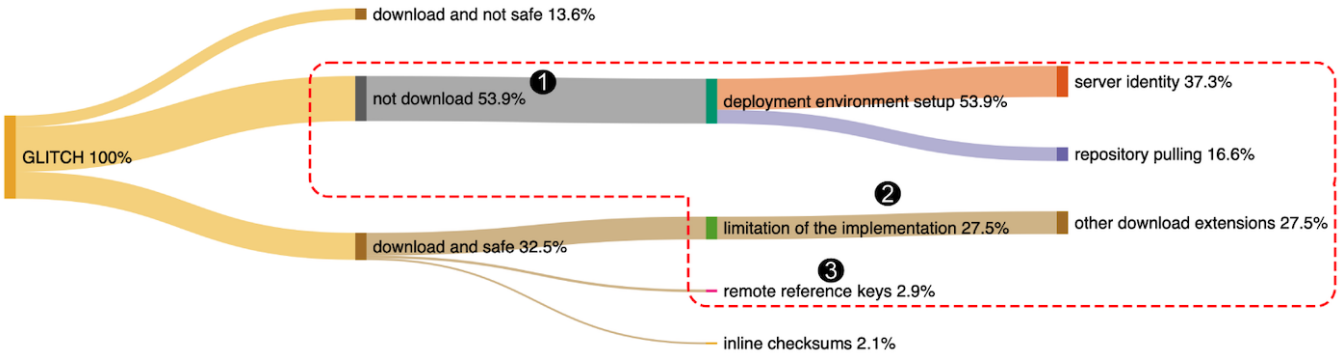


Figure 5: The breakdown of the integrity security smell detection using GLITCH with our extended dynamic analysis in Ansible. With our analysis ❶, ❷, and ❸ are the false negatives that were generated by GLITCH

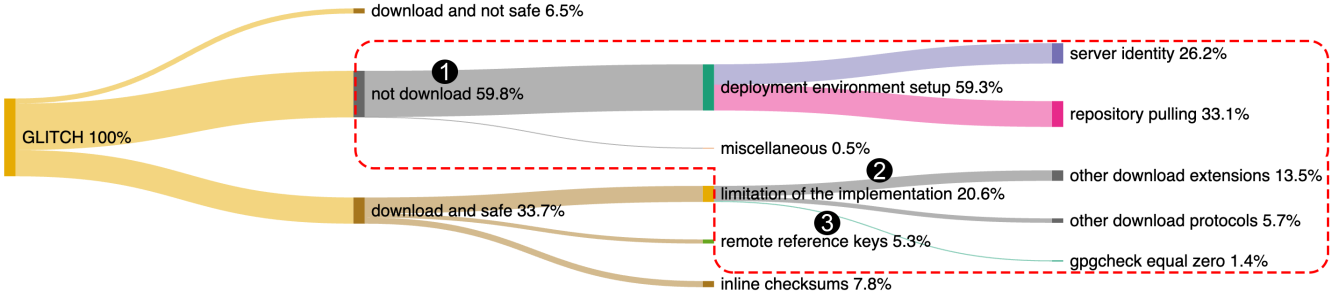


Figure 6: The breakdown of the integrity security smell detection using GLITCH with our extended dynamic analysis in Puppet. With our analysis ❶, ❷, and ❸ are the false negatives that were generated by GLITCH

registers `spacewalkproject.org` will also be able to provide a new key for his own malicious software. We discovered that 16.6% of Ansible scripts and 33.1% of Puppet scripts involve repository pulling.

• ❷ **Limitation of implementation:** This refers to the limitations in the current implementation of GLITCH, as evidenced by 27.5% cases in Ansible and 20.6% cases in Puppet. These limitations fall into three categories: other types of download protocols (e.g., `git://`), additional download extensions such as `dmg`, `git`, `msi`, `db`, `exe`, `iso`, and false negatives where GLITCH does not account for disabled values of `gpgkey` (i.e., set to `0`). All of these limitations can be addressed by adjusting the configuration of GLITCH. We will be reaching out to the authors of GLITCH upon publication of this paper to suggest these improvements.

