WEB APPLICATION PENETRATION TEST

Report for:	
Date:	

This document contains confidential information about IT systems and network infrastructure of the customer, as well as information about potential vulnerabilities and methods of their exploitation. This confidential information is for internal use by the customer only and shall not be disclosed to third parties.

HackControl
info@hackcontrol.org

Table of Contents

HackControl
info@hackcontrol.org

Introduction

We thank Client for the opportunity to carry out a security assessment of the web application. This document describes a methodology, limitations and results of the assessment.

HackControl
info@hackcontrol.org

Executive Summary

Hackcontrol (Provider) was contracted by CLIENT (Customer) to carry out a penetration test of the Client's web application.

This report presents findings of the penetration test conducted between DD/MM/YYYY - DD'/MM'/YYYY.

The main subject of testing is CLIENT`s exchange web system.

Penetration test has the following objectives:

- identify technical and functional vulnerabilities
- evaluate a severity level (ease of use, impact on information systems, etc);
- make a prioritized list of recommendations to address identified weaknesses

According to our research after performing the penetration testing, security rating of CLIENT`s infrastructure was identified as **Low**.

Team

Role	Name	EMAIL
Project Manager	John Doe (CEH, ISO27001 LA)	info@hackcontrol.org
Penetration Testing Engineer	John Doe (OSCP, eWPT, eCPPT)	engineer@hackcontrol.org

Scope of Security Assessment

The following list of the information systems was the scope of the Security Assessment.

#	Name	Description
1.	client.com www.client.com h5.client.com openws.client.com ws-manager.client.com ws.client.com gitlab.infra.client.com registry.infra.client.com nexus.infra.client.com wiki.infra.client.com	Web
2.	35.220.000.000 35.240.00.000 35.190.00.000 35.240.00.000 35.220.000.000 130.210.00.00	IP
3.	api.Client.com openapi.Client.com (https://github.com/Client/Client-official-api-docs)	API

HackControl
info@hackcontrol.org

Methodology

The testing methodology is based on generally accepted industry-wide approaches to perform penetration testing for web applications (OWASP Testing Guide);

Application-level penetration tests include, at a minimum, checking for the following types of vulnerabilities:

- injections, in particular, SQL injections, noSQL, XPath, etc.;
- Local File Inclusion (LFI), Remote File Inclusion (RFI);
- Cros-Site Scripting (XSS);
- errors in access control mechanisms (for example, unsafe direct links to objects, lack of restriction of access by URL, directory traversal and lack of restriction of user access rights to functions);
- Cross-Site Request Forgery (CSRF);
- web server configuration errors;
- incorrect error handling;
- Counteracting the compromise of authentication mechanisms and session management (Session Management Testing);

HackControl
info@hackcontrol.org

Severity Definition

The level of criticality of each risk is determined based on the potential impact of loss from successful exploitation as well as ease of exploitation, existence of exploits in public access and other factors.

Severity	Description
High 🟥	High-level vulnerabilities are easy in exploitation and may provide an attacker with full control of the affected systems, also may lead to significant data loss or downtime. There are exploits or PoC available in public access.
Medium 🟨	Medium-level vulnerabilities are much harder to exploit and may not provide the same access to affected systems. No exploits or PoCs available in public access. Exploitation provides only very limited access.
Low 🟩	Low-level vulnerabilities provide an attacker with information that may assist them in conducting subsequent attacks against target information systems or against other information systems, which belong to an organization. Exploitation is extremely difficult, or impact is minimal.
Info 🟦	These vulnerabilities are informational and can be ignored.

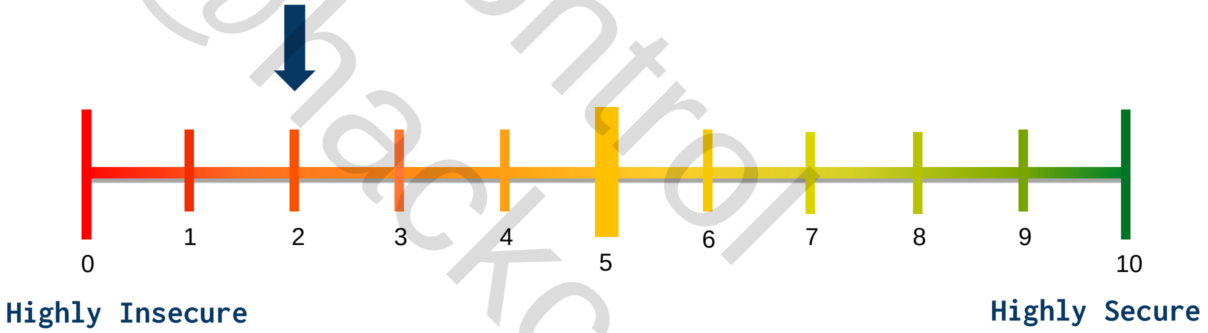
Hackcontrol
info@hackcontrol.org

Summary of Findings

According to the following in-depth testing of the environment, CLIENT’s web application require some improvements.

Value	Number of risks
High	5
Medium	2
Low	1
Info	1

Based on our understanding of the IT Infrastructure, as well as the nature of the vulnerabilities discovered, their exploitability, and the potential impact we have assessed the level of risk for your organization to be High.



