HALBORN

Prime Trader - Backend Whitebox

WebApp Pentest

Prepared by: **Halborn**

Date of Engagement: **January 4th, 2024 - January 23rd, 2024**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE
0.1	Document Creation	01/15/2024
0.2	Document Edits	01/23/2024
0.3	Draft Review	01/24/2024
1.0	Remediation Plan	02/14/2024
1.1	Remediation Plan Review	02/16/2024

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Erlantz Saenz	Halborn	erlantz.saenz@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Prime Trader engaged Halborn to conduct a security assessment on their web application backend, beginning on January 4th, 2024 and ending on January 23rd, 2024. The security assessment was scoped to the codebase provided to the Halborn team.

1.2 ASSESSMENT SUMMARY

The team at Halborn was provided two weeks for the engagement and assigned a full-time security engineer to verify the security of the backend. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this assessment is to:

- Ensure that backend functions operate as intended
- Identify potential security issues within the backend

The web application's backend displayed a commendable level of maturity; nevertheless, several issues demand the team's attention. Notably, a weak JWT token secret was identified in the session generation process, posing a risk of brute force attacks that could result in the acquisition of valid user sessions, potentially enabling token transfers by malicious actors. Furthermore, the transfer functionality exhibited vulnerabilities susceptible to spam attacks against other users, endangering the platform's reputation and raising the possibility of being flagged as malicious by antivirus vendors.
While the codebase contained several unfiltered parameters, no exploitable pathways were discovered during the assessment period. Nonetheless, it is strongly advised to enforce stringent parameter validation procedures before executing any actions to proactively safeguard the application.

In terms of backend deployment, numerous misconfigurations were unearthed, including instances of cleartext communications and insecure credentials. It is imperative to establish robust password policies for all elements within the infrastructure, creating a defense-in-depth strategy, and encrypting all traffic exchanged between these components to enhance overall security.

In summary, Halborn identified some security risks that were successfully addressed by the Prime Trader team.

1.3 SCOPE

https://github.com/prime-trader/prime

- Commit ID: 55552bce5036c83a6e271a6de8197af8f394b6bb

Deployed sandbox environment:

- https://sandbox.primetrader.com/

Remediation commits:
- Commit ID: 33add499f2e2d7783679e23b507941111a3fdd20
- Commit ID: 0d5e208274c52c7fd64e6315ed0c2e2f02edc1a6

1.4 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this assessment. While manual testing is recommended to uncover flaws in logic, process and implementation; automated testing techniques help enhance coverage of the code and can quickly identify items that do not follow security best practices.

The following phases and associated tools were used throughout the term of the assessment:

- Mapping Application Content and Functionality
- Technology stack-specific vulnerabilities and Code Assessment
- Known vulnerabilities in 3rd party / OSS dependencies
- Application Logic Flaws
- Authentication / Authorization flaws
- Input Handling
- Fuzzing of all input parameters
- Testing for different types of sensitive information leakages: memory, clipboard, etc.
- Test for Injection (SQL/JSON/HTML/JS/Command/Directories. . .)
- Brute Force Attempts
- Perform static analysis on code
- Ensure that coding best practices are being followed by Prime Trader team
- Technology stack-specific vulnerabilities and Code Assessment
- Known vulnerabilities in 3rd party / OSS dependencies.
- Identify potential vulnerabilities that may pose a risk to Prime Trader

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.