• ❸ **Remote reference keys:** This category includes scripts that contain remote reference keys. We discovered 2.9% such Ansible scripts and 5.3% Puppet ones. GLITCH flags these scripts as safe because the scripts make reference of integrity-related checksums. However, these integrity checksums are pulled from remote repositories, as opposed to being listed inline with the playbooks. As a result, these playbooks are equally vulnerable to supply-chain attacks as those that do not include any integrity checking.

5 Disclosure and Countermeasures

Disclosure to affected IaC developers To understand how IaC developers perceive the threat of dangling remote references in their scripts and whether they agree with our assessment, we reached out to the owners of the most popular repos where we identified expired domain names. We sorted the affected repos by GitHub stars to determine their popularity, and attempted to contact the owners of the fifty most popular repositories in both Ansible and Puppet.

We were able to locate the contact information for 27 developers of Ansible projects and 23 developers for Puppet and then contacted them. On the Ansible side, we received nine responses, with only one person not considering our finding to be a security issue. Of the remaining eight responses, three developers informed us that they updated the relevant parts of their code after receiving our email, and four acknowledged it as a security issue without committing that they would change it. Two of them stated that their public IaC scripts including remote references are meant to be used as examples with placeholder domain names (originally discussed in Section 4.3.5) and therefore not intended to be directly downloaded and executed. One developer who updated their code after receiving our email, admitted that the domain was previously their own, but they had let it expire. On the Puppet front, we only received a response from one developer who acknowledged it as a problem and subsequently deleted the repository. One more Puppet developer deleted their repository after we contacted them, but never responded to us. The main content of all developers' responses is summarized in Table 5, located in Appendix.

Overall, in the vast majority of cases ($N-1$) where developers responded to us, they agreed with our assessment that the placeholder/expired domain names used in their public IaC scripts could be abused by attackers.

Protecting IaC developers The problem of residual trust on the web is difficult to solve as a general case. However, in recognition that developers are sophisticated users and therefore easier to assist

than everyday users, we developed DependoScope, an extension for VS code, to identify remote references in Ansible and Puppet playbooks, as IaC developers are developing them or whenever they open them up to review their source code. For every detected remote reference, DependoScope performs a series of common-sense tests and warns the developer if any one of these tests fail. Our tests are as follows:

• **DNS resolutions.** Ensure that all domains (standalone or part of URLs) resolve to IP addresses.

• **Placeholder domain detection.** Detects placeholder domains using heuristics. If found, DependoScope steers them towards IANA-approved example domains like example.com which will never fall in the wrong hands.

• **Remote-service detection.** DependoScope ensures that not only a remote reference resolves but that the expected network service (e.g. a web server) is running on the remote host. To do that, DependoScope conducts a network scan of remote hosts and compares their open ports (if any) against the scheme that the developer is using in a playbook (e.g. matching an `https` scheme to a service on port 443).

• **Typo detection.** Our extension detects domain name typos vulnerable to typosquatting attacks on IaC code. Detection operates both globally, by checking variations of popular domains (from the Tranco list), and locally, by checking variations of domains previously used within the same playbook.

Since not all of this functionality can be implemented through the native capabilities offered to VS Code extensions, we offload some of them to a remote service (such as the network scanning of remote hosts). We developed this remote service as a Docker image which can be executed locally or remotely, depending on the needs to a given IaC developer. We will be open-sourcing the code of DependoScope along with the rest of our study artifacts, upon publication of this paper. A video demonstration of DependoScope is available here: https://vimeo.com/1020224971.

6 Limitations

Our analysis of the official Ansible and Puppet documentation was a best-effort one, where we used the names of parameters, their description (where available) and code examples to conclude that they were related to remote references. Specifically for Ansible, we discovered a total of 7,087 parameters that could be remote references. Among them, the description of some tasks is "no description" (2,455 of the total number of parameters). In these cases, we had to rely solely on the parameter names to conclude whether they were related to remote references. As a result, it is possible that we may have flagged some remote-referencing parameters as non-remote-referencing and vice versa. This does not affect our analysis of the real-world usage of remote references in GitHub repos where we rely on our documentation crawls to filter out unrelated YAML code and interpret the capabilities available to attackers who can take control of a remote reference.