HackControl
info@hackcontrol.org

Key Findings

Rate limit bypass via X-Forwarded-For

#1	Description	Type: Real

X-Forwarded-For is a well-established HTTP header used by proxies, to pass along other IP addresses in the request. This is often the same as CF-Connecting-IP, but there may be multiple layers of proxies in a request path.

There is dynamically changing value can attackers do brute force 6-digits approve code and other attacks witch based on brute force method.

Evidences

Steps to reproduce:

1. Get request for restore password
2. Input some code
3. Intercept request and set header X-Forwarded-For with something value
4. The count of the number of attempts will be restored to the initial value

Request:

```
POST [redacted].com/api/user_findPwd HTTP/1.1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: application/json, text/plain, */*
Accept-Language: uk-UA,uk;q=0.8,en-US;q=0.5,en;q=0.3
Referer: [redacted]forget/en
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
X-Forwarded-For: TEST12312X
Content-Length: 150
Connection: keep-alive
[redacted]
Host: [redacted]
```

```
loginName=bus[redacted]-mail.net&loginType=1&pwdType=0&emailCode=718315&newPwd=
146d2f289749a5c70b1dbe65ef6[redacted]&reNewPwd=146d2f289749a5c70b1dbe65ef6[redacted]
```

Recommendations

- Check value of headers
- Add a “one-time token”

HackControl
info@hackcontrol.org

Broken Authentication and Session Management

<table>
<tr><th>#2</th><th>Description</th><th>Type: Real</th></tr>
<tr><td colspan="3">Incorrect logic in the transfer of the session between domains allows to intercept user session.
The WebSocket application at client.com is responsible for mediating the session for the main casino application, which can be located on one of the mirrors, for example at client.com and client.com.
This functionality is used to dynamically transfer the session to different mirrors, which allows the user not to log into the system every time when changing such a mirror. Also, the websocket of the application on client.com does not have a built-in validation of the domain from which the session request comes, which allows to get a user session for any domain.
An example of such session interception is located at https://ps29.net/client-dwju3726ks/. This page contains the authorization.js (https://www.client.com/files/js/authorization.js) code that pinup uses for authorization.</td></tr>
<tr><td colspan="2">Evidences</td><td><b>Steps to reproduce:</b>
1. Login to any account on client domain
2. Go to https://ps29.net/client-dwju3726ks/</td></tr>
<tr><td colspan="2">Recommendations</td><td>● Add domain validation</td></tr>
</table>

HackControl
info@hackcontrol.org

Open redirect

#3	Description	Type: Real

Open redirection in the inter-domain session transfer functionality that allows to issue a session for a malicious domain. The application /v2/verify/ is responsible for issuing a session for the main casino application, which can be located at one of the mirrors, for example, client.com and client.com. This functionality is used to dynamically transfer the session to different mirrors, which allows the user not to log into the system every time when changing such a mirror.

Evidences

Steps to reproduce:
https://client.com/v2/verify/<login>/<hash>?url=<currentUrl>&domain=<origin>

Open redirection in the domain parameter allows to get a user session for any domain. The following link was used to illustrate this vulnerability.
https://client.com/v2/verify/x/x?url=x&domain=../../../%5Cexample.com/

Response:

```
HTTP/1.1 302 Found
Server: nginx
Date: Mon, 17 Dec 2018 13:56:50 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 195
Connection: keep-alive
Location:
/\example.com/crossdomain/set/1599129/43875a650f865a828e14e133bba1a
0987145adba0b72361dcb919618a1c0d51a0cf2369f51e13c5487b3b1069d8d1948
34c49cf517a46b2cb3250e1f9e8a76a0?url=x
```

Recommendations	• Add a “one-time token” or set up rate limits for this request

HackControl
info@hackcontrol.org

IDOR for change or remove API-keys

#4	Description	Type: Real

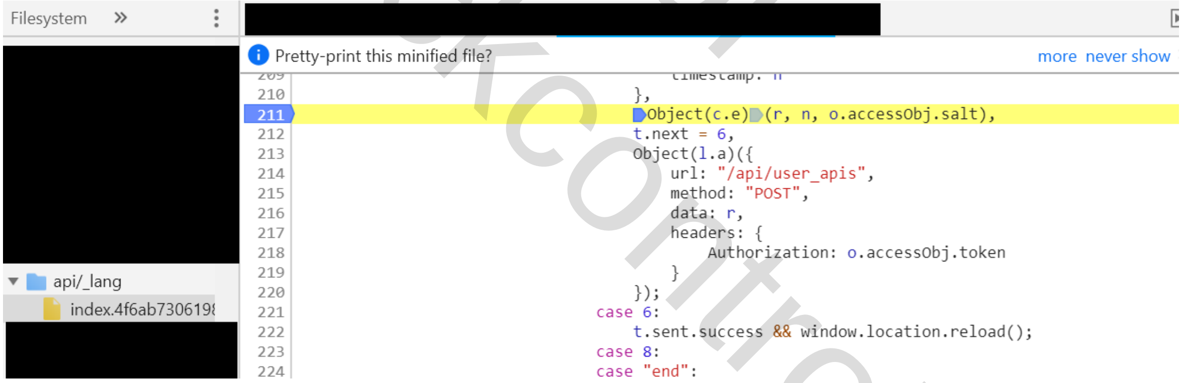
Insecure Direct Object References occur when an application provides direct access to objects based on user-supplied input. As a result of this vulnerability attackers can bypass authorization and access resources in the system directly, for example database records or files. Insecure Direct Object References allow attackers to bypass authorization and access resources directly by modifying the value of a parameter used to directly point to an object. Such resources can be database entries belonging to other users, files in the system, and more. This is caused by the fact that the application takes user supplied input and uses it to retrieve an object without performing sufficient authorization checks.