4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

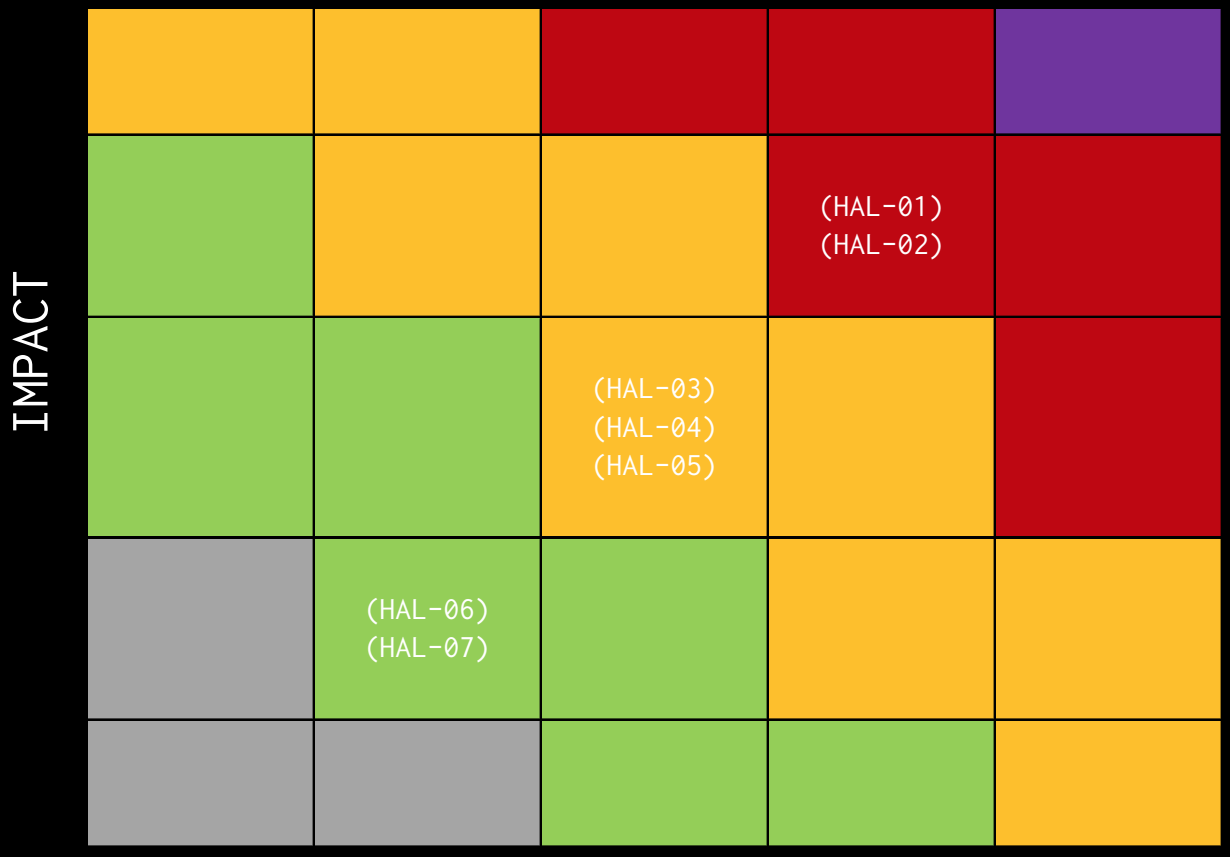
5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	2	3	2	0



EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) WEAK JWT TOKEN SECRET	High	SOLVED - 02/14/2024
(HAL-02) UNAUTHORIZED SPAM VIA TOKENS TRANSFER FUNCTIONALITY	High	SOLVED - 02/14/2024
(HAL-03) UNFILTERED PARAMETERS	Medium	SOLVED - 02/14/2024
(HAL-04) CLEARTEXT COMMUNICATION ALLOWED	Medium	SOLVED - 02/14/2024
(HAL-05) INSECURE CREDENTIALS DURING DEPLOYMENT	Medium	SOLVED - 02/14/2024
(HAL-06) GET METHODS ACCEPTING POST PARAMETERS	Low	SOLVED - 02/14/2024
(HAL-07) USE OF DEPRECATED FUNCTIONS	Low	SOLVED - 02/14/2024

FINDINGS & TECH DETAILS

3.1 (HAL-01) WEAK JWT TOKEN SECRET - HIGH

Description:

The security JWT token secret was made with insufficient complexity or strength, making it vulnerable to bruteforce attacks.
A weak token secret could potentially lead to token forging, impersonation, and unauthorized access to sensitive resources within the application, leading to token transfers

Evidence:

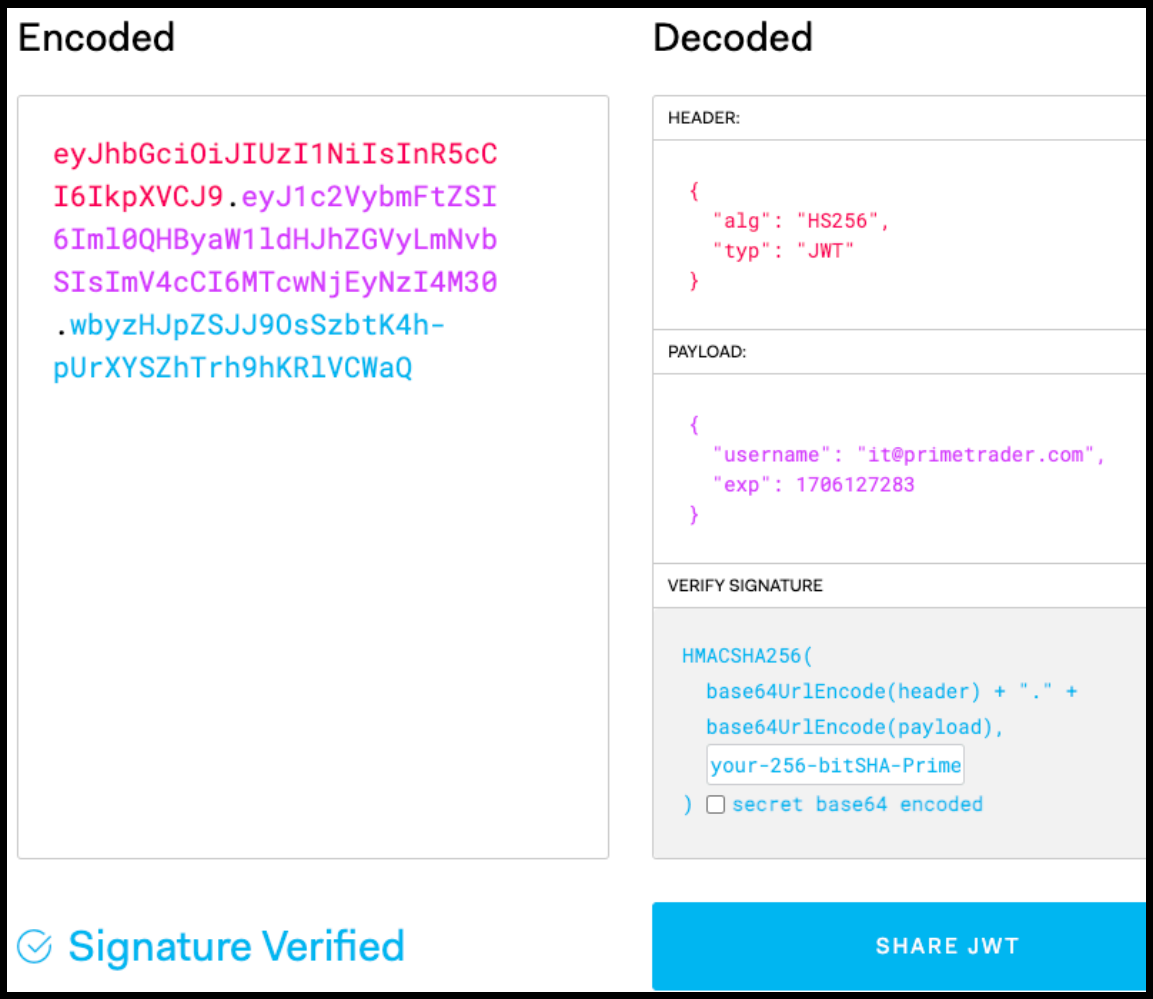


Figure 1: JWT Secret allowed to be bruteforced

Risk Level:

Likelihood - 4
Impact - 4

Recommendation:

- Token Secret Strength: Generate strong and cryptographically secure JWT token secrets with sufficient entropy and complexity.
- Secret Management: Implement robust secret management practices, including secure storage and rotation of token secrets.

Remediation Plan:

SOLVED: The Prime Trader team solved the issue by modifying the JWT Token Secret with a non-guessable string, increasing the complexity of an attack.

3.2 (HAL-02) UNAUTHORIZED SPAM VIA TOKENS TRANSFER FUNCTIONALITY - HIGH

Description:

This security issue enabled a malicious user to abuse the functionality by initiating transfers of 0 tokens, effectively sending an unlimited number of emails to other users without depleting their personal token balance.

