

Unclaimed Territory: Analyzing Subdomain Takeover Risks in GitHub Pages

Abstract—GitHub is among the most widely-used platforms for open-source collaboration. GitHub Pages enables repositories to publish project websites using `*.github.io` and to map a custom domain via DNS. In practice, the custom-domain configuration spans multiple artifacts and administrative planes (repository settings/CNAME, DNS records, and Pages hosting state), which can drift over time due to routine lifecycle events such as repository deletion/transfer, archival, or changes in publishing mode. This drift can create *subdomain takeover risk*: a dangling custom domain that still points to GitHub Pages may be re-associated with an unrelated Pages site.

In this paper, we present a lightweight testing and analysis approach to identifying GitHub Pages subdomain takeover risk using public version-control history together with DNS and HTTP measurements. In a large-scale study, we identify 284 custom subdomains whose DNS CNAME records point to deleted or unclaimed GitHub Pages sites. We validate exploitability using a controlled, non-disruptive proof-of-control to establish ground truth, and we conduct a capability and risk study of the GitHub Pages hosting model, showing how a reclaimed custom domain can stage static content, modular browser side code, worker APIs, service-worker registration, and visible browser-mediated requests to external dynamic services. Finally, we present a survey of 288 participants to evaluate how the public and GitHub users perceive subdomain takeover risk and domain trust cues in the GitHub ecosystem.

Index Terms—Open-Source Software, GitHub Pages, Configuration Drift

I. INTRODUCTION

GitHub is widely used as a platform for open-source development, hosting millions of repositories that foster global collaboration and innovation with a robust system for developers to share code linked with their identity [1], [2]. GitHub Pages is a related service that enables users to host static websites directly from a GitHub repository. By default, GitHub Pages are hosted on subdomains of the form `bob.github.io`, but users can also link custom domains to point to their GitHub Page by adding Canonical Name (CNAME) records to their custom Domain Name Service records and repository (e.g., Figure 1 shows the domain `bobby.cloud.com` pointing to the GitHub Page `bob.github.io`). This functionality improves accessibility and simplifies website hosting for personal and project-based sites [3], [4]. As of February 2025, more than 250,000 websites are currently hosted using GitHub Pages [5].

Meanwhile, *subdomain takeover* commonly arises when a domain’s DNS points to an external service that is no longer serving the intended resource, leaving the subdomain *dangling* and potentially claimable [6], [7]. This situation commonly arises during routine lifecycle events such as repos-

itory deletion, project migration, ownership transfer, or organizational restructuring, where code and infrastructure evolve asynchronously. In particular, DNS configuration typically lives outside the repository’s version control and code review process, so stale entries can persist long after the project’s hosting endpoint has changed. As a result, the canonical subdomain, often treated as part of the project’s public interface for documentation, onboarding, and release communication, can become “dangling” while still appearing legitimate to users and downstream tooling.

The CNAME entries in the DNS records for a domain may continue to resolve to a service that has been deleted, creating an opportunity for an unauthorized takeover. Attackers can then take over the subdomain by exploiting the window of time after the original service has been deleted while the CNAME record persists. Once in control, they can redirect traffic, host phishing sites, or distribute malware, all while appearing as part of the legitimate organization due to the unchanged DNS configuration [6]. Numerous real-world incidents highlight the severity of subdomain takeovers, affecting major companies and platforms [7], [8], [9]. For instance, an unclaimed Uber subdomain `saostatic.uber.com` hosted on Amazon CloudFront was exploited, enabling attackers to manipulate cookies and bypass authentication across `*.uber.com` in 2017 [10]. Subdomain takeover typically requires the target service to be removed and for the attacker to create a new malicious service using the same name. However, in this paper, we explore a specific threat within the GitHub Pages architecture that allows attackers to take over domains pointing to `*.github.io` sites *without* having to take over the underlying repository.

GitHub’s ecosystem amplifies this dynamic: canonical subdomains connected to OSS projects are frequently referenced across READMEs, documentation, blog posts, tutorials, and issue discussions, which gives them durable visibility and implicit credibility [11], [12], [13]. This makes subdomain takeover particularly harmful because the project website is not merely a marketing page, it is part of the project’s maintenance surface. When a canonical documentation URL silently changes hands or simply decays over time, the project can suffer misleading setup instructions, broken references (“link rot”), and downstream coordination cost that slows onboarding and increases support burden [14], [15], [13]. Beyond these reliability and maintenance harms, the same drift can become an abuse vector: a newly controlled canonical subdomain can stage arbitrary static content, redirects, files, and browser-side code under a name that users and downstream tooling may still associate with the original project [16], [17], [18], [19].

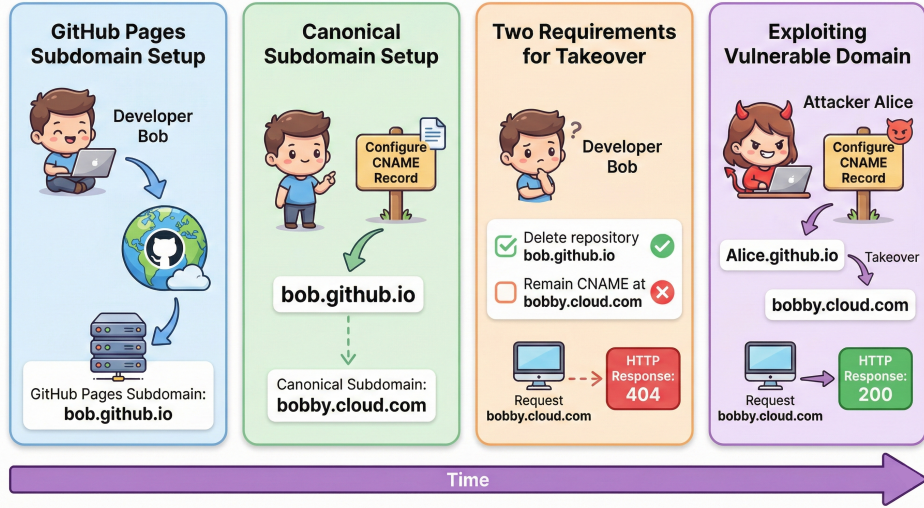


Fig. 1: Subdomain takeover risk induced by configuration drift in GitHub Pages. It begins with Bob, a developer, hosting a GitHub Pages subdomain (`bob.github.io`). Next, Bob configures a canonical subdomain (`bobby.cloud.com`) by linking it to his GitHub Pages subdomain using a CNAME record. The vulnerability arises when two requirements are met: (1) Bob deletes his GitHub repository (`bob.github.io`), and (2) the CNAME record for `bobby.cloud.com` remains unchanged, still pointing to the now non-existent GitHub Pages subdomain. This enables an attacker, Alice, to claim the abandoned subdomain (`bobby.cloud.com`) by linking it to her own repository (`alice.github.io`). Even though `bobby.cloud.com` continues to point to `bob.github.io`, GitHub Pages forwards requests for `bobby.cloud.com` to `alice.github.io` without Alice having to compromise `bob.github.io`.

However, detecting subdomain takeovers on GitHub Pages presents a significant challenge, as existing tools such as Subjack, Amass, and Subfinder rely on outdated wordlists, leading to high false positives and inefficient scanning processes [20], [21], [22]. These limitations make current detection methods ineffective for GitHub Pages.