There is possibility to change another API-keys by just change id value. There is no session or access checking for this operation. No current The attacker can access, edit or delete any of other user`s API-keys by changing the values.

Evidences

Steps to reproduce:

1. Go to https://www.Client.com/api/en in Chrome and open dev tools.
2. In Sources open https://www.client.com/_nuxt/pages/api/_lang/index.4f6ab73061981ec9a06e.js and choose pretty-print.
3. Set breakpoint in line 2



[redacted]

[redacted]

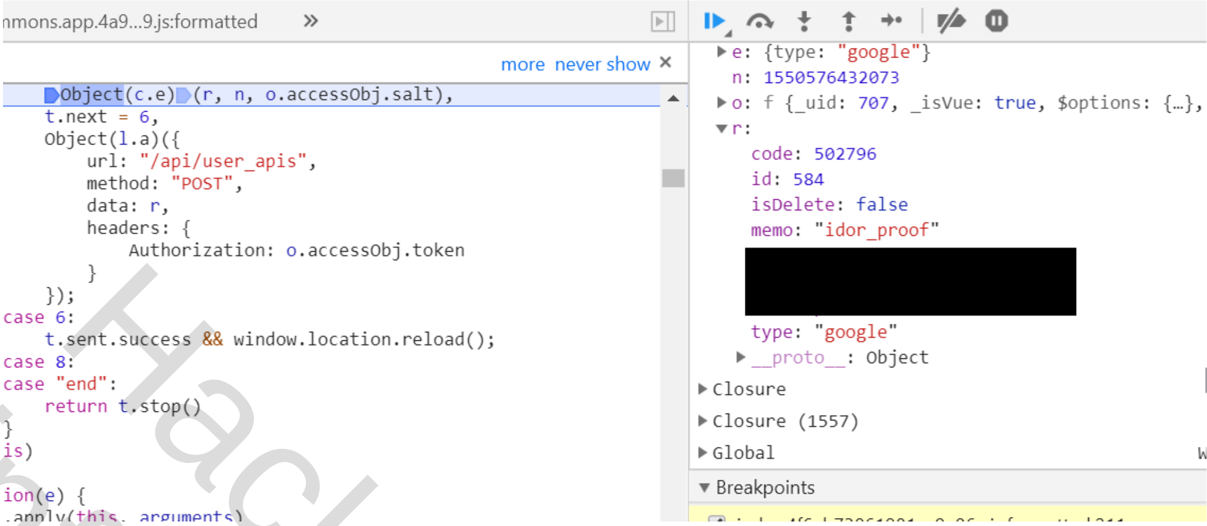
[redacted]

```
209                     timestamp: n
210                 },
211                 Object(c.e)(r, n, o.accessObj.salt),
212                 t.next = 6,
213                 Object(l.a)({
214                     url: "/api/user_apis",
215                     method: "POST",
216                     data: r,
217                     headers: {
218                         Authorization: o.accessObj.token
219                     }
220                 });
221             case 6:
222                 t.sent.success && window.location.reload();
223             case 8:
224             case "end":
```

4. Press Edit across one of your keys, input new data, 2FA-code and send requests
5. In the same time breakpoint trigger is work. You can change in id-field and resume script work

HackControl
info@hackcontrol.org

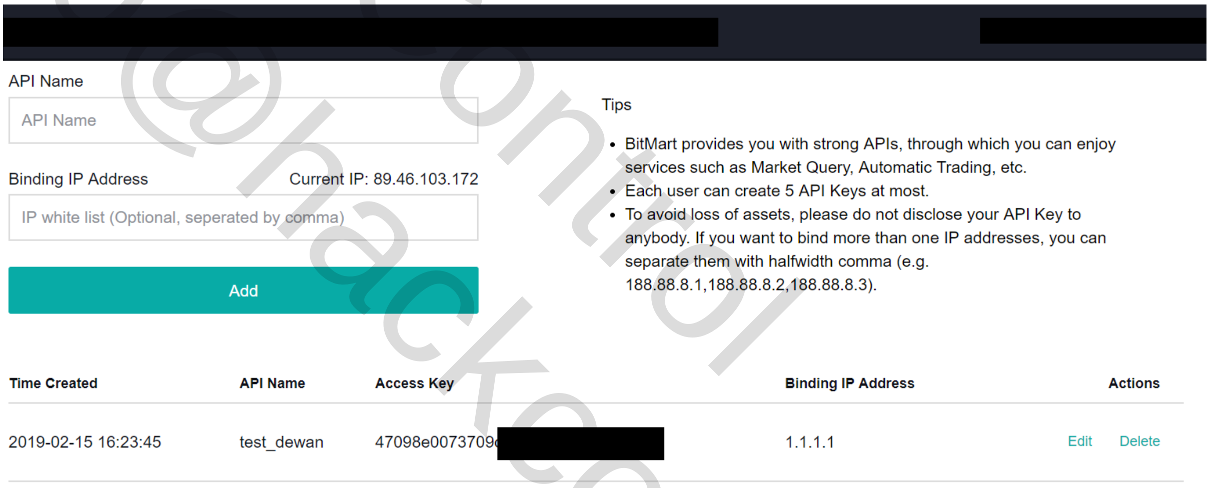
HackControl
info@hackcontrol.org



```
    Object(c.e)(r, n, o.accessObj.salt),
    t.next = 6,
    Object(l.a)({
        url: "/api/user_apis",
        method: "POST",
        data: r,
        headers: {
            Authorization: o.accessObj.token
        }
    });
case 6:
    t.sent.success && window.location.reload();
case 8:
case "end":
    return t.stop()
}
is)

ion(e) {
```

```
▶e: {type: "google"}
 n: 1550576432073
▶o: f {_uid: 707, _isVue: true, $options: {…},
▼r:
   code: 502796
   id: 584
   isDelete: false
   memo: "idor_proof"
   [redacted]
   type: "google"
  ▶__proto__: Object
▶Closure
▶Closure (1557)
▶Global
```