Finally, throughout our analysis, whenever we had to choose between false positives and false negatives, we opted for false negatives to ensure that we are not artificially magnifying the problem of remote references in IaC tools. For instance, we only highlight domain names that are available for registration by attackers, and completely ignore domain names that are listed in domain-name marketplaces. Dedicated attackers could purchase these domains and attack the infrastructure that depends on them. Similarly, we do not try to identify whether a responsive remote host is still in the hands of the original owner vs. a new owner who is unaware of the significance of their allocated IP address. Overall, we argue that the number of presented vulnerabilities and misconfigurations are a lower bound of the true size of the problem in the wild. Finally, our use of ChatGPT-4o for placeholder domain detection (Section 4.3.5) may influence the reproducibility. Although we manually validated a sample of its outputs and are confident in the overall findings, some misclassifications remain possible. We will leave more reproducible alternatives for future research to explore.

7 Related Work

To the best of our knowledge, this paper is the first one to underline the issue of remote references in Infrastructure-as-Code (IaC) technologies, evaluate the prevalence of this issue in the wild, and design pragmatic countermeasures to assist developers in avoiding unintended remote references in their code. Below, we discuss other relevant work in IaC security and in the general area of residual trust on the web.

Security of IaC. Beyond software vulnerabilities in the IaC tools themselves (i.e. the interpreters translating playbooks into concrete actions), IaC security issues arise because of the specific ways developers use them. A general theme of past work in how developers use IaC is centered around "code smells." A code smell is a characteristic in a program's source code that indicates that a specific piece of code may be defective, because it does not follow best software-engineering practices. Prior work has looked at different types of code smells in IaC tools, from both a security and a non-security perspective [5, 24, 31–35, 40, 41].

Our work was inspired by the security-specific code smells defined by Rahman et al. [32, 34] which were then made tool-agnostic by Saavedra et al. through their GLITCH tool [38]. One of the smells defined by these works is that of "Integrity checking" where prior tools attempt to statically detect downloads and missing integrity checks associated with these downloads. As we showed throughout the paper, a purely static analysis of remote dependencies is not sufficient since that analysis cannot reason about the state of the outside world (e.g. is the domain name in an IaC playbook still registered and resolving to a reachable host). Moreover, existing integrity smells are focused on individual files being downloaded and entirely miss the cloning of repositories, the installation of packages, and the imbuing of identity on remote servers. As a result, the state-of-the-art GLITCH tool missed more than half of the remote references we discovered in our analysis (Section 4.4). We therefore argue that existing static-analysis tools for IaC playbooks must be supplemented by dynamic analysis, at the time when playbooks are developed (as our proposed DependoScope extension does) and ideally in regular intervals, as part of a generic attack-surface-management solution.

Residual trust of domain names. The issue of remote references in IaC tools is another instance of the general problem of residual trust and content integrity on the web. Prior work has studied instances of this problem extensively (including websites, browser extensions, email addresses, and Android applications) [10, 14, 15,

19, 22, 25, 36, 44], yet general-purpose solutions are absent because of the challenge of reliably defining identity over domain names (i.e. how does one know when a domain name changes hands) [15], and the difficulty of designing code-integrity constructs that are flexible enough to allow for updates yet rigid enough to detect abuses [18, 42, 45].

Compared to other instances of residual trust, the IaC domain has distinct advantages and disadvantages. For example, since IaC is used to provision computing infrastructure and deploy software, remote references are definitionally more dangerous than a general-purpose remote reference on the web. Unlike web resources that are fetched on every page visit, IaC scripts are inherently idempotent and developers routinely rerun the same playbooks to maintain their infrastructure. As a result, a hijacked remote reference will compromise not only new infrastructures, but any infrastructure that is reconfigured using the same playbook in the future. At the same time, since IaC's audience is primarily developers, this allows us to design tools and countermeasures that would be too complex for a general audience, as demonstrated by our extension.