To the best of our knowledge, this paper presents the first study to explore subdomain takeovers within the open-source community, which is a distinct and arguably easier subdomain takeover mechanism specific to the GitHub Pages architecture. Our investigation focuses on four key aspects: (1) Feasibility of Subdomain Takeover Detection in GitHub Pages: GitHub Pages takeover risk is rooted in a cross-artifact inconsistency between version-controlled configuration and out-of-repository operational state. This motivates developer and workflow-aware detection: rather than enumerating subdomains blindly, we leverage public version control history to recover candidate custom domains and then validate their current DNS/HTTP state to identify subdomains susceptible to takeover. Analyzing a dataset containing version control records from 90 million repositories, we identify 19,632 domains using GitHub Pages services and confirm 284 subdomains susceptible to takeover, and present an algorithmic approach to detecting when new such subdomains appear. (2) Capability and Risk of Vulnerable Subdomains: We use author-owned GitHub Pages resources to empirically measure what a takeoverable custom domain can support: static staging, benign client-side JavaScript, ES module loading, Web Worker execution, Service Worker registration, client-side routing, metadata presentation, redirects, file hosting, cross-origin resource loading, browser-mediated

external requests, and design-analysis of dynamic API staging through visible third-party services. (3) User Perception and Trust in Compromised Domains: We examine how GitHub users and the general public perceive subdomains susceptible to takeover with a survey of 288 participants. Our results reveal that these vulnerable subdomains are considered more trustworthy than phishing domains or those generated by Domain Generation Algorithms (DGA). When participants were asked to choose the more trustworthy domain, vulnerable subdomains were selected up to 5.8 times more than phishing domains and 17.25 times more than DGA domains. (4) Mitigation Strategies: We propose strategies to mitigate subdomain takeovers on GitHub.

Specifically, we claim the following contributions:

- 1) We present the first systematic study on subdomain takeover on GitHub, focusing on a weaker adversary who can re-associate a dangling custom domain with a new GitHub Pages site without compromising the original repository.
- 2) We propose a consistency-checking technique to detect GitHub Pages subdomain takeover risk by leveraging public version-control history together with DNS and HTTP measurements.
- 3) We conduct a controlled capability and risk study showing that the GitHub Pages hosting model supports static staging, modular browser-side code, and browser-mediated external requests, and can be composed with visible third-party dynamic services through ordinary browser APIs.
- 4) We survey GitHub users and the public to assess trust in

vulnerable subdomains versus other malicious domains. We also explore mitigation strategies to enhance security and raise awareness of subdomain takeovers.

The remainder of this paper is organized as follows: Section II presents a motivating example, Section III analyzes related work, Section IV describes our research questions, approach to detect subdomain takeover, and survey design, Section V presents the results, Section VI describes the discussion and implications, Section VII discusses limitations, Section VIII clarify the ethics of our study and Section IX concludes the paper.

II. MOTIVATING EXAMPLE

In this section, we illustrate how a website is hosted using GitHub Pages with a custom domain and how ordinary maintenance actions (e.g., deletion or migration of a Pages repository) can leave a dangling configuration that enables subdomain takeover, as shown in Figure 1. GitHub Pages is a feature that enables users to host static websites directly from a GitHub repository. Users can either use the default **GitHub Pages subdomain** (*.github.io) or configure a custom **canonical subdomain** through a Canonical Name (CNAME) entry in the Domain Name System (DNS) records associated with the canonical domain.

GitHub Pages Subdomain Setup: To deploy a website using GitHub Pages, a user (e.g., Bob) first creates a repository named `bob.github.io` and uploads static website files (HTML, JavaScript, etc.). Once GitHub Pages is enabled, GitHub automatically serves the website under the subdomain `bob.github.io` (corresponding to the GitHub username), which we refer to as the **GitHub Pages Subdomain**.

Canonical Subdomain Setup: In addition to the default `bob.github.io` subdomain, Bob can also map his own **canonical subdomain** (e.g., `bobby.cloud.com`) to his GitHub Pages site following the steps below:

- 1) Creates a CNAME file in the repository's root directory containing the canonical subdomain (`bobby.cloud.com`).
- 2) Updates the DNS settings on his domain registrar to add a CNAME record pointing `bobby.cloud.com` to `bob.github.io`.
- 3) GitHub detects the CNAME file and ensures that requests to `bobby.cloud.com` are redirected to the content hosted on `bob.github.io`.

CNAME records are a type of DNS records that link an alias domain like `bobby.cloud.com` to another like `bob.github.io`, rerouting requests from one domain to the target of the CNAME record instead.

In practice, a canonical subdomain like `bobby.cloud.com` is rarely “just a website”: it is often referenced from README files, release notes, package metadata, and issue templates, and it can serve as a stable entry point for documentation across versions. Yet DNS registrar configuration typically lives outside the repository's version control and review process. This separation makes it

easy for a project's operational state to drift from its code state, especially in volunteer-driven OSS where ownership changes and infrastructure migrations are common[23], [24].

Two Requirements For Takeover: Subdomain takeover occurs when a previously-configured canonical subdomain remains pointed to a deleted GitHub Pages repository. As shown in Figure 1, Bob originally hosted a website on `bob.github.io` and had a canonical subdomain `bobby.cloud.com` point to their website. However, if Bob (1) deletes his repository but (2) forgets to update or remove the CNAME record in his domain registrar, the canonical subdomain `bobby.cloud.com` remains *dangling* — it continues to point to `bob.github.io`, even though the repository no longer exists. `bobby.cloud.com` is now a **vulnerable canonical subdomain**.

Exploiting Vulnerable Domain: Typical subdomain takeover attacks would require a malicious user to gain control of the target resource (in this example, `bob.github.io`) to take over an external subdomain with a CNAME record pointing to that resource. However, GitHub Pages does not perform further verification without additional configuration. Unlike traditional scenarios, if a GitHub user deletes their `github.io` repository, an attacker does not need to compromise or reclaim the subdomain. Instead, an attacker, Alice, can exploit this misconfiguration by creating a new GitHub repository named `alice.github.io`. Alice can complete the takeover by creating a CNAME file within their `alice.github.io` repository that claims `bobby.cloud.com`, effectively gaining control of the vulnerable canonical subdomain. Critically, even though the CNAME record `bobby.cloud.com` still points to the deleted `bob.github.io` page, subsequent requests to `bobby.cloud.com` will now serve the content hosted at `alice.github.io` *without Alice having to compromise the original `bob.github.io` resource*. She can then serve any content of her choice, potentially injecting malicious scripts, phishing pages, or other harmful content into what appears to be Bob's legitimate website.

III. RELATED WORK

A. Open Source Security

The openness and accessibility of Open Source Software introduce significant security challenges. One major factor contributing to this issue is identity management, which complicates authorship verification. Despite the availability of mechanisms such as commit signing and push certificates [25], platforms such as GitHub allow developers to use aliases, leading to widespread identity obfuscation. Studies have identified more than 14.8 million author aliases, demonstrating the scale of this issue [26]. Furthermore, the risk of impersonation on GitHub, where users can easily masquerade as others when making commits, has been identified as a significant security issue [27]. Also, the role of less experienced developers in introducing security flaws. Research has shown that new contributors are disproportionately responsible for vulnerable

patches, accounting for more than 40% of reported vulnerabilities [28], [29]. Malicious activities further exacerbate OSS security risks. Attackers have exploited OSS ecosystems by introducing vulnerabilities disguised as minor fixes, commonly referred to as “hypocrite commits” [30]. These exploits, coupled with the increasing prevalence of supply chain attacks targeting OSS components, pose substantial threats to software integrity [31], [32].

B. Subdomain Takeover

A subdomain takeover occurs when an attacker gains control over a subdomain by exploiting misconfigured DNS records that point to decommissioned or unclaimed external services. This vulnerability often arises when organizations discontinue the use of a third-party service but neglect to remove or update the corresponding DNS entries, leaving the subdomain susceptible to takeover [33]. Once compromised, these subdomains can be used to host malicious content, intercept sensitive communications, or impersonate legitimate services, thereby undermining both user trust and organizational integrity [34].

