



If You Can't Beat 'Em Join 'Em

Practical Tips For Running a Successful Bug Bounty Program



Grant McCracken
Shpend Kurtishaj

AppsecEU Rome April 1, 2016

Grant

Technical Account Manager @Bugcrowd

(formerly an ASE)

Before that, Whitehat

Did some traveling

Music





Shpend

AppSec Engineer (ASE) @Bugcrowd

Team Lead

Bugbounty Hunter

Gamer





Bug Bounty Programs









A (Brief) History of Bug Bounty Programs

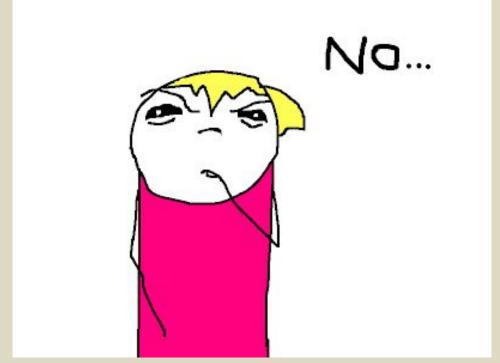




Why?



Do you really want to let people attack you?





Source: http://hyperboleandahalf.blogspot.com/2010_06_01_archive.html

Yes! (They're doing it anyways...)





Source: http://hyperboleandahalf.blogspot.com/2010_06_01_archive.html

Who are these people?

All over the place!

All ages

All levels of experience

All over the world

Users and non-users

Passionate about security





Value

Lots of eyes

Only pay for valid results

Shows a more advanced security posture

Better overall reputation!





How?



How?

Pre-Launch

Scope Focus Exclusions Environment Access



Post-Launch

Managing Expectations
Communicating Effectively
Defining a Vulnerability Rating Taxonomy (VRT)





"Touch the code, pay the bug."





You vs. and Them





Pre-Launch



Scope, scope, scope

Step 0...

Basic resources/requirements to run a program...

Scope defines the researcher's universe

Leave nothing open to interpretation Understand your attack surface The path of least resistance





Focus

You might care about specific:

Targets
Vuln types
Functionalities (e.g. payment processing)

How?

Incentives
Create a focused program





Source: https://xkcd.com/1361/

Exclusions

You might not care about:

(Low-impact) "Low-hanging fruit"
Intended functionality
Known issues
Accepted Risks
Issues resulting from pivoting





Environment

Prod vs. Staging?

Make sure it can stand up to testing!

Scanners

Contact forms

Pentesting requests

Special bounty type? IoT?
Researcher environments?





What a shared environment looks like...





Access

- Easier = better
- Provide researchers with the resources they'll need to be successful (e.g. credit cards, etc).
- No shared creds





Remember...





Post-Launch

Be prepared
 Triage Process
 Communication
 Vulnerability Rating Taxonomy
 Horror Stories
 Success Stories



\$UNPREPARED_COMPANY

Recipe for disaster:

Does not have human resources
Bad/Unclear exclusions
Don't provide known issues
Pays bad rewards





Triage Process

Reproduction Steps

Screenshots

Pocs

NOT Videos (without a supported writeup)





Triage Process

Good report

- 1. Go to url: http://target.com
- 2. Click on "button" x
- 3. Check burp for request z
- 4. Send to repeater
- 5. Modify param p to payload:"><svg/onload=alert(1)
- 6. Send request
- 7. Browse http://target.com/me.php for xss payload

Bad report

- 1. Login
- 2. Make the following request: POST /suppliers/15 HTTP/1.1
- 3. XSS





Triage Process

Check Domain/Bug type if in scope
Check for duplicates
Replication Steps
Have accounts with diff roles ready
Have multiple browsers ready

Keep burp open (you'll need it)
Keep the scope handy
Rename valid bug titles



Communication is Key

Researchers like:

Concise, unambiguous responses
ESL
Quick responses
Predictable time to reward
Stay on top of these issues!





Define a Vulnerability Rating Taxonomy

For you:

Speed up triage process
Track your organization's posture
Arrive at reward amount more quickly

For them (if published):

Focus on high-value bugs Avoid reporting won't fix issues Feel a sense of trust (goes with brief)





Discuss the VRT at a Roundtable

Priority will change as your organization does.