8 Conclusion

In this paper, we investigated the abuse potential of remote references in Infrastructure as Code (IaC) tools. We focused on Ansible and Puppet IaC playbooks and showed how thousands of IaC modules across both tools enable developers to include remote references in their code. These remote references carry residual trust and thus the domains and hosts associated with them can be hijacked by attackers to perform supply-chain attacks and compromise an organization's computing infrastructure. Through a large scale analysis involving 3,580,699 IaC playbooks across 247,131 public GitHub repos, we discovered 463 expired domains and 10,400 misconfigured remote hosts. Among others, we discover the issue of placeholder domain names that has never been encountered in prior works focusing on residual trust and as well as instances of typosquatting vulnerabilities across Ansible and Puppet repositories. To aid developers in authoring secure IaC playbooks, we proposed DependoScope, a code-editor extension that can identify likely-vulnerable remote references in playbooks, during their development. To enable additional work in this new area of IaC, we will be open-sourcing our code and data upon publication of this paper.

Acknowledgments

We thank the reviewers for their helpful comments. This work was supported by the Office of Naval Research (ONR) under grant N00014-24-1-2193 as well as by the National Science Foundation (NSF) under grants CNS-1941617 and CNS-2211575.

References

[1] 2024. Buy a Domain Name - Register Cheap Domain Names from $0.99 - Namecheap. https://www.namecheap.com/. Accessed on Jan. 25, 2024.
[2] 2024. Domain Names, Websites, Hosting & Online Marketing Tools - GoDaddy. https://www.godaddy.com/. Accessed on Jan. 25, 2024.
[3] 2024. Register Domains & Build Your Website with Dynadot! https://www.dynadot.com/. Accessed on Jan. 25, 2024.
[4] 2024. Using Variables — Ansible Community Documentation. https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_variables.html. Accessed on Aug. 13, 2024.
[5] E. Van Der Bent, J. Hage, J. Visser, and G. Gousios. 2018. 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)*. IEEE, 164–174. https://doi.org/10.1109/SANER.2018.8330206
[6] Henry Birge-Lee, Yixin Sun, Anne Edmundson, Jennifer Rexford, and Prateek Mittal. 2018. Bamboozling certificate authorities with {BGP}. In *27th USENIX Security Symposium (USENIX Security 18)*. 833–849.
[7] K. Borgolte, T. Fiebig, S. Hao, C. Kruegel, and G. Vigna. 2018. Cloud Strife: Mitigating the Security Risks of Domain-Validated Certificates. In *Proceedings 2018 Network and Distributed System Security Symposium*. Internet Society, San Diego, CA. https://doi.org/10.14722/ndss.2018.23327
[8] Davide Canali and Davide Balzarotti. 2013. Behind the scenes of online attacks: an analysis of exploitation behaviors on the web. In *20th Annual Network & Distributed System Security Symposium (NDSS 2013)*.
[9] Davide Canali, Davide Balzarotti, and Aurélien Francillon. 2013. The role of web hosting providers in detecting compromised websites. In *Proceedings of the 22nd international conference on World Wide Web*. 177–188.
[10] Nicholas Carlini, Matthew Jagielski, Christopher A Choquette-Choo, Daniel Paleka, Will Pearce, Hyrum Anderson, Andreas Terzis, Kurt Thomas, and Florian Tramèr. 2024. Poisoning web-scale training datasets is practical. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 407–425.
[11] Thomas Claburn. 2022. Email domain for NPM lib with 6m downloads a week grabbed by expert to make a point. https://www.theregister.com/2022/05/10/security_npm_email/.