Prior work and practitioner guidance discuss how trusted or long-lived domains can be abused as part of broader attack infrastructure [35], [33], [36]. For GitHub Pages, the platform-specific risk is compositional: a reclaimed custom domain supplies a trusted static staging surface, while browser-side code can load resources or call dynamic services hosted at visible external endpoints.

C. Botnet Communication via DNS

Botnet communication leveraging Domain Name System (DNS) protocols has become increasingly sophisticated, with threat actors exploiting DNS infrastructure to evade detection and maintain resilient command-and-control (C&C) channels [37], [38]. One prevalent technique is DNS tunneling, where botnets encapsulate malicious traffic within DNS queries or responses, enabling covert communication between infected devices and C&C servers [39], [40]. For example, malware families like ViperSoftX and DarkGate abuse DNS TXT records to retrieve encoded commands, such as URLs for downloading additional payloads, effectively bypassing traditional security filters that overlook DNS traffic scrutiny [41], [42]. Older domains are increasingly favored in modern botnet communication due to their inherent reliability and ability to evade detection mechanisms [43]. These domains leverage established DNS reputations and stable traffic patterns, avoiding the scrutiny faced by newly registered domains, which often trigger alerts due to irregular registration patterns, algorithmic generation, or sudden query spikes typical of fast-flux networks [44]. Security systems prioritize flagging newer domains with sparse histories or randomized naming conventions [45], [46], [47], [48].

IV. METHODOLOGY

Our work consists of three high-level components. First, we present an approach to detect canonical subdomains that are vulnerable to takeover through GitHub Pages. Second,

we conduct a controlled capability and boundary study using GitHub Pages resources. Third, we present a user study on the awareness and perception of subdomain takeover, with a particular focus on GitHub Pages.

A. Canonical Subdomain Takeover Detection on GitHub

We introduce a systematic approach to detecting vulnerable canonical subdomains on GitHub. Our approach identifies repositories with CNAME records pointing to a non-existent GitHub Pages Subdomain (e.g, `github.io`) while the mapped canonical subdomain in the CNAME record remains active. The primary challenge lies in detecting repositories that previously hosted a GitHub Pages subdomain but have since been deleted. To address this, we used historical version control data on GitHub repositories. Specifically, we leveraged the World of Code (WoC)¹ that includes version control data from more than 90 million public Git repositories [49].

1) *CNAME File Detection within GitHub Pages Subdomain*: To identify subdomains susceptible to takeover, we analyzed commits that modify the CNAME file within GitHub repositories. The CNAME file specifies canonical subdomain configurations for GitHub Pages by recording the external subdomain associated with the repository. Using WoC’s commit-to-file mapping command (i.e., `c2f`), which links each commit’s SHA-1 hash to the files it altered, we can efficiently locate commits involving the CNAME file. By querying this dataset for such commit logs, we extracted about 3 million commits for further investigation.

We then performed a data cleaning step to remove incomplete, irrelevant, or malformed commit records by: (1) Removing duplicate entries; (2) Filtering out commits that do not contain a valid repository reference; (3) Excluding commits that lack a recognizable GitHub username.

2) *Verify Domain Name in CNAME Records*: Next we used the collected commit SHA to retrieve the content of the commit in the CNAME file in the WoC, allowing us to extract the set of canonical subdomains. To determine whether a canonical subdomain remains active, we verified its status against DNS records. Specifically, we used the `dig` command in Linux to check if the domain currently has a valid CNAME record. This ensures that we only consider subdomains that are actively pointing to a GitHub Pages hosting service at present, rather than those that did so in the past.

3) *Identify Conditions for Takeover*: To determine whether a subdomain is at risk of takeover, we performed the following checks: (1) Sending HTTP requests to each subdomain; (2) Analyzing response codes to identify inactive or misconfigured domains. A return code of 404 indicates that the GitHub Pages subdomain no longer exists; (3) Checking if the subdomain points to a deleted or unclaimed GitHub repository. A subdomain is classified as vulnerable if it remains accessible but lacks valid content, while the associated GitHub repository has been removed.

¹<https://worldofcode.org/>

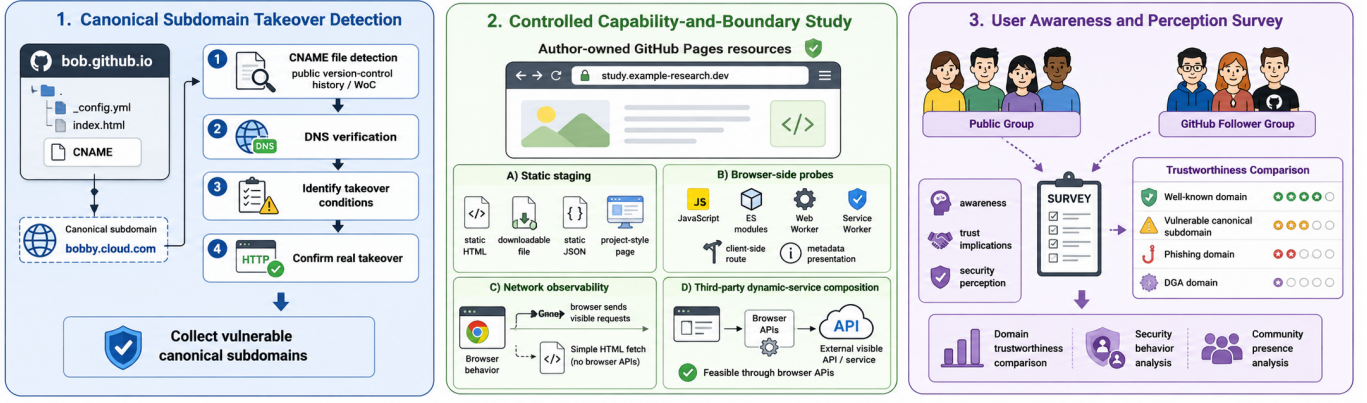


Fig. 2: Overview of the three methodology components: canonical subdomain takeover detection, a controlled capability-and-boundary study using author-owned GitHub Pages resources, and a user survey on awareness and trust perception.

4) *Confirm Real Takeover*: Even though we filtered out canonical subdomains that meet the requirements for takeover based on the collected data, there may still be other issues. For example, the subdomain might no longer be valid, or the owner has the subdomain protection. Therefore, we still need to test each subdomain to ensure it can actually be taken over. We attempted to claim and serve content on the canonical subdomain. Once a subdomain has been successfully taken over, we deploy a controlled proof of concept (PoC) to confirm the takeover. All the while, minimizing disruption following the common approach in vulnerable subdomain verification [50], [51]. This approach ensures ethical disclosure by avoiding alterations to the subdomain’s index page, which could otherwise damage the associated reputation.

To deploy an undisrupted PoC, we configured a new GitHub repository, and ensured that the potentially vulnerable canonical subdomain was correctly pointing to the repository. For verification, we constructed an HTML document and embedded a unique identifier, such as a code comment containing our research handle, within a randomly generated, non-indexed directory (e.g., /random-path.html). This prevents unintended visibility while still allowing us to demonstrate control over the subdomain. The takeover is confirmed by sending an HTTP request to the deployed file and verifying a response code of 200. After confirming a successful subdomain takeover, we compiled a list of vulnerable canonical subdomains. we collected relevant information of these subdomains from GitHub, including repository names, GitHub usernames, and other demographic details of the GitHub owner.

B. Controlled Capability and Boundary Study

We also examine what capabilities are practically enabled when a GitHub Pages custom domain becomes takeoverable. To avoid causing additional interaction with affected third-party domains, we conduct this study only in an author-controlled sandbox. The experiment uses GitHub Pages resources controlled by the authors, including a GitHub Pages origin domain and an author-owned custom domain configured for GitHub Pages.