[redacted] [redacted]

API Name

API Name

Binding IP Address Current IP: 89.46.103.172

IP white list (Optional, seperated by comma)

Add

Tips

- BitMart provides you with strong APIs, through which you can enjoy services such as Market Query, Automatic Trading, etc.
- Each user can create 5 API Keys at most.
- To avoid loss of assets, please do not disclose your API Key to anybody. If you want to bind more than one IP addresses, you can separate them with halfwidth comma (e.g. 188.88.8.1,188.88.8.2,188.88.8.3).

Time Created	API Name	Access Key	Binding IP Address	Actions
2019-02-15 16:23:45	test_dewan	47098e0073709 [redacted]	1.1.1.1	Edit Delete

Recommendations	• It is not recommended to use any id for request, especially like user id, it is better to use session management keys (cookies for example) and identify user by session keys. Also every operation has to be checked for permission access for current user and his permissions. For more details please visit: https://www.owasp.org/index.php/Top_10_2013-A4-Insecure_Direct_Object_References

Reflected Cross-Site Scripting

#5	Description	Type: Real

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted web sites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

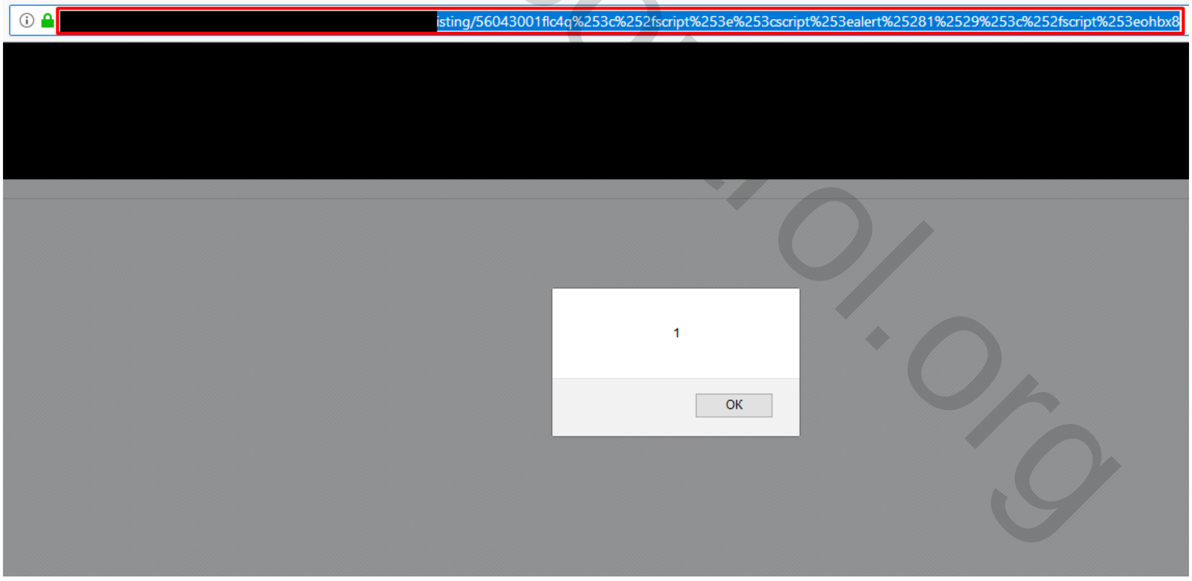
An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page.

There were found 2 Real (Validated) XSS.