[12] Chris C Demchak and Yuval Shavitt. 2018. China's maxim–leave no access point unexploited: The hidden story of China telecom's BGP hijacking. *Military Cyber Affairs* 3, 1 (2018), 7.
[13] Artyom Gavrichenkov. 2015. Breaking HTTPS with BGP hijacking. *Black Hat. Briefings* (2015).
[14] D. Gruss, M. Schwarz, M. Wübbeling, S. Guggi, T. Malderle, S. More, and M. Lipp. 2018. Use-After-FreeMail: Generalizing the Use-After-Free Problem and Applying It to Email Services. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. ACM, New York, NY, USA, 297–311. https://doi.org/10.1145/3196494.3196514
[15] C. Lever, R. Walls, Y. Nadji, D. Dagon, P. McDaniel, and M. Antonakakis. 2016. Domain-Z: 28 Registrations Later Measuring the Exploitation of Residual Trust in Domains. In *2016 IEEE Symposium on Security and Privacy (SP)*.
[16] Liam Tung. 2022. Microsoft warns: This forgotten open-source web server could let hackers 'silently' gain access to your system. https://www.zdnet.com/article/microsoft-warns-this-forgotten-open-source-web-server-could-let-hackers-silently-gain-access-to-your-system/.
[17] Microsoft. 2024. What is infrastructure as code (IaC)? https://learn.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code.
[18] Dimitris Mitropoulos, Konstantinos Stroggylos, Diomidis Spinellis, and Angelos D Keromytis. 2016. How to train your browser: Preventing XSS attacks using contextual script fingerprints. *ACM Transactions on Privacy and Security (TOPS)* 19, 1 (2016), 1–31.
[19] T. Moore and R. Clayton. 2014. The Ghosts of Banking Past: Empirical Analysis of Closed Bank Websites. In *Financial Cryptography and Data Security*.
[20] Tyler Moore and Benjamin Edelman. 2010. Measuring the perpetrators and funders of typosquatting. In *International Conference on Financial Cryptography and Data Security*. Springer, 175–191.
[21] N. Nikiforakis, L. Invernizzi, A. Kapravelos, S. Van Acker, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna. 2012. You Are What You Include: Large-Scale Evaluation of Remote JavaScript Inclusions. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*.
[22] N. Nikiforakis, L. Invernizzi, A. Kapravelos, S. Van Acker, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna. 2012. You Are What You Include: Large-Scale Evaluation of Remote JavaScript Inclusions. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. ACM, New York, NY, USA, 736–747. https://doi.org/10.1145/2382196.2382274
[23] R. Opdebeeck, A. Zerouali, and C. De Roover. 2023. Control and Data Flow in Security Smell Detection for Infrastructure as Code: Is It Worth the Effort?. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. IEEE, 534–545.
[24] R. Opdebeeck, A. Zerouali, and C. De Roover. 2023. Control and Data Flow in Security Smell Detection for Infrastructure as Code: Is It Worth the Effort?. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. IEEE, Melbourne, Australia, 534–545. https://doi.org/10.1109/MSR59073.2023.00079
[25] E. Pariwono, D. Chiba, M. Akiyama, and T. Mori. 2018. Don't Throw Me Away: Threats Caused by the Abandoned Internet Resources Used by Android Apps. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. ACM, New York, NY, USA, 147–158. https://doi.org/10.1145/3196494.3196554
[26] E. Pauley, R. Sheatsley, B. Hoak, Q. Burke, Y. Beugin, and P. McDaniel. 2022. Measuring and Mitigating the Risk of IP Reuse on Public Clouds. In *2022 IEEE Symposium on Security and Privacy (SP)*. 558–575.
[27] J.M Porup. 2022. Why abandoned domain names are so dangerous. https://www.csoonline.com/article/566127/dont-abandon-that-domain-name.html.