We design a set of benign static pages that correspond to capabilities relevant to risk assessment: arbitrary static HTML, client-side JavaScript, static redirects, file hosting, project-style visual presentation, HTTPS availability, cross-origin resource loading, and browser-mediated external requests. Each test page contains a unique marker string so that the checker can verify that the expected sandbox content was served. The JavaScript tests are intentionally limited to local DOM updates, and the file-hosting test serves only a harmless text file. The mock project page contains no login fields, password fields, or submitting forms.

We implement automated checks for each capability. For static content, the checker issues HTTP requests and records the response status, final URL, MIME type, content hash when applicable, and whether the expected marker text is present. For browser-mediated behavior, the checker uses a headless browser when available to observe whether benign JavaScript executes, whether redirects remain within the author-controlled sandbox boundary, and whether external resources are requested from the separate author-owned origin. The checker records browser console messages, observed network endpoints, and whether cross-origin requests succeed or are blocked by browser policy. These observations allow us to distinguish content staged on the GitHub Pages domain from resources loaded from a separate endpoint.

If a capability succeeds in the sandbox, we report that the GitHub Pages hosting model supports that benign static capability under author-controlled conditions. We do not infer that every affected real-world domain supports every capability, because individual repositories, DNS settings, HTTPS configuration, browser policies, and lifecycle states may differ. RQ2 provides an empirical capability-and-boundary analysis of what a takeoverable GitHub Pages custom domain can enable under the same hosting model, while avoiding additional interaction with third-party affected domains.

C. Survey

To comprehensively assess the perception to subdomain takeovers, particularly on GitHub developers and the general public, we conducted two targeted surveys. The primary objec-

tive was to evaluate (1) awareness levels, (2) trust implications, and (3) the potential security risks associated with vulnerable canonical subdomains.

- **Survey of Followers of Vulnerable Canonical Subdomain Owners on GitHub (Follower Group):** To understand whether familiarity with a developer influences trust in their online presence, we surveyed GitHub users who followed owners of vulnerable canonical subdomains.
- **Survey of the General Population (Public Group):** To gauge the broader implications of subdomain takeovers beyond the GitHub ecosystem, we conducted a survey among the general public using Prolific.

1) *Protocol:* This study is approved by the *institution elided* IRB. We designed two surveys, one for the Public Group consisting of 15 questions and another for the Follower Group with 16 questions. Both surveys assess participants’ demographics, exposure history to vulnerable domains, general security practices, and perceptions of domain trustworthiness. However, the Follower group includes an additional question to determine whether they are more likely to trust a domain released by someone they follow. We reached out to Follower Group using their publicly available email addresses on GitHub following the approach outlined in a prior study [52], [53]. We recruited the Public Group via Prolific. As an incentive, participants in Public Group received a compensation of \$1. Participants in Follower Group recruited via email were entered into a random drawing, with one \$30 gift card awarded for every 30 participants. The complete survey is available in our release artifact.

2) *Data Analysis:* We enlisted 160 participants from Prolific for the Public Group. For the Follower Group, out of 4,628 email invitations sent for the survey, we received 128 valid responses, resulting in a 2.77% response rate. The response rate in our study is lower compared to previous work, likely because we relied on historical data from WoC, which only includes version control history up to 2022, rather than targeting actively engaged GitHub users as done in prior studies [54], [53]. We performed a comparative analysis on domain trustworthiness to evaluate participants’ ability to identify suspicious domains across two different population groups: the *follower group* and the *public group*. The evaluation focuses on four domain categories:

- 1) **Well-known Legitimate Domains (Known):** Well-known trustworthy domains to the public (e.g., `www.google.com`).
- 2) **DGA Domains (DGA):** Algorithmically generated domains (DGA) commonly associated with botnets. We conducted five DGA algorithms in our study: Necurs [55], Ranbyus [56], Symmi [57], Torpig [58], ZeusBot [59]. Using the date 2025-01-20 as input, we generated random domains through these DGA algorithms.
- 3) **Phishing Domains (Phishing):** Phishing domains documented in public GitHub repositories [60].
- 4) **Vulnerable Canonical Subdomains (Takeover):** Sub-

domains collected using our detection approach as vulnerable to takeover.

In the survey, participants were presented with two primary comparisons, each comprising three domain groups. Consequently, the first comparison included well-known domains, DGA-generated domains, and our detected subdomains, while the second comparison substituted DGA domains with GitHub-published phishing domains, retaining the well-known and detected subdomain categories. Within each comparison, three pairwise evaluations were performed to assess participants’ differentiation between every possible pair of domain groups (e.g., well-known vs. DGA, DGA vs. detected subdomains, and well-known vs. detected subdomains in the first comparison). To ensure diversity, domains in each pairwise comparison were randomized from a domain pool containing all four categories.

V. RESULTS

A. RQ1: Detection at Scale of GitHub Pages Subdomain Takeover Risk

In our study, we propose an approach to identify subdomains that are vulnerable to takeover on GitHub by leveraging version control data from the WoC as in Section IV-A.

- 1) **CNAME File Detection within GitHub Pages:** We identified 3 million commits that contain modifications to CNAME files using WoC. These commits serve as the foundation for subsequent analysis.
- 2) **Verify Domain Name in CNAME Records:** Through DNS lookups, we identified 52,564 unique domains that currently have valid CNAME records. After filtering, we identified 19,632 subdomains associated with `github.io` repositories. These domains form the primary candidates for further security analysis.
- 3) **Identify Conditions for Takeover:** Our analysis flags 1,626 domains as potentially vulnerable. These domains meet the criteria for takeover based on their response patterns and hosting status.
- 4) **Confirm Real Takeover:** Our tests confirm that 284 domains can be successfully taken over. This demonstrates the severity of the issue and highlights the security risks associated with unclaimed or misconfigured GitHub-hosted subdomains.

Our analysis identified 284 canonical subdomains within GitHub repositories that are susceptible to takeover. Starting from 52,564 unique domains with currently valid DNS CNAME records, 19,632 (37.35%) were associated with `github.io` repositories. Among these, 1,626 (8.28%) satisfied our vulnerability preconditions and were flagged as potentially vulnerable, and we confirmed 284 (17.47% of flagged candidates; 1.45% of all GitHub Pages-linked domains) as concretely takeable at the time of measurement. In other words, while the end incidence is modest relative to all configured domains, the absolute number of confirmed cases is non-trivial given that each case represents a stable, reputable

TABLE I: Controlled capability and risk results on author-owned GitHub Pages resources.

Capability	Result	Capability interpretation	Risk enabled
Static HTML	Supported	Controls page content under the reclaimed hostname	Content substitution under a trusted project URL
Benign JavaScript	Supported	Runs browser-side logic from the trusted-looking origin	Browser-side decisions that plain HTTP fetches miss
Static redirect	Supported	Forwards users from the reclaimed hostname	Traffic diversion from stale project links
Benign file hosting	Supported	Hosts arbitrary static files under the project-facing name	Distribution of untrusted artifacts through a familiar domain
Project-style presentation	Supported	Presents a realistic project page to visitors	Project impersonation and trust-transfer risk
Static JSON endpoint	Supported	Serves machine-readable static configuration	Configuration or lure staging for browser-side code
ES module import	Supported	Loads modular browser-side code	Modular code composition from the reclaimed origin
Web Worker	Supported	Runs browser-side background computation	Background browser computation during visits
Service Worker	Supported	Registers origin-scoped browser behavior for visitors	Persistent origin-scoped behavior for repeat visits
Client-side route	Supported	Presents SPA-style views under controlled URLs	Multi-view project surface without server routing
Metadata presentation	Supported	Controls title and preview metadata	Link preview and search-result trust manipulation
Cross-origin resources	Partial	Loads visible external resources from browser visits	External resource staging from a trusted landing page
Third-party dynamic API staging	Feasible	Browser code can call visible external APIs when policy allows	Adaptive browser-side behavior via external services
Network observability	Observed	Browser telemetry exposes the external endpoints used	Need for browser-based detection beyond static fetches

project-facing entry point that can silently change ownership due to configuration drift.