Evidences

Steps to reproduce:

1. Reflected XSS in url https://www.client.com/store/ /listing/56043001flc4q%253c%252fscript%253e%253cscript%253ealert%25281%2529%253c%252fscript%253eohbx8
2. Reflected XSS in x-ncpl-csrf anti CSRF token. Change value of x-ncpl-csrf anti CSRF token to x-ncpl-csrf=44cab53c34ff44f6bc1993d42bbe9bfbkz4tu%22%3e%3cscript%3ealert(1)%3c%2fscript%3ef7ncy



[redacted]isting/56043001flc4q%253c%252fscript%253e%253cscript%253ealert%25281%2529%253c%252fscript%253eohbx8

[redacted]

Recommendations

- It is not recommended to use any id for request, especially like user id, it is better to use

HackControl info@hackcontrol.org

	session management keys (cookies for example) and identify user by session keys. Also every operation has to be checked for permission access for current user and his permissions. For more details please visit: https://www.owasp.org/index.php/Top_10_2013-A4-Insecure_Direct_Object_References

HackControl
info@hackcontrol.org

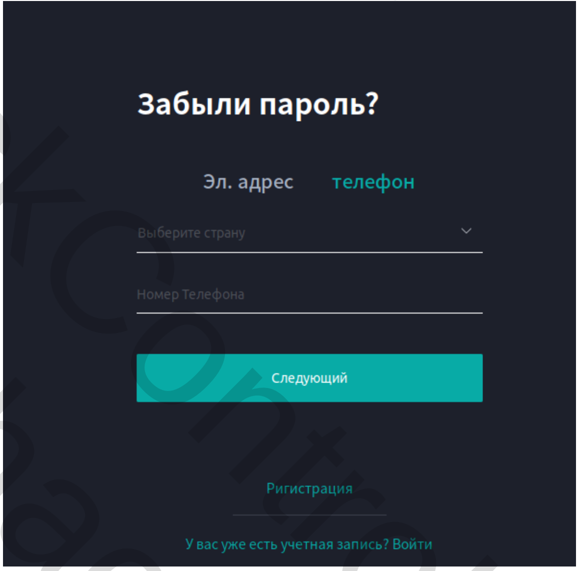
Email disclosure via Forgot password

#6	Description	Type: Real
It is possible to get information about registered e-mail.		

Evidences

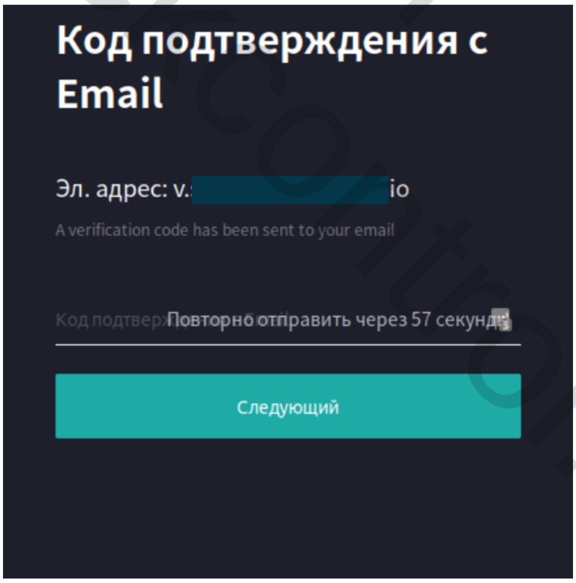
Steps to reproduce:

1. Go to page https://www.Client.com/forget/ru



Забыли пароль?

Response:



Код подтверждения с Email

Эл. адрес: v.[redacted]io

A verification code has been sent to your email

Recommendations	
	● Shouldn`t show the email address when restore a password via the phone.

HackControl
info@hackcontrol.org

HackControl
info@hackcontrol.org

User enumeration

#7	Description	Type: Real

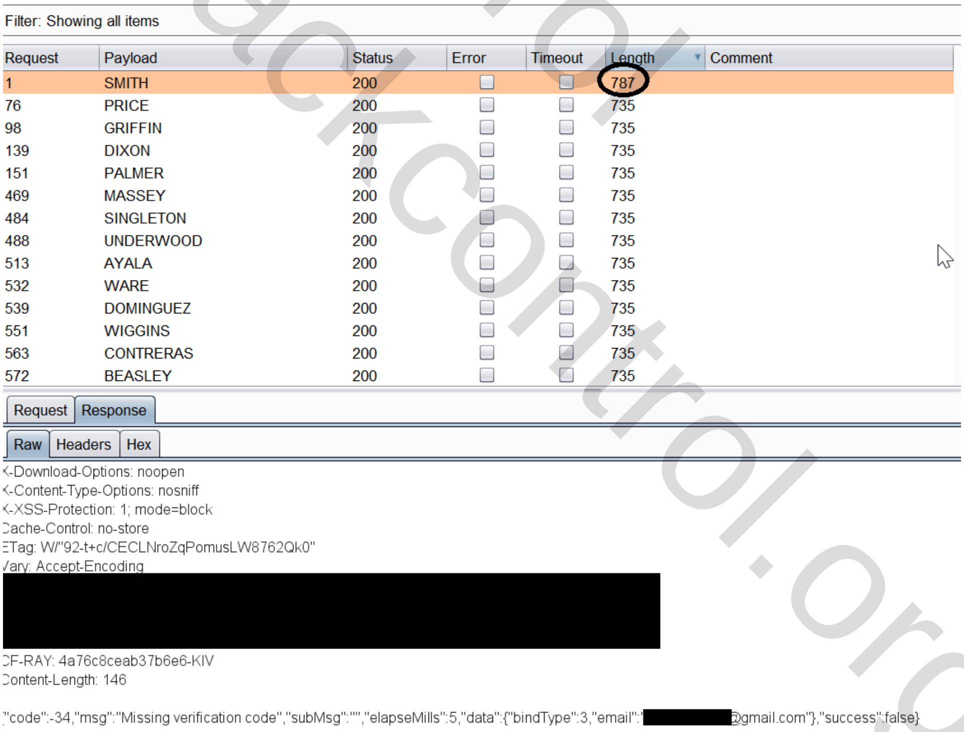
The scope of this test is to verify whether it's possible to collect a set of valid usernames by interacting with the authentication mechanism of the application. This test will be useful for a brute force testing, in which we verify if, given a valid username, it's possible to find a corresponding password. Often, web applications reveal when a username exists in a system, either as a consequence of a misconfiguration or as a design decision.

