Sri Lanka Institute of Information Technology

IE2062

Web Security

Bug Bounty Report IV

Submitted by:

Student Registration Number	Student Name
IT21197550	Nihila Premakanthan

Date of Submission: 28.05.2023

Acknowledgement

I would like to express my special thanks to our mentor Ms. Chethana Liyanapathirana and Dr. Lakmal Rupansighe for their time and efforts she provided throughout the course, and for the Web Security lecture panel for guiding us through this semester and for helping us by giving examples, guidelines, and advice about the project. Your useful advice and suggestions were really helpful to me during the project's completion. In this aspect, I am eternally grateful to you.

Executive Summary

This report aims to provide an overview of the vulnerability identified in a particular domain. The bug bounty platform called Hackerone was used for this purpose. This report analyses the domain of Doppler (https://www.doppler.com/).

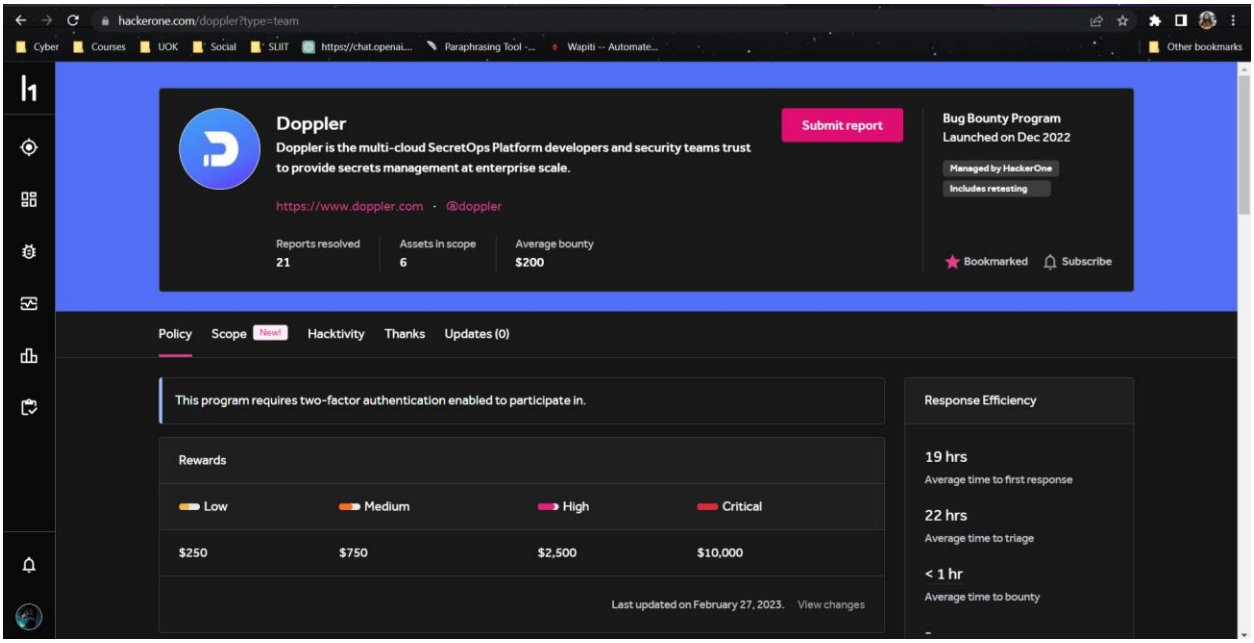
This report uses different tools to gather information detect vulnerabilities and perform penetration testing. The tool name Netsparker and Owasp Zap was mainly used to identify the vulnerability. Further this report provides the vulnerability title, vulnerability description, Affected Components, Impact Assessment, Steps to reproduce the vulnerability, proof of concept and the proposed mitigation.

By including these comprehensive details for each vulnerability, the report provides a comprehensive overview of the security weaknesses present within the system and offers actionable insights for remediation and improvement.

Contents

Introduction

Affected URL: https://www.doppler.com/



Vulnerability title

Source Code Disclosure

Severity Level

Risk Level:
MEDIUM

Vulnerability Description

The term "source code disclosure" describes the unintentional release or illegal access of a software application's source code, which is the version of the program that can be read by humans and is created by programmers using programming languages.

Source code leaks can happen for a number of different causes, such as unintentional disclosure, malicious assaults, or insider threats. When source code is accidentally released or made publicly accessible as a result of configuration errors or human error, this is known as accidental disclosure. Hackers or online criminals launch malicious assaults against software repositories or development servers in an effort to acquire the source code without authorization. When employees or contractors with authorized access mistakenly or intentionally leak the code to outside parties, this is known as an insider threat.