To understand whether this drift disproportionately affects “long-tail” accounts or also impacts visible maintainers, we examined owner demographics for the affected repositories. Our findings indicate that many of these owners maintain a significant online presence. On average, affected repository owners have 43.17 followers, with the maximum reaching 854. Additionally, 15 owners have more than 50 followers, 6 owners exceed 100 followers, and 5 owners have over 200. Similarly, in terms of public repositories, the average count among affected users is 43.73, with the most active owner maintaining 303 public repositories. In total, 31 owners have more than 50 repositories, while 10 owners exceed 100 and 2 surpass 200. These numbers suggest that some affected users are well-established within the developer community, which can amplify the downstream impact of a takeover when their domains are referenced across READMEs, package metadata, tutorials, and issue reports.

B. RQ2: Controlled Capability and Risk Study for Takeoverable Custom Domains

Table I summarizes the controlled run. The custom-domain sandbox served static HTML, a static JSON endpoint, a benign downloadable text file, a project-style static mockup, and an HTTPS check page. Browser automation observed benign client-side JavaScript execution, ES module import, Web Worker execution, Service Worker registration, client-side hash-route rendering, static metadata presentation, and a static redirect that remained within the controlled sandbox. These results show that a reclaimed GitHub Pages custom domain can present a realistic static project site with browser-side computation and origin-scoped browser behavior. Cross-origin tests from the custom-domain page to the separate author-owned GitHub pages subdomain origin showed that browser visits can load visible external resources: CSS, image loading, and a static redirect/link completed in this configuration, and browser network logs recorded the external origin explicitly. This is the same web mechanism that makes third-party dynamic API staging feasible: a static page from the reclaimed

domain can call an external HTTPS API, WebSocket, server-sent-event endpoint, or hosted script when browser policy and the external service allow it.

The third-party dynamic-service design analysis strengthens the risk claim without requiring a live external-backend experiment. A reclaimed Pages domain can deliver the initial HTML and JavaScript under the trusted project-facing hostname; that JavaScript can then request dynamic data or instructions from a separately hosted API. CORS determines whether the page can read a cross-origin response, but it does not prevent all external resource loading, redirects, script loading, images, iframes, WebSocket handshakes, or opaque browser requests. Thus the practical capability is a compositional one: static takeover staging plus visible external dynamic services can create adaptive browser-side behavior for visitors.

The risk is not limited to a fixed static page. A stable project-facing hostname can stage a rich browser-side application, register origin-scoped behavior for visited clients, and direct browsers to external dynamic services. In the network-observability experiment, a browser visiting a page with an external resource produced requests to both the custom domain and GitHub pages subdomain; a `curl` request fetched only the static HTML. This gap matters operationally because crawlers or validators that fetch only HTML may miss browser-mediated external requests. A real dynamic backend linked from a Pages-hosted page would be a separate visible endpoint, but the reclaimed custom domain still provides the trusted staging surface that initiates the interaction.

C. RQ3: Human Perception of GitHub Subdomain Takeover

In this research question, we examine perceptions of subdomain takeovers on GitHub through a survey conducted with two participant groups: the **Public Group** and the **Follower Group**. We present the findings on participants’ demographics and exposure profiles, their online behavior related to domain trustworthiness, and a comparison of domain trust perceptions between the two groups.

1) *Demographic and Exposure Profile:* We collected demographic information and assessed participants’ exposure to phishing and click fraud in both the Follower Group and

TABLE II: This table presents the responses from two participant groups Public Group and Follower Group on factors that make a domain suspicious, precautions taken before clicking an unfamiliar domain, and prior awareness of subdomain takeover.

Options to consider	Percentage of Answer	
	Public Group	Follower Group
WHAT MAKES YOU THINK A DOMAIN MIGHT BE SUSPICIOUS?		
A. Unfamiliar or rare domain endings	81.88%	74.22%
B. Bad reviews or complaints	72.5%	71.09%
C. Weird spelling or extra symbols	90%	95.31%
D. Recently created or very old domain	41.25%	37.5%
E. No secure connection (HTTPS)	73.75%	79.69%
WHAT DO YOU DO BEFORE CLICKING AN UNFAMILIAR DOMAIN FOR THE FIRST TIME?		
A. I only click if someone I trust shares it	20%	28.12%
B. I hover over the link to check the URL	55.62%	69.53%
C. I ignore and never click personal domains	50.62%	25%
D. I verify it using a website reputation checker	38.75%	35.94%
HAD YOU HEARD OF 'SUBDOMAIN TAKEOVER' BEFORE THIS SURVEY?		
Yes, I heard about someone gaining control of another's subdomain to publish content.	41.25%	46.09%

the Public Group. The data reveals significant differences between the two groups, particularly in technical engagement, education levels, and awareness of domain security issues. The follower group is highly active on GitHub, with 60.16% using it daily, whereas 37.5% of the public group has never used GitHub. Age distribution also differs, with the follower group skewing younger—42.19% are aged 25–34, compared to the public group, which is more evenly spread, including 11.25% aged 45–54 and 1.88% aged 65+. Gender distribution further reflects this distinction, as the follower group is predominantly male (88.28%), while the public group is more balanced, with 51.25% female and 45.0% male.

In terms of phishing and click fraud exposure, nearly 61% of the follower group encounters phishing at least a few times a month, with 30% experiencing it weekly. Similarly, in the public group, 64% report encountering phishing monthly or more often, with 27.5% facing it weekly. Notably, only 3.91% of the follower group and just 1.25% of the public group have never experienced phishing or click fraud. These findings underscore the widespread risk posed by malicious domains, including those arising from subdomain takeovers.

2) *Security Perceptions and Online Behavior on Domain trustworthiness*: In Table II, we present an analysis of security perceptions and online behavior between two groups: the Public Group (general population) and the Follower Group (GitHub users who follow developers with vulnerable canonical subdomains). The results are structured into three themes.

For factors that make a domain suspicious, both groups share similar concerns regarding domain suspiciousness, with weird spelling or extra symbols being the most frequently cited indicator (90% public, 95.3% followers). However, followers are more likely to consider no secure connection (HTTPS) a risk (79.69%) compared to the public (73.75%), indicating stronger security awareness. Conversely, the public is more concerned about unfamiliar domain endings (81.88%) than followers (74.22%).

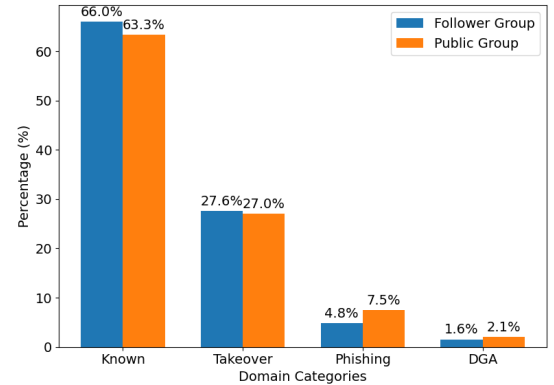


Fig. 3: The percentage of times each domain category was chosen as the more trustworthy option in pairwise comparisons by two groups: GitHub followers of developers with vulnerable canonical subdomains (Follower Group) and the general public (Public Group). The domain categories are outlined in Section IV-C2. The *Takeover* category represents vulnerable canonical subdomains and the *Known* category refers to well-known legitimate domains.