[28] Puppet by Perforce. 2023. Puppet. https://www.puppet.com/. Accessed on Dec. 28, 2023.
[29] Puppet by Perforce. 2024. Puppet Forge. https://forge.puppet.com/. Accessed on Jan. 23, 2024.
[30] A. Rahman, D. B. Bose, Y. Zhang, and R. Pandita. 2024. An Empirical Study of Task Infections in Ansible Scripts. *Empirical Software Engineering* 29, 1 (2024), 34.
[31] A. Rahman and C. Parnin. 2023. Detecting and Characterizing Propagation of Security Weaknesses in Puppet-Based Infrastructure Management. *IEEE Transactions on Software Engineering* (2023), 1–18. https://doi.org/10.1109/TSE.2023.3265962
[32] A. Rahman, C. Parnin, and L. Williams. 2019. The Seven Sins: Security Smells in Infrastructure as Code Scripts. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, Montreal, QC, Canada, 164–175. https://doi.org/10.1109/ICSE.2019.00033
[33] A. Rahman, A. Partho, D. Meder, and L. Williams. 2017. Which Factors Influence Practitioners' Usage of Build Automation Tools?. In *2017 IEEE/ACM 3rd International Workshop on Rapid Continuous Software Engineering*. IEEE, 20–26.
[34] A. Rahman, M. R. Rahman, C. Parnin, and L. Williams. 2020. Security Smells in Ansible and Chef Scripts: A Replication Study. arXiv:arXiv:1907.07159 arXiv:1907.07159.
[35] A. Rahman and L. Williams. 2018. Characterizing Defective Configuration Scripts Used for Continuous Deployment. In *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 34–45. https://doi.org/10.1109/ICST.2018.00014
[36] S. Roth, T. Barron, S. Calzavara, N. Nikiforakis, and B. Stock. 2020. Complex Security Policy? A Longitudinal Analysis of Deployed Content Security Policies. In *Proceedings of the 27th Network and Distributed System Security Symposium*.
[37] N. Saavedra and J. F. Ferreira. 2022. GLITCH: Automated Polyglot Security Smell Detection in Infrastructure as Code. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. ACM, Rochester MI USA, 1–12. https://doi.org/10.1145/3551349.3556945
[38] N. Saavedra, J. Gonçalves, M. Henriques, J. F. Ferreira, and A. Mendes. 2023. Polyglot Code Smell Detection for Infrastructure as Code with GLITCH. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, Luxembourg, Luxembourg, 2042–2045. https://doi.org/10.1109/ASE56229.2023.00162
[39] Samantha Schwartz. 2018. 11K organizations use outdated software linked to Equifax breach. https://www.ciodive.com/news/11k-organizations-use-outdated-software-linked-to-equifax-breach/522960/.
[40] J. Schwarz, A. Steffens, and H. Lichter. 2018. Code Smells in Infrastructure as Code. In *2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC)*. IEEE, 220–228. https://doi.org/10.1109/QUATIC.2018.00040
[41] T. Sharma, M. Fragkoulis, and D. Spinellis. 2016. Does Your Configuration Code Smell?. In *Proceedings of the 13th International Conference on Mining Software Repositories*. ACM, Austin Texas, 189–200. https://doi.org/10.1145/2901739.2901761
[42] Johnny So, Michael Ferdman, and Nick Nikiforakis. 2023. The more things change, the more they stay the same: Integrity of modern javascript. In *Proceedings of the ACM Web Conference 2023*. 2295–2305.
[43] J. So, N. Miramirkhani, M. Ferdman, and N. Nikiforakis. 2022. Domains Do Change Their Spots: Quantifying Potential Abuse of Residual Trust. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2130–2144.
[44] J. So, N. Miramirkhani, M. Ferdman, and N. Nikiforakis. 2022. Domains Do Change Their Spots: Quantifying Potential Abuse of Residual Trust. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, San Francisco, CA, USA, 2130–2144. https://doi.org/10.1109/SP46214.2022.9833609
[45] Pratik Soni, Enrico Budianto, and Prateek Saxena. 2015. The sicilian defense: Signature-based whitelisting of web javascript. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 1542–1557.
[46] Sophos. 2018. Server? What server? Site forgotten for 12 years attracts hacks, fines. https://news.sophos.com/en-us/2018/05/22/server-what-server-site-forgotten-for-12-years-attracts-hacks-fines/.
[47] Jeffrey Spaulding, DaeHun Nyang, and Aziz Mohaisen. 2017. Understanding the effectiveness of typosquatting techniques. In *Proceedings of the fifth ACM/IEEE Workshop on Hot Topics in Web Systems and Technologies*. 1–8.
[48] United States Government Accountability Office. 2018. DATA PROTECTION: Actions Taken by Equifax and Federal Agencies in Response to the 2017 Breach. https://www.warren.senate.gov/imo/media/doc/2018.09.06%20GAO%20Equifax%20report.pdf.
[49] Marie Vasek and Tyler Moore. 2014. Identifying risk factors for webserver compromise. In *Financial Cryptography and Data Security: 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers 18*. Springer, 326–345.
[50] Pierre-Antoine Vervier, Olivier Thonnard, and Marc Dacier. 2015. Mind Your Blocks: On the Stealthiness of Malicious BGP Hijacks.. In *NDSS*.
[51] Thomas Vissers, Timothy Barron, Tom Van Goethem, Wouter Joosen, and Nick Nikiforakis. 2017. The Wolf of Name Street: Hijacking Domains Through Their Nameservers. In *Proceedings of the 24th ACM Conference on Computer and Communications Security (CCS)*.
[52] Thomas Vissers, Wouter Joosen, and Nick Nikiforakis. 2015. Parking Sensors: Analyzing and Detecting Parked Domains. In *Proceedings of the 22nd Network and Distributed System Security Symposium (NDSS)*.
[53] Yi-Min Wang, Doug Beck, Jeffrey Wang, Chad Verbowski, and Brad Daniels. 2006. Strider Typo-Patrol: Discovery and Analysis of Systematic Typo-Squatting. *SRUTI* 6, 31-36 (2006), 2–2.
[54] Johannes Zirngibl, Steffen Deusch, Patrick Sattler, Juliane Aulbach, Georg Carle, and Mattijs Jonker. 2022. Domain Parking: Largely Present, Rarely Considered!. In *Proc. Network Traffic Measurement and Analysis Conference (TMA) 2022*.