Affected components

The affected components of Source Code Disclosure can include:

- Source code repositories: If the source code is kept in version control programs like Git, SVN, or Mercurial, an unauthorized disclosure could make all of the code's branches, tags, or commits publicly available.
- Web servers or other hosting environments: If the source code is stored on web servers or cloud computing platforms, errors or security flaws in these environments could allow unauthorized access to the source code files.
- Servers used for development or staging: These servers are frequently used for the deployment and testing of source code. The source code, including any unpublished or ongoing code changes, may be stored on these servers. The source code may be made available if these servers are compromised.
- Establish CI/CD pipelines or servers: Software application compilation, testing, and deployment are handled by build servers and Continuous Integration/Continuous Deployment (CI/CD) pipelines. The source code can be available during the build or deployment processes if these systems are compromised.
- Applications or software packages: Installable files or packages are sometimes used to distribute software applications. The source code may be made public if it is contained in these packages or if the packages are made available.
- Shared or distributed code: Shared or distributed code libraries, frameworks, or modules are sometimes used in software development. Several programs or systems that depend on these shared components may be affected if the source code for them is made public.
- Third-party components that have been incorporated: Software programs frequently use libraries, APIs, or services from outside sources. If the source code for these components is made public, the integration details may be exposed, possibly disclosing sensitive data or vulnerabilities.

Impact assessment

The process of determining the potential effects and dangers of exposing or disclosing a software application's source code without authorization is known as the impact assessment of source code disclosure. When evaluating the impact, keep the following important factors in mind:

- Risk related to intellectual property (IP): Evaluate the worth and sensitivity of the source code in terms of IP. Think about the possible effects on the organization's or software's financial worth, market position, and competitiveness if the source code is compromised. This evaluation may take into account the code's originality, secret algorithms, trade secrets, or any patented features.
- Security Implications: Consider the security ramifications of releasing source code. Analyze the likelihood that attackers may use the exposed code's flaws or vulnerabilities to gain access without authorization, compromise sensitive data, or compromise other systems. Take into account the effects on the software's availability, confidentiality, and integrity.
- Reputational Damage: Consider the potential reputational harm that could result from the source code's exposure. Analyze the effects on consumer trust, brand reputation, and how the software or company is viewed in the sector. Analyze the possible loss of clients, collaborators, or commercial possibilities brought on by the erosion of confidence.

Steps to Reproduce

The procedures to replicate the source code disclosure vulnerability can change depending on the target system and the particular set of conditions. However, the following generic procedures can help to clarify the idea:

- Choose the software program or system you want to provide the source code for. This could be any kind of software system, including a web application or a mobile app.
- Gather information on the target by doing reconnaissance, including identifying the technology stack, programming languages, and any known flaws or vulnerabilities in the software.
- Search for possible points of entry or attack that might give you access to the source code. This can involve improperly configured servers, openly accessible code repositories, or weak software parts.

- Attempt to exploit misconfigurations to get unauthorized access if you come across misconfigured servers or repositories. This can entail using weak or default passwords, taking advantage of software flaws known to exist, or abusing insecure access controls.
- Attacks that are specifically targeted: Try to exploit software flaws like SQL injection, remote file inclusion, or directory traversal if you have found them in order to get the source code. This can entail creating particular requests or inputs to deceive the program into disclosing the source code.
- If you have insider access or privileges, you can duplicate the vulnerability by intentionally or unintentionally releasing the source code. This could entail transferring the code to a different location or sharing it with unauthorized users.
- Once you have successfully accessed the source code, you should record the actions you performed, the vulnerabilities you exploited, and the scope of the exposure. This data may be helpful for analysis, reporting, or corrective action.

Proposed mitigation

Implementing strong security procedures is essential to reducing the dangers connected with source code leak.

- Access controls: Grant only authorized people access to development environments and source code repositories. Put in place reliable authentication procedures and routinely check access rights.
- Secure transmission and encryption: When storing or transmitting source code, use encryption to prevent unauthorized access. When exchanging or disseminating code, use safe methods like safe File Transfer Protocol (SFTP) or encrypted connections (like HTTPS).
- Tests and reviews for security: To find and fix potential flaws or vulnerabilities in the source code, do regular security assessments, code reviews, and vulnerability scans.
- Monitoring systems and incident response processes should be put in place in order to quickly identify and address any illegal access to or publication of source code.

Proof of concept

1. [Possible] Source Code Disclosure (Generic)