For precautions taken before clicking an unfamiliar domain, the Follower Group exhibits a more cautious approach overall. A higher percentage of followers only click links shared by trusted sources (28.12%) compared to the public (20%). They also check URLs more frequently (69.53% vs. 55.62%) than the public. Interestingly, the public is more likely to ignore and never click personal domains (50.62%) compared to followers (25%), suggesting that followers engage more with technical content. This suggests that GitHub users are more engaged with technical content and thus more prone to interacting with developer-owned domains, including potentially vulnerable canonical subdomains.

For prior awareness of subdomain takeover, awareness of subdomain takeover is higher among followers (46.09%) com-

TABLE III: The responses to the question “Click Likelihood: If you encountered this pair of domains (e.g., on social media or in an email), which one do you think is more trustworthy? Choose the one you trust more in each pair.”.

Pair Trustiness Comparison Which one is more Trustworthy	Public Group Percentage (n=160)		Follower Group Percentage (n=128)	
	Domain in A	Domain in B	Domain in A	Domain in B
COMPARISON WITH WELL-KNOWN LEGITIMATE DOMAINS				
A: Well-known Legitimate Domains				
B: Vulnerable Canonical Subdomains	 93.5%	 6.5%	 98.4%	 1.6%
COMPARISON WITH DGA DOMAINS				
A: Well-known legitimate domains				
B: DGA Domains	 96.9%	 3.1%	 100%	 0%
A: Vulnerable Canonical Subdomains				
B: DGA Domains	 89.4%	 10.6%	 90.6%	 9.4%
COMPARISON WITH PHISHING DOMAINS				
A: Well-known Legitimate Domains				
B: Phishing Domains	 93.8%	 6.2%	 99.2%	 0.8%
A: Vulnerable Canonical Subdomains				
B: Phishing Domains	 60%	 40%	 71.8%	 28.2%

pared to the general public (41.25%), though the gap is not substantial. This suggests that even among technically engaged users, knowledge of this specific threat remains limited.

We also assess whether participants in the follower group are more likely to trust and click on links associated with users they follow on GitHub by analyzing their responses on a 5-point Likert scale, ranging from 1 (Not at all) to 5 (Absolutely). A majority (64.84%) of respondents (scores 4 and 5) are inclined to trust domains associated with GitHub users they follow. This highlights a critical security risk: subdomain takeovers of domains linked to GitHub repositories can be particularly effective in phishing or exploitation, as followers are more likely to trust and click links associated with developers they recognize.

3) *Domain Trustworthiness Comparison:* In table III, we present survey responses assessing user trust perception of domains with different backgrounds, particularly focusing on vulnerable canonical subdomains compared to well-known legitimate domains, DGA (Domain Generation Algorithm) domains, and phishing domains as in Section IV-C2. We observed that both the public group and follower group overwhelmingly trust well-known legitimate domains more than domains vulnerable to takeover, with preferences of 93.5% vs. 6.5% and 97.7% vs. 2.3%, respectively. This trend remains consistent when comparing well-known legitimate domains with DGA domains and phishing domains, with near-total trust in the well-known domains (100% in the follower group).

Both groups strongly prefer well-known legitimate domains over vulnerable canonical subdomains. When comparing domains vulnerable to takeover with phishing domains, 71.8% of followers trusted the vulnerable domains over phishing, compared to 60% of the public group. The followers exhibit a dual effect: while they show more trust in vulnerable canonical subdomain, due to familiarity with developers, they also demonstrate better ability to distinguish DGA and phishing

threats.

The Table III indicates that both the general public and GitHub follower groups perceive vulnerable canonical subdomains from GitHub as more trustworthy than DGA and phishing domains. However, GitHub followers, despite being more security-aware, still exhibit a higher level of trust in vulnerable canonical subdomains compared to the public group. This suggests that familiarity with developers may lead to a reduced perception of risk associated with subdomain takeovers.

Also in Figure 3, we present how often each category was chosen as the more trustworthy option in all pairwise comparisons. The majority of trust votes were given to known legitimate domains, with 66% of the follower group and 63.3% of the public group selecting them in their comparisons. 27.6% of the follower group and 27% of the public group selected vulnerable canonical subdomains as more trustworthy in their comparisons. Despite the known risks of subdomain takeovers, these domains were still chosen significantly more often than phishing or DGA domains. GitHub followers are better at recognizing phishing and DGA threats, whereas the general public is more prone to misjudging these malicious domains.

VI. DISCUSSION AND IMPLICATION

A. GitHub’s Stance on Subdomain Takeover

GitHub’s documentation explains that creating a TXT record with your domain hosting service for domain verification is an optional security measure for GitHub Pages, leaving DNS management entirely up to the user [61]. This verification process ensures that only repositories tied to your personal account or organization can publish to your verified custom domain and its immediate subdomains.

However, GitHub does not enforce these security measures—it simply provides tools you may choose to use. This means that maintaining proper DNS hygiene is your responsibility. For example, if your GitHub Pages site is discontinued

or deleted, the user must promptly remove or update any related DNS records to prevent potential subdomain takeover. Moreover, there aren't any built-in preventive measures to automatically monitor or update your DNS settings, so regular manual audits are essential to ensure that obsolete DNS entries are cleared.

While the guide in setting up custom domain with GitHub Pages emphasizes user responsibility, it also gives a misleading impression of security [61]. Specifically, GitHub's documentation reassures users that verifying `example.com` prevents attackers from taking over `a.example.com`. However, GitHub simultaneously warns against wildcard DNS records (`*.example.com`) because they expose subdomains to immediate takeover risks, even if `example.com` is verified. This contradiction creates a false sense of security: while users may assume that setting up `a.example.com` eliminates risks of subdomain takeover.

B. Mitigation for Subdomain Takeover on GitHub

Improvements GitHub Can Implement: (1) Automated DNS Monitoring: GitHub could integrate DNS monitoring tools directly into the GitHub Pages dashboard. Such a feature would automatically detect orphaned or misconfigured DNS records, alerting users to discrepancies between their repository settings and their DNS configurations. (2) Enforced Domain Management Policies: GitHub might consider implementing policies that automatically flag or even temporarily disable custom domains if associated repositories are deleted or if there is a significant change in the account's status (e.g., billing downgrades). (3) Fallback or Holding Page Options: GitHub could offer a built-in fallback page or redirection service that activates when a custom domain is no longer linked to an active repository. This would minimize the window during which an unclaimed subdomain is vulnerable to takeover.

Best Practices for Users: (1) Integrate Automated Audits: Incorporate DNS checks into your CI/CD pipelines or periodic audit routines to verify that DNS entries match your current GitHub Pages configuration. This automation can catch potential issues faster than manual reviews alone. (2) Regular DNS Hygiene: Make it a routine practice to review and update DNS configurations, ensuring that obsolete records are removed promptly to minimize any residual risk.

VII. LIMITATION

Subdomain Takeover Detection: Our detection pipeline relies on historical GitHub data from WoC. As a result, a vulnerable canonical subdomain can only be detected if the associated repository history and its custom-domain artifacts are present in WoC. This dependency may introduce coverage gaps if relevant repositories are missing, mirrored incompletely, or appear only partially (e.g., truncated histories). While these constraints are inherent to large-scale archival datasets, our approach is not specific to WoC; it can be generalized to other version-control corpora or directly to GitHub archival pipelines that preserve repository history and file contents.

Participant Representation: Our public-group participants were recruited through Prolific, which may over-represent individuals with stronger technical literacy than the general population. To increase diversity, we recruited participants from varied occupational backgrounds, including Healthcare (5.62%), Business/Management (24.38%), Service Industry (9.38%), Technology/IT (22.5%), Unemployed (8.12%), Creative Arts (4.38%), Trades (3.75%), Retired (2.5%), and Students (11.88%). Nonetheless, survey responses are self-reported and may be influenced by participants' guessing strategies, prior familiarity with GitHub, or the framing effects of the survey.