For example, sometimes, when we submit wrong credentials, we receive a message stating that either the username is present in the system or the provided password is wrong. The information obtained can be used by an attacker to gain a list of users in the system. This information can be used to attack the web application, for example, through a brute force or default username/password attack.

Evidences

Steps to reproduce:

1. Intercept request POST /api/user_findPwd
2. Send request to Intruder
3. Set payload to loginName=<email>&loginType=1&pwdType=0
4. Run attack



Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
1	SMITH	200	□	□	787	
76	PRICE	200	□	□	735	
98	GRIFFIN	200	□	□	735	
139	DIXON	200	□	□	735	
151	PALMER	200	□	□	735	
469	MASSEY	200	□	□	735	
484	SINGLETON	200	□	□	735	
488	UNDERWOOD	200	□	□	735	
513	AYALA	200	□	□	735	
532	WARE	200	□	□	735	
539	DOMINGUEZ	200	□	□	735	
551	WIGGINS	200	□	□	735	
563	CONTRERAS	200	□	□	735	
572	BEASLEY	200	□	□	735	

```
<-Download-Options: noopen
<-Content-Type-Options: nosniff
<-XSS-Protection: 1; mode=block
Cache-Control: no-store
ETag: W/"92-t+c/CECLNroZqPomusLW8762Qk0"
Vary: Accept-Encoding
[redacted]
CF-RAY: 4a76c8ceab37b6e6-KIV
Content-Length: 146

"code":-34,"msg":"Missing verification code","subMsg":"","elapseMills":5,"data":{"bindType":3,"email":"[redacted]@gmail.com"},"success":false}
```

Recommendations	- It’s recommended not to show whether the user is logged in the system or not

HackControl
info@hackcontrol.org

Vulnerability Lucky13 and BREACH

#8	Description	Type: Potential

BREACH

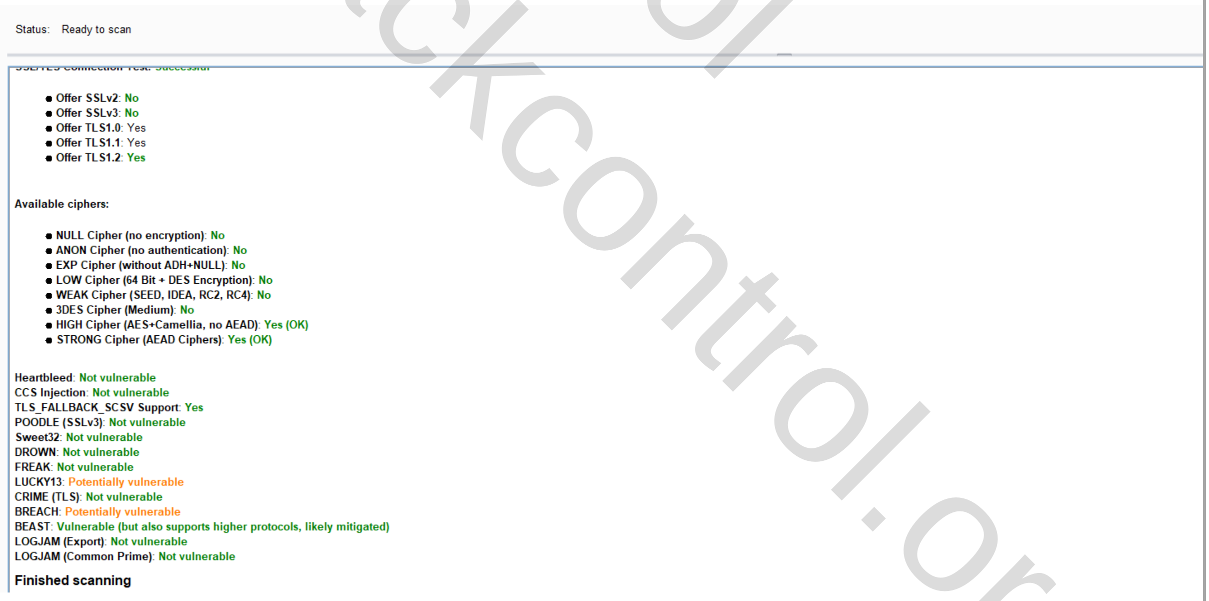
Short for Browser Exploit Against SSL/TLS, BREACH is a browser exploit against SSL/TLS that was revealed in late September 2011. This attack leverages weaknesses in cipher block chaining (CBC) to exploit the Secure Sockets Layer (SSL)/Transport Layer Security (TLS) protocol. The CBC vulnerability can enable man-in-the-middle (MITM) attacks against SSL in order to silently decrypt and obtain authentication tokens, thereby providing hackers access to data passed between a Web server and the Web browser accessing the server.