Code Location:

Listing 1

```
POST /api/v1/create_transaction HTTP/2
Host: devapi.primetrader.com
Cookie: refresh_token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
↳ eyJ1c2VybmFtZSI6Iml0QHByaW1ldHJhZGVyLmNvbSIsImV4cCI6MTcwNjEyNzI4M30
↳ .PK-Fm0UKWHXdP7lxlvBkLLnY1Oqa-eXKgPMob9CyWsY; access_token=
↳ eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
↳ eyJ1c2VybmFtZSI6Iml0QHByaW1ldHJhZGVyLmNvbSIsImV4cCI6MTcwNTUyNjA4M30
↳ .Lpz_ez_fBpsLWZycjivmxkCX8_Mp6RW0By6HOyYpZpk
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv
↳ :109.0) Gecko/20100101 Firefox/116.0
Accept: */*
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Referer: https://sandbox.primetrader.com/
Authorization: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
↳ eyJ1c2VybmFtZSI6Iml0QHByaW1ldHJhZGVyLmNvbSIsImV4cCI6MTcwNTUyNjA4M30
↳ .Lpz_ez_fBpsLWZycjivmxkCX8_Mp6RW0By6HOyYpZpk
Content-Type: application/json
Origin: https://sandbox.primetrader.com
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
Cache-Control: max-age=0
Te: trailers
```

```
Content-Length: 74

{
  "pt_tokens": 0,
  "receiver_email": "erlantz.saenz+1@halborn.com"
}
```

Listing 2

```
HTTP/2 200 OK
Date: Wed, 17 Jan 2024 20:24:57 GMT
Content-Type: text/plain; charset=utf-8
Content-Length: 75
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: https://sandbox.primetrader.com
Vary: Origin

{
  "code": 200,
  "data": {
    "message": "transaction successfull"
  }
}
```

Risk Level:

Likelihood - 4
Impact - 4

Recommendation:

- Tokens Transfer Validation: Implement rigorous validation checks within the tokens transfer functionality to prevent unauthorized or malicious transfers of 0 tokens.
- Rate Limiting: Introduce rate limiting or anti-spam measures to restrict the frequency and volume of token transfers initiated by a single user.
- Monitoring and Logging: Set up monitoring and logging mechanisms to detect and record suspicious or excessive token transfer activities.

- User Notification: Notify users of potential spam attempts and provide reporting mechanisms to report abusive behavior.

Remediation Plan:

SOLVED: The Prime Trader teamincluded enough validations on the structs' definition, not allowing an attacker to abuse the transfer functionality.

3.3 (HAL-03) UNFILTERED PARAMETERS - MEDIUM

Description:

Numerous unsanitized parameters were detected within the codebase. Such a scenario could potentially enable malicious actors to inject harmful data into the backend, subsequently executing unintended operations on both the application and the underlying infrastructure.
Numerous parameters were injected without undergoing sanitization into the parameterized SQL queries, thereby placing the entirety of security reliance on the effectiveness of the parameterized queries.

Code Location:

Listing 3: server/api/ep_signup_verify.go (Line 42)

```
41     // Parse out referral token variable.
42     ReferralCode := qVals.Get("rc")
```

Listing 4: server/api/ep_signup_verify.go (Line 154)

```
152 if ReferralCode != "" {
153         // Get referrer info.
154         referrerUsertup, err := c.UserDb.GetUserByReferralCode(
 ↳ ReferralCode)
155         if err != nil {
156             fmt.Printf("failed to get referred user info: %v\n",
 ↳ err)
157             resp.SendData(http.StatusBadRequest, "failed to get
 ↳ referred user info")
158             return
159         }
```

Listing 5: db/lib/user/auth.go (Lines 297,295)

```
295 func (c *UserDbC) GetUserByReferralCode(code string) (*UserTup,
 ↳ error) {
```

```
296     tup := &UserTup{}
297     err := c.Pg.QueryRow("SELECT * FROM users WHERE referral_code=
 ↳ $1", code).Scan(tup.allFields()...)
298     if err != nil {
299         return nil, err
300     }
301 
302     return tup, nil
303 }
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

- Input Validation: Robust input validation and filtering mechanisms should be implemented to ensure that user-supplied data is safe and conforms to expected data types and formats.
- Parameterized Queries: The use of parameterized queries or prepared statements should be employed to interact with databases, preventing SQL injection attacks.

Remediation Plan:

SOLVED: The Prime Trader team included validations on the code that verifies the type of the data before performing any operation.

3.4 (HAL-04) CLEARTEXT COMMUNICATION ALLOWED - MEDIUM

Description:

The postgres communication was enforced in sslmode=disable. This configuration enforces the data transmission without encryption. The absence of encryption exposes transmitted data to potential interception and unauthorized access.

Code Location:

Listing 6

```
1 docker run --network host migrator -path=/migrations/ -database "
↳ postgres://primetrader:primetrader@localhost:5433/primetrader?
↳ sslmode=disable" up
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

Enable SSL/TLS: Modify the PostgreSQL SSLMode configuration to “require” or “prefer” to ensure the enforcement of encrypted communication between client and server. This guarantees the confidentiality of data exchanged during communication.