VIII. ETHICAL CLARIFICATION

Our research methodology adhered to strict ethical guidelines and institutional review board (IRB) approval. The process for confirming potential subdomain takeover vulnerabilities, as detailed in Section IV-A4, was designed for verification purposes only and did not involve disruptive actions. We employed a controlled, non-intrusive Proof-of-Concept, consisting of a unique, non-indexed HTML file within a randomly generated directory, to confirm takeover feasibility without altering existing content or services. This approach aligns with common practices in subdomain takeover detection.

IX. CONCLUSION

Subdomain takeover in GitHub Pages is a critical yet often overlooked software maintenance failure mode in modern web projects. Our study presents the first systematic investigation of subdomain takeovers on GitHub, a weaker adversary model where attackers can seize control of custom subdomains by creating new, unrelated GitHub Pages sites, rather than needing to compromise the original target. This platform-specific mechanism lowers the barrier for attackers and underscores an often-overlooked risk vector in widely-used developer services.

We demonstrate that subdomain takeovers can be effectively detected using large-scale version control history data. We examined the capability and risk of takeoverable GitHub Pages custom domains using author-owned resources. The experiment shows support for static staging, client-side routing redirects, file hosting, and visible browser-mediated external requests. Design analysis further shows that a reclaimed static Pages domain can be composed with third-party dynamic services through ordinary browser APIs, allowing adaptive browser-side behavior while the external service remains a visible endpoint. Our findings reveal that users perceive vulnerable canonical subdomains as more trustworthy than phishing or DGA domains, increasing the risk of exploitation. GitHub followers, in particular, exhibit higher trust in compromised subdomains. This underscores the need for better security awareness in the developer community.

DATA AVAILABILITY

We provide a supporting artifact for this paper at: <https://doi.org/10.5281/zenodo.19336012>.

REFERENCES

- [1] R. Naraine, "Open-source proftpd hacked, backdoor planted in source code," 2010.
- [2] GitHub, "Types of github accounts," 2025, accessed: 2025-02-26. [Online]. Available: <https://docs.github.com/en/get-started/learning-about-github/types-of-github-accounts>
- [3] GitHub, Inc., "Github pages," 2025, accessed: 2025-01-21. [Online]. Available: <https://pages.github.com/>
- [4] —, "Github pages rest api documentation," 2025, accessed: 2025-01-21. [Online]. Available: <https://docs.github.com/en/rest/pages?apiVersion=2022-11-28>
- [5] BuiltWith, "Github usage statistics," 2025, accessed: 2025-02-25. [Online]. Available: <https://trends.builtwith.com/link/GitHub>
- [6] Y. Wang, Z. Li, T. Wu, I. Duncan, and Q. Lyu, "An empirical study: automated subdomain takeover threat detection," in *2021 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*. IEEE, 2021, pp. 1–10.
- [7] S. Z. U. Rashid, M. I. Kamrul, and A. Islam, "Understanding the security threats of esoteric subdomain takeover and prevention scheme," in *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*. IEEE, 2019, pp. 1–4.
- [8] P. Hudak, "Subdomain takeover: Yet another starbucks case," 2018, accessed: 2025-02-27. [Online]. Available: <https://0xpatrik.com/subdomain-takeover-starbucks-ii/>
- [9] T. Wrinn, "Subdomain takeover on open.itu.edu via shopify," 2023, accessed: 2025-02-27. [Online]. Available: <https://infosecwriteups.com/subdomain-takeover-on-open-itu-edu-via-shopify-6b83ea970f3d>
- [10] A. Swinnen, "Authentication bypass on uber's single sign-on via subdomain takeover," 2017, accessed: 2025-02-26. [Online]. Available: <https://www.arneswinnen.net/2017/06/authentication-bypass-on-ubers-ssso-via-subdomain-takeover/>
- [11] A. S. M. Venigalla and S. Chimalakonda, "An empirical study on correlation between readme content and project popularity," *arXiv preprint arXiv:2206.10772*, 2022.
- [12] L. Li, Z. Ren, X. Li, W. Zou, and H. Jiang, "How are issue units linked? empirical study on the linking behavior in github," in *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2018.
- [13] M. Lima, I. Steinmacher, D. Ford, G. Vorreuter, L. Gonçalves, T. Conte, and B. Gadelha, "How are discussions linked? a link analysis study on github discussions," *Journal of Systems and Software*, vol. 219, p. 112196, 2025.
- [14] H. Hata, C. Treude, R. G. Kula, and T. Ishio, "9.6 million links in source code comments: Purpose, evolution, and decay," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE/ACM, 2019.
- [15] T. Xiao, S. Baltes, H. Hata, C. Treude, R. G. Kula, T. Ishio, and K. Matsumoto, "18 million links in commit messages: purpose, evolution, and decay," *Empirical Software Engineering*, 2023.
- [16] Microsoft, "Prevent dangling dns entries and avoid subdomain takeover," Microsoft Learn (Azure Security Fundamentals), 2026, accessed: 2026-01-26. [Online]. Available: <https://learn.microsoft.com/en-us/azure/security/fundamentals/subdomain-takeover>
- [17] Palo Alto Networks, "What is dangling dns?" Palo Alto Networks Cyberpedia, 2026, accessed: 2026-01-26. [Online]. Available: <https://www.paloaltonetworks.com/cyberpedia/what-is-a-dangling-dns>
- [18] UNICRI, "Clicks, links & tricks, oh my! how serious organized criminals exploit digital trust pathways," United Nations Interregional Crime and Justice Research Institute (UNICRI), Tech. Rep., Oct. 2025, accessed: 2026-01-26. [Online]. Available: <https://unicri.org/sites/default/files/2025-10/Clicks-Links-Tricks-How-Serious-Organized-Criminals-Exploit-Digital-Trust-Pathways-Oct-2025.pdf>
- [19] Infoblox Threat Intel, "Cloudy with a chance of hijacking: Forgotten dns records enable scam actor," Infoblox Blog, May 2025, accessed: 2026-01-26. [Online]. Available: <https://www.infoblox.com/blog/threat-intelligence/cloudy-with-a-chance-of-hijacking-forgotten-dns-records-enable-scam-actor/>
- [20] ProjectDiscovery, "Subfinder: Fast passive subdomain enumeration tool," <https://github.com/projectdiscovery/subfinder>, 2025, accessed: 2025-02-02.
- [21] haccr, "Subjack: Subdomain takeover tool," <https://github.com/haccr/subjack>, 2018, accessed: 2025-02-02.
- [22] O. A. Project, "Amass: In-depth attack surface mapping and asset discovery," <https://github.com/owasp-amass/amass>, 2025, accessed: 2025-02-02.
- [23] A. Rahman, R. Mahdavi-Hezaveh, and L. Williams, "A systematic mapping study of infrastructure as code research," *Information and Software Technology*, vol. 108, pp. 65–77, 2019.
- [24] A. Weiss, A. Guha, and Y. Brun, "Tortoise: Interactive system configuration repair," in *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2017, pp. 625–636.
- [25] "Signing commits." [Online]. Available: <https://ghdocs-prod.azurewebsites.net/en/authentication/managing-commit-signature-verification/signing-commits>
- [26] T. Fry, T. Dey, A. Karnauch, and A. Mockus, "A Dataset and an Approach for Identity Resolution of 38 Million Author IDs extracted from 2B Git Commits," in *Proceedings of the 17th International Conference on Mining Software Repositories*, ser. MSR '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 518–522. [Online]. Available: <https://dl.acm.org/doi/10.1145/3379597.3387500>
- [27] Y. Zhang, A. Liang, X. Wang, P. Wisniewski, F. Zhang, K. Leach, and Y. Huang, "Who's pushing the code? an exploration of github impersonation," in *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 2025, pp. 602–602.
- [28] A. Bosu, J. C. Carver, M. Hafiz, P. Hilley, and D. Janni, "Identifying the characteristics of vulnerable code changes: An empirical study," in *Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering*, 2014, pp. 257–268.
- [29] A. Meneely, H. Srinivasan, A. Musa, A. R. Tejeda, M. Mokary, and B. Spates, "When a patch goes bad: Exploring the properties of vulnerability-contributing commits," in *2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. IEEE, 2013, pp. 65–74.
- [30] Q. Wu and K. Lu, "On the feasibility of stealthily introducing vulnerabilities in open-source software via hypocrite commits," *Proc. Oakland*, 2021.
- [31] P. Ladisa, H. Plate, M. Martinez, and O. Barais, "Sok: Taxonomy of attacks on open-source software supply chains," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 1509–1526.
- [32] M. Ohm, H. Plate, A. Sykosch, and M. Meier, "Backstabber's Knife Collection: A Review of Open Source Software Supply Chain Attacks," *Detection of Intrusions and Malware, and Vulnerability Assessment*, vol. 12223, pp. 23–43, Jun. 2020.
- [33] T. Dai, P. Jeitner, H. Shulman, and M. Waidner, "The hijackers guide to the galaxy: {Off-Path} taking over internet resources," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 3147–3164.
- [34] M. Squarcina, M. Tempesta, L. Veronese, S. Calzavara, and M. Maffei, "Can i take your subdomain? exploring {Same-Site} attacks in the modern web," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2917–2934.
- [35] Y. Xing, H. Shu, H. Zhao, D. Li, and L. Guo, "Survey on botnet detection techniques: Classification, methods, and evaluation," *Mathematical Problems in Engineering*, vol. 2021, no. 1, p. 6640499, 2021.
- [36] M. Zhang, X. Li, B. Liu, J. Lu, Y. Zhang, J. Chen, H. Duan, S. Hao, and X. Zheng, "Detecting and measuring security risks of hosting-based dangling domains," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 7, no. 1, pp. 1–28, 2023.
- [37] K. Alieyan, A. Almomani, A. Manasrah, and M. M. Kadhum, "A survey of botnet detection based on dns," *Neural Computing and Applications*, vol. 28, pp. 1541–1558, 2017.
- [38] M. Singh, M. Singh, and S. Kaur, "Issues and challenges in dns based botnet detection: A survey," *Computers & Security*, vol. 86, pp. 28–52, 2020, accessed: 2025-02-02.
- [39] Y. Wang, A. Zhou, S. Liao, R. Zheng, R. Hu, and L. Zhang, "A comprehensive survey on dns tunnel detection," *Computer Networks*, vol. 197, p. 108322, 2021.
- [40] N. M. Hands, B. Yang, and R. A. Hansen, "A study on botnets utilizing dns," in *Proceedings of the 4th Annual ACM Conference on Research in Information Technology*, 2015, pp. 23–28.
- [41] J. Rubin, "Complex obfuscation? meh." 2024, accessed: 2025-02-16. [Online]. Available: <https://decoded.avast.io/janrubin/complex-obfuscation-meh/>
- [42] —, "Vipersoftx: Hiding in system logs and spreading venomsoftx," 2024, accessed: 2025-02-