LUCKY13

The TLS 1.1 and 1.2 protocols and the DTLS 1.0 and 1.2 protocols, as used in OpenSSL, OpenJDK, PolarSSL, and other products, do not properly consider timing side-channel attacks on a MAC check requirement during the processing of malformed CBC padding. This allows remote attackers to conduct distinguishing attacks and plaintext-recovery attacks via statistical analysis of timing data for crafted packets, aka the "Lucky Thirteen" issue.

Evidences

Scanning https://www.client.com vith SSLscan



```
Status:   Ready to scan

SSL/TLS Connection Test: Successful

    ● Offer SSLv2: No
    ● Offer SSLv3: No
    ● Offer TLS1.0: Yes
    ● Offer TLS1.1: Yes
    ● Offer TLS1.2: Yes

Available ciphers:

    ● NULL Cipher (no encryption): No
    ● ANON Cipher (no authentication): No
    ● EXP Cipher (without ADH+NULL): No
    ● LOW Cipher (64 Bit + DES Encryption): No
    ● WEAK Cipher (SEED, IDEA, RC2, RC4): No
    ● 3DES Cipher (Medium): No
    ● HIGH Cipher (AES+Camellia, no AEAD): Yes (OK)
    ● STRONG Cipher (AEAD Ciphers): Yes (OK)

Heartbleed: Not vulnerable
CCS Injection: Not vulnerable
TLS_FALLBACK_SCSV Support: Yes
POODLE (SSLv3): Not vulnerable
Sweet32: Not vulnerable
DROWN: Not vulnerable
FREAK: Not vulnerable
LUCKY13: Potentially vulnerable
CRIME (TLS): Not vulnerable
BREACH: Potentially vulnerable
BEAST: Vulnerable (but also supports higher protocols, likely mitigated)
LOGJAM (Export): Not vulnerable
LOGJAM (Common Prime): Not vulnerable

Finished scanning
```

HackControl
info@hackcontrol.org

Recommendations	● Disable TLS 1.0 and make user connections using TLS 1.1 or TLS 1.2 protocols which are immune to the BEAST attack. TLS 1.0 is now considered insecure. Disabling the TLS 1.0 protocol improves the overall security. ● Avoid using TLS in CBC-mode and switch to AEAD algorithms.

Cacheable HTTPS response

#9	Description	Type: Real
Unless directed otherwise, browsers may store a local cached copy of content received from web servers. Some browsers, including Internet Explorer, cache content accessed via HTTPS. If sensitive information in application responses is stored in the local cache, then this may be retrieved by other users who have access to the same computer at a future time.(Cache-control: no-store, Pragma: no-cache)		
Recommendations	Add the following headers: ● Cache-control: no-store ● Pragma: no-cache	

HackControl
info@hackcontrol.org

Appendix A. OWASP Testing Checklist

Category	Test Name	Result
Information Gathering		
OTG-INFO-001	Conduct Search Engine Discovery and Reconnaissance for Information Leakage	🟩 Tested
OTG-INFO-002	Fingerprint Web Server	🟩 Tested
OTG-INFO-003	Review Webserver Metafiles for Information Leakage	🟩 Tested
OTG-INFO-004	Enumerate Applications on Webserver	🟩 Tested
OTG-INFO-005	Review Webpage Comments and Metadata for Information Leakage	🟩 Tested
OTG-INFO-006	Identify application entry points	🟦 Tested
OTG-INFO-007	Map execution paths through application	🟦 Tested
OTG-INFO-008	Fingerprint Web Application Framework	🟩 Tested
OTG-INFO-009	Fingerprint Web Application	🟩 Tested
OTG-INFO-010	WAF	🟩 Tested
Configuration and Deploy Management Testing		
OTG-CONFIG-001	Test Network/Infrastructure Configuration	🟨 Tested
OTG-CONFIG-002	Test Application Platform Configuration	🟩 Tested
OTG-CONFIG-003	Test File Extensions Handling for Sensitive Information	🟩 Tested
OTG-CONFIG-004	Backup and Unreferenced Files for Sensitive Information	🟩 Tested
OTG-CONFIG-005	Enumerate Infrastructure and Application Admin Interfaces	🟩 Tested
OTG-CONFIG-006	Test HTTP Methods	🟩 Tested
OTG-CONFIG-007	Test HTTP Strict Transport Security	🟩 Tested
OTG-CONFIG-008	Test RIA cross domain policy	🟨 Tested
Identity Management Testing		
OTG-IDENT-001	Test Role Definitions	🟦 N/A
OTG-IDENT-002	Test User Registration Process	🟩 Tested
OTG-IDENT-003	Test Account Provisioning Process	🟦 N/A
OTG-IDENT-004	Testing for Account Enumeration and Guessable User Account	🟥 Tested
OTG-IDENT-005	Testing for Weak or unenforced username policy	🟩 Tested
OTG-IDENT-006	Test Permissions of Guest/Training Accounts	🟦 N/A
OTG-IDENT-007	Test Account Suspension/Resumption Process	🟩 Tested
Authentication Testing		
OTG-AUTHN-001	Testing for Credentials Transported over an Encrypted Channel	🟨 Tested
OTG-AUTHN-002	Testing for default credentials	🟦 N/A
OTG-AUTHN-003	Testing for Weak lock out mechanism	🟩 Tested
OTG-AUTHN-004	Testing for bypassing authentication schema	🟩 Tested
OTG-AUTHN-005	Test remember password functionality	🟩 Tested
OTG-AUTHN-006	Testing for Browser cache weakness	🟨 Tested
OTG-AUTHN-007	Testing for Weak password policy	🟨 Tested
OTG-AUTHN-008	Testing for Weak security question/answer	🟩 Tested
OTG-AUTHN-009	Testing for weak password change or reset functionalities	🟥 Tested