References:

Postgresql documentation

Remediation Plan:

SOLVED: The Prime Trader team solved the issue by enforcing the parameter `sslmode=required`that it would enforce a secure communication.

3.5 (HAL-05) INSECURE CREDENTIALS DURING DEPLOYMENT - MEDIUM

Description:

It was found that during the deployment instructions weak credentials were used. When such insecure credentials are utilized during deployment, it exposes the system or application to a high-risk scenario, potentially allowing unauthorized access, data breaches, and security compromises.

Code Location:

Listing 7

```
1 psql -U postgres -c "CREATE USER primetrader PASSWORD 'primetrader
↳ ' SUPERUSER"
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

Credential Management: Implement robust credential management practices. Utilize strong, complex passwords, keys, or tokens for authentication, ensuring they are not easily guessable.
Credential Rotation: Regularly update and rotate credentials to prevent long-term exposure. Invalidate old or compromised credentials promptly.
Least Privilege Principle: Adhere to the principle of least privilege when assigning permissions to credentials during deployment. Grant only the necessary privileges to reduce potential damage in case of compromise.

Remediation Plan:

SOLVED: The Prime Trader team solved the issue, modifying the password of the DB_URL environmental variable with a non-guessable password. However, it should be highlighted that the TESTDB_URL still contained a guessable password.

3.6 (HAL-06) GET METHODS ACCEPTING POST PARAMETERS - LOW

Description:

The HTTP GET method, intended for retrieving data and not designed to accept parameters in the request body, was found to be processing POST parameters.
This could lead to unexpected application behavior, data exposure, or security vulnerabilities, depending on how POST parameters were processed within the GET request.

Code Location:

Listing 8

```
GET /api/v1/username_exist HTTP/2
Host: devapi.primetrader.com
Cookie: refresh_token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
↳ eyJ1c2VybmFtZSI6Iml0QHByaW1ldHJhZGVyLmNvbSIsImV4cCI6MTcwNjEyNzI4M30
↳ .PK-Fm0UKWHXdP7lxlvBkLLnY1Oqa-eXKgPMob9CyWsY; access_token=
↳ eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
↳ eyJ1c2VybmFtZSI6Iml0QHByaW1ldHJhZGVyLmNvbSIsImV4cCI6MTcwNTUyNjA4M30
↳ .Lpz_ez_fBpsLWZycjivmxkCX8_Mp6RW0By6HOyYpZpk
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv
↳ :109.0) Gecko/20100101 Firefox/116.0
Accept: */*
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Referer: https://sandbox.primetrader.com/
Authorization: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
↳ eyJ1c2VybmFtZSI6Iml0QHByaW1ldHJhZGVyLmNvbSIsImV4cCI6MTcwNTUyNjA4M30
↳ .Lpz_ez_fBpsLWZycjivmxkCX8_Mp6RW0By6HOyYpZpk
Content-Type: application/json
Origin: https://sandbox.primetrader.com
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
Cache-Control: max-age=0
Te: trailers
```

```
Content-Length: 23

{
"username":"test"
}
```

Listing 9

```
HTTP/2 200 OK
Date: Wed, 17 Jan 2024 20:41:38 GMT
Content-Type: text/plain; charset=utf-8
Content-Length: 53
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: https://sandbox.primetrader.com
Vary: Origin

{
  "code": 200,
  "data": {
    "exist": false
  }
}
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

- HTTP Method Compliance: Ensure that HTTP methods are used in accordance with their intended purpose. The GET method should be reserved for data retrieval, while POST should be used for data submission.
- Request Validation: Implement rigorous request validation mechanisms to ensure that the correct HTTP methods are used for their intended purposes.

Remediation Plan:

SOLVED: The Prime Trader team solved the issue, adding the appropriate checks to handle the variables correctly.

3.7 (HAL-07) USE OF DEPRECATED FUNCTIONS - LOW

Description:

strings.Title is deprecated: The rule Title uses for word boundaries does not handle Unicode punctuation properly.

Code Location:

Listing 10: server/api/ep_signup_verify.go (Line 190)

```
190 username := strings.Title(tup.Username)
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

Use golang.org/x/text/cases instead.

Remediation Plan:

SOLVED: The Prime Trader team solved the issue, updating the dependencies and deprecated functions.

THANK YOU FOR CHOOSING

// HALBORN