16. [Online]. Available: <https://decoded.avast.io/janrubin/vipersoftx-hiding-in-system-logs-and-spreading-venomsoftx/>
- [43] J. So, N. Miramirkhani, M. Ferdman, and N. Nikiforakis, "Domains do change their spots: Quantifying potential abuse of residual trust," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 2130–2144.
- [44] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon, "From {Throw-Away} traffic to bots: Detecting the rise of {DGA-Based} malware," in *21st USENIX Security Symposium (USENIX Security 12)*, 2012, pp. 491–506.
- [45] S. Al-Mashhadi, M. Anbar, S. Karuppayah, and A. K. Al-Ani, "A review of botnet detection approaches based on dns traffic analysis," *Intelligent and Interactive Computing: Proceedings of IIC 2018*, pp. 305–321, 2019.
- [46] Hackers Terminal, "Dns tunneling: What it is & how it works," 2024, accessed: 2025-02-16. [Online]. Available: <https://hackerterminal.com/dns-tunneling/>
- [47] H. Li, Z. Li, S. Zhang, and X. Pu, "Malicious dns detection by combining improved transformer and cnn," *Scientific Reports*, vol. 14, no. 1, pp. 1–16, 2024.
- [48] G. Cluley, "Block newly-registered domains to reduce security threats in your organisation," 2019, accessed: 2025-02-27. [Online]. Available: <https://www.tripwire.com/state-of-security/block-newly-registered-domains-to-reduce-security-threats-in-your-organisation>
- [49] Y. Ma, C. Bogart, S. Amreen, R. Zaretski, and A. Mockus, "World of code: an infrastructure for mining the universe of open source vcs data," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 2019, pp. 143–154.
- [50] HackerOne, "Guide to subdomain takeovers 2.0," 2025, accessed: 2025-02-22. [Online]. Available: <https://www.hackerone.com/blog/guide-subdomain-takeovers-20>
- [51] EdOverflow, "Can I take over XYZ?" 2025, accessed: 2025-02-22. [Online]. Available: <https://github.com/EdOverflow/can-i-take-over-xyz>
- [52] Y. Huang, D. Ford, and T. Zimmermann, "Leaving my fingerprints: Motivations and challenges of contributing to oss for social good," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 1020–1032.
- [53] Y. Zhang, A. Liang, X. Wang, P. Wisniewski, F. Zhang, K. Leach, and Y. Huang, "Who's pushing the code? an exploration of github impersonation," in *2025 IEEE/ACM 47rd International Conference on Software Engineering (ICSE)*, 2025, pp. 3–15.
- [54] Z. Fang, M. Endres, T. Zimmermann, D. Ford, W. Weimer, K. Leach, and Y. Huang, "A four-year study of student contributions to oss vs. oss4sg with a lightweight intervention," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2023, pp. 3–15.
- [55] Wikipedia contributors, "Necurs botnet," https://en.wikipedia.org/wiki/Necurs_botnet, 2024, accessed: 2024-02-04.
- [56] Federal Office for Information Security (BSI), "Ranbyus malware - avalanche botnet," 2024, accessed: 2024-02-04. [Online]. Available: <https://www.bsi.bund.de/EN/Themen/Verbraucherinnen-und-Verbraucher/Cyber-Sicherheitslage/Methoden-der-Cyber-Kriminalitaet/Botnetze/Botnetz-Avalanche/Malware/Ranbyus/Ranbyus.html>
- [57] FortiGuard Labs, "Fortiguards ips encyclopedia entry - id 42310," 2024, accessed: 2024-02-04. [Online]. Available: <https://www.fortiguards.com/encyclopedia/ips/42310>
- [58] Wikipedia contributors, "Torpig," <https://en.wikipedia.org/wiki/Torpig>, 2024, accessed: 2024-02-04.
- [59] —, "Zeus (malware)," [https://en.wikipedia.org/wiki/Zeus_\(malware\)](https://en.wikipedia.org/wiki/Zeus_(malware)), 2024, accessed: 2024-02-04.
- [60] N. Schunk, "Discord phishing links repository," <https://github.com/nikolaischunk/discord-phishing-links/tree/main>, 2024, accessed: 2024-02-04.
- [61] GitHub, "Verifying your custom domain for github pages," 2025, accessed: 2025-02-27. [Online]. Available: <https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/verifying-your-custom-domain-for-github-pages>