HackControl
info@hackcontrol.org

OTG-AUTHN-010	Testing for Weaker authentication in alternative channel	Tested
Authorization Testing		
OTG-AUTHZ-001	Testing Directory traversal/file include	Tested
OTG-AUTHZ-002	Testing for bypassing authorization schema	Tested
OTG-AUTHZ-003	Testing for Privilege Escalation	Tested
OTG-AUTHZ-004	Testing for Insecure Direct Object References	Tested
Session Management Testing		
OTG-SESS-001	Testing for Bypassing Session Management Schema	Tested
OTG-SESS-002	Testing for Cookies attributes	Tested
OTG-SESS-003	Testing for Session Fixation	Tested
OTG-SESS-004	Testing for Exposed Session Variables	Tested
OTG-SESS-005	Testing for Cross Site Request Forgery	Tested
OTG-SESS-006	Testing for logout functionality	Tested
OTG-SESS-007	Test Session Timeout	Tested
OTG-SESS-008	Testing for Session puzzling	Tested
Data Validation Testing		
OTG-INPVAL-001	Testing for Reflected Cross Site Scripting	Tested
OTG-INPVAL-002	Testing for Stored Cross Site Scripting	Tested
OTG-INPVAL-003	Testing for HTTP Verb Tampering	Tested
OTG-INPVAL-004	Testing for HTTP Parameter pollution	Tested
OTG-INPVAL-005	Testing for SQL Injection	Tested
OTG-INPVAL-006	Testing for LDAP Injection	Tested
OTG-INPVAL-007	Testing for ORM Injection	Tested
OTG-INPVAL-008	Testing for XML Injection	Tested
OTG-INPVAL-009	Testing for SSI Injection	Tested
OTG-INPVAL-010	Testing for XPath Injection	Tested
OTG-INPVAL-011	IMAP/SMTP Injection	Tested
OTG-INPVAL-012	Testing for Code Injection	Tested
OTG-INPVAL-013	Testing for Command Injection	Tested
OOTG-INPVAL-014	Testing for Buffer overflow	Tested
OTG-INPVAL-015	Testing for incubated vulnerabilities	Tested
OTG-INPVAL-016	Testing for HTTP Splitting/Smuggling	Tested
Error Handling		
OTG-ERR-001	Analysis of Error Codes	Tested
OTG-ERR-002	Analysis of Stack Traces	Tested
Cryptography		
OTG-CRYPST-001	Testing for Weak SSL/TSL Ciphers, Insufficient Transport Layer Protection	Tested
OTG-CRYPST-002	Testing for Padding Oracle	Tested
OTG-CRYPST-003	Testing for Sensitive information sent via unencrypted channels	Tested
Business Logic Testing		
OTG-BUSLOGIC-001	Test Business Logic Data Validation	Tested
OTG-BUSLOGIC-002	Test Ability to Forge Requests	Tested
OTG-BUSLOGIC-003	Test Integrity Checks	Tested
OTG-BUSLOGIC-004	Test for Process Timing	Tested
OTG-BUSLOGIC-005	Test Number of Times a Function Can be Used Limits	Tested
OTG-BUSLOGIC-006	Testing for the Circumvention of Work Flows	Tested
OTG-BUSLOGIC-007	Test Defenses Against Application Mis-use	Tested

Hackcontrol
info@hackcontrol.org

OTG-BUSLOGIC-008	Test Upload of Unexpected File Types	Tested
OTG-BUSLOGIC-009	Test Upload of Malicious Files	Tested
Client Side Testing		
OTG-CLIENT-001	Testing for DOM based Cross Site Scripting	Tested
OTG-CLIENT-002	Testing for JavaScript Execution	Tested
OTG-CLIENT-003	Testing for HTML Injection	Tested
OTG-CLIENT-004	Testing for Client Side URL Redirect	Tested
OTG-CLIENT-005	Testing for CSS Injection	Tested
OTG-CLIENT-006	Testing for Client Side Resource Manipulation	Tested
OTG-CLIENT-007	Test Cross Origin Resource Sharing	Tested
OTG-CLIENT-008	Testing for Cross Site Flashing	Tested
OTG-CLIENT-009	Testing for Clickjacking	Tested
OTG-CLIENT-010	Testing WebSockets	Tested
OTG-CLIENT-011	Test Web Messaging	Tested
OTG-CLIENT-012	Test Local Storage	Tested

HackControl
info@hackcontrol.org

Appendix B. Automated Tools

Scope	Tools Used
Application Security	Acunetix 11 BurpSuite 1.7.30 Owasp-zap Maltego Classic Detectify Sqlmap
Network Security	Nmap Recon-ng Nessus Nexpose

HackControl
info@hackcontrol.org