MEDIUM | 1

Request

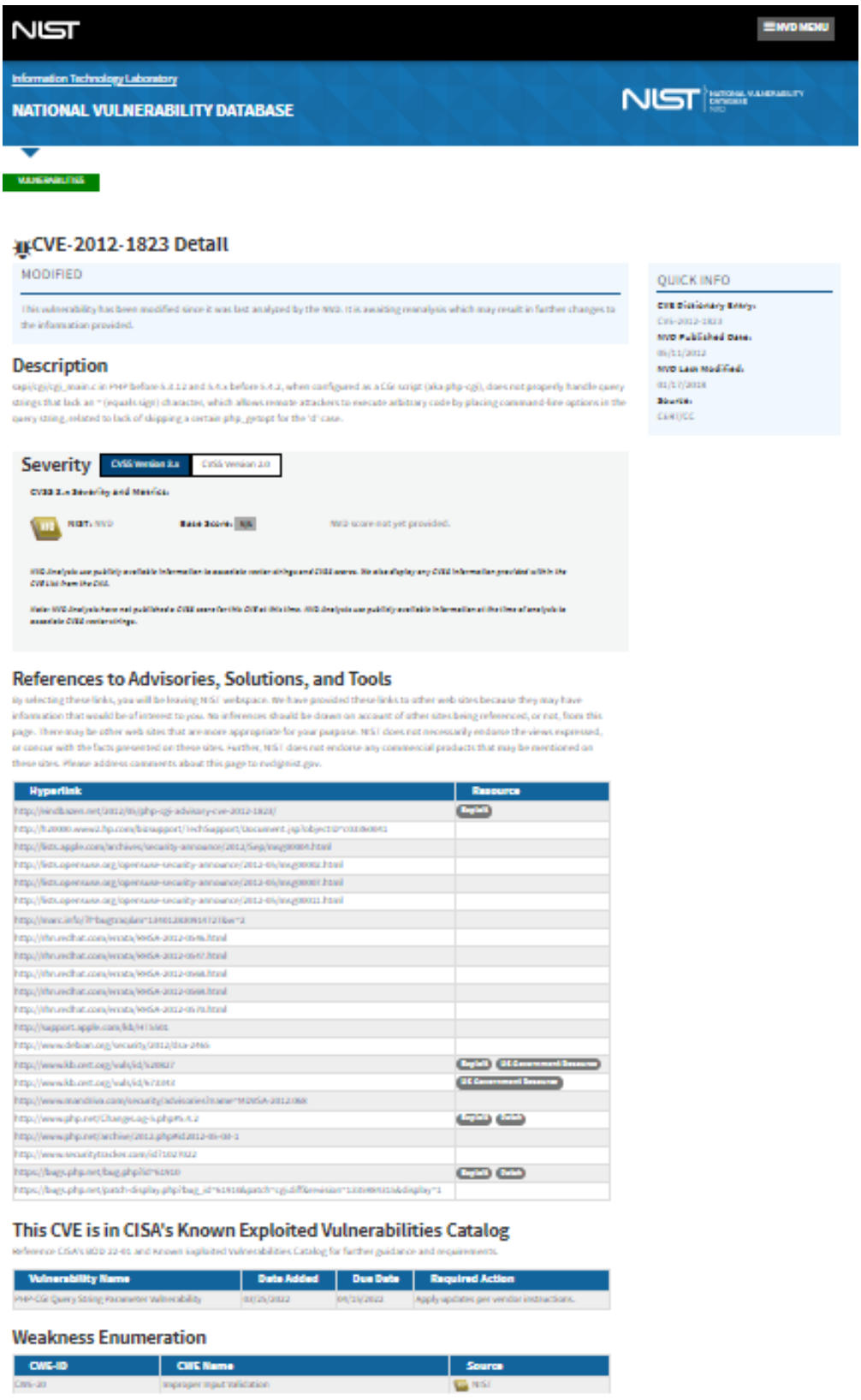
```
GET /401 HTTP/1.1
Host: www.doppler.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-us,en;q=0.5
Cache-Control: no-cache
Cookie: _iub_cs-30208872=%7B%22timestamp%22%3A%222023-05-25T04%3A37%3A55.970Z%22%2C%22version%22%3A%22
1.46.3%22%2C%22purposes%22%3A%7B%221%22%3Atrue%2C%222%22%3Atrue%2C%223%22%3Atrue%2C%224%22%3Atrue%2C%22
5%22%3Atrue%7D%2C%22id%22%3A30208872%2C%22cons%22%3A%7B%22rand%22%3A%22b326dd%22%7D%7D
Referer: https://www.doppler.com/legacy/demo-calendly
User-Agent: Mozilla/5.0 (Windows NT 10.0; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.
77 Safari/537.36
X-Scanner: Netsparker
```

Response

Response Time (ms) : 181.12 Total Bytes Received : 13611 Body Length : 13126 Is Compressed : No

```
HTTP/1.1 200 OK
X-Timer: S1684989806.723123,VS0,VE1
X-Served-By: cache-iad-kiad7000071-IAD, cache-hnd18748-HND
Connection: keep-alive
Content-Security-Policy: frame-ancestors 'self'
Content-Length: 4763
X-Frame-Options: SAMEORIGIN
Accept-Ranges: bytes
X-Cache-Hits: 0, 1
Vary: Accept-Encoding,x-wf-forwarded-proto
Content-Type: text/html
Content-Encoding:
Age: 169
Date: Thu, 25 May 2023 04:43:25 GMT
X-Cache: MISS, HIT
X-Cluster-Name: ap-northeast-1
…
iv class="w-password-page w-form-fail"><div>Incorrect password. Please try again.</div></div><div style
="display:none" class="w-password-page w-embed w-script"><input type="hidden" name="path" value="<%WF_F
ORM_VALUE_PATH%>" /><input type="hidden" name="page" value="<%WF_FORM_VALUE_PAGE%>" /></div><div style
="display:none" class="w-password-page w-embed w-script"><script type="application/javascript">(functio
n _handlePasswordPageOnload() {
if (/[?&]e=1(&|$)/.test(document.location.
…
```

External Reference



Some of the other vulnerabilities identified in the domain.