A. Developers Feedback

Table 5 shows the anonymized responses we received when contacting the owners of Github repos regarding their vulnerable Ansible and Puppet repos. $(N-1)$ of the developers who responded to us agreed with our assessment.

Table 5: Highlight of developer feedback. All but one of the developers who responded to us agreed with our assessment of the risk posed by expired domain names in their code.

Tool	Feedback
Ansible	"Although it's highly unlikely anyone would be opening the site since it's not represented as a clickable URL on a network device, someone might get curious as to what the URL goes to. In that case, I see the concern."
Ansible	"We have investigated this and do not believe it is a security issue that needs to be addressed at this time."
Ansible	"Yep, you are absolutely right. And it will be an ongoing effort to educate us and all other developers to know about the risk and resolution of problems like this"
Ansible	"I agree, that it might be better to ask for a domain explicitly"
Ansible	"We added those scripts with an assumption that the user who copies from examples to create an end-to-end playbook would modify the values according to the needs."
Ansible	"I removed the old webpage. It was mine, but I no longer cared for it."
Ansible	"I think the assessment is still valid."
Ansible	"This repo has not been updated many years ago and actually it is not being used and it doesn't provide with any guarantees. Anyway, I'll review it and update soon."
Puppet	"It's a personal prototype project which is not in use for many years. I just deleted it to avoid being used maliciously."
Ansible	"Thank you!"

B. Analysis of Placeholder Domains and Embedded Keywords

Table 6 shows the most popular placeholder domains that Ansible and Puppet developers used in their public code. Any non-IANA-designated placeholder domains can be weaponized by attackers to deliver malware and infect developers' computing infrastructures.

Table 6: Common Placeholder Domains and Their Frequency

Domain	Frequency
example.com	6,568
example.org	994
example.net	392
test.com	353
domain.com	265
test.business	162
example2.com	122
foobar.com	112
sample.com	104
ansibletestzone.com	103