Establish a discussion meeting to:

Review interesting bugs
Discuss additions to VRT
Propose changes to vulnerability classification/priorities
This is an ongoing process!







Diffie-Hellman (DH) key exchange parameters vurnavility

· 09/27/2015

Reference Number

Bug Type SQL Injection

XSS Location URL Empty

Affected Parameter No FS 1 No SNI 2 TLS 1.0 TLS_RSA_WITH_3DES_EDE_CBC_SHA (0xa) No FS

Affected Users ALL

Attack String TLS_RSA_WITH_AES_256_CBC_SHA256 (0x3d) No FS

Browser Empty

Bug URL

Device Empty

HTTP Request

Hello team

I am Wamim , and I am here to report a vulnerability on your SITE! ity : Diffie-Hellman (DH) key exchange parameters

Severity: Medium/High.

Affected site:

Here more information about the vulnerability and the impact: https://weakdh.org

Attack details/Proof of Concept by ssllabs.com(100% affidability) Please, check the full scan here: https://www.ssllabs.com/ssltest/analyze.html?





Tools Used fire bug

Reference Number Bug Type XSS XSS Location URL .js libarary Affected Parameter update your .js libarary Affected Users ALL Attack String \$("").on("error",function(){alert(9)}); Browser Empty Bug URL Device Empty HTTP Request User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:46.0) Gecko/20100101 Firefox/46.0 Accept: image/png,image/*;q=0.8,*/*;q=0.5 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate, br Referer: Cookie: Connection: close Cache-Control: max-age=0 Method of Finding manual Platform Empty Platform Version Empty Proof of Concept Empty Replication Steps 1. install firebug in your firefox 2. execute this command in your console after the opening of the given link of your official page.





Content-Security-Policy header is missing [must patch soon]

hello support,

CSP header is missisng, that the CSP response headers served are missing, but the page without these headers can be cached by server. This makes it easier to mount a XSS attack or injuction attacks.

What is CSP?

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware.

CSP is designed to be fully backward compatible; browsers that don't support it still work with servers that implement it, and vice-versa. Browsers that don't support CSP simply ignore it, functioning as usual, defaulting to the standard same-origin policy for web content. If the site doesn't offer the CSP header, browsers likewise use the standard same-origin policy.

Is there any Risk ?? Yeah offcourse,

The risk with CSP can have 2 main sources:

- 1] Policies misconfiguration,
- 2] Too permissive policies.





Cross-site request forgery, also known as a one-click attack or session riding and abbreviated as CSRF or XSRF, is a type of malicious exploit of a website whereby unauthorized commands are transmitted from a user that the website trusts

- 09/03/2015

Bug Type CSRF

XSS Location URL Empty

Affected Parameter Empty

Affected Users Empty

Attack String Empty

Browser Empty

Bug URL www.google.sn/advanced_search | www.google.sn/intt/fr/chrome/business/devices/tco.html | www.google.sn/intt/fr/contact | www.google.sn/intt/fr/coducts/products/productivity-tools/classroom/index.html | www.google.sn/intt/wo/contact | www.google.sn/intx/wo/work/search/products/gss.html | www.google.sn/movies | www.google.sn/trends/explore

Device Empty



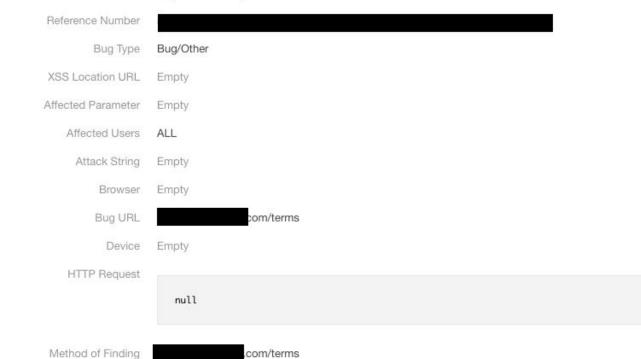




Terms and Conditions missing severability clause

0% • 11/17/2015

Terms and Conditions missing sever ability clause.





Success Stories

Instructure

	2013 (Pentest)	2014 (Bug Bounty)
Critical	0	0
High	1	25
Medium	1	8
Low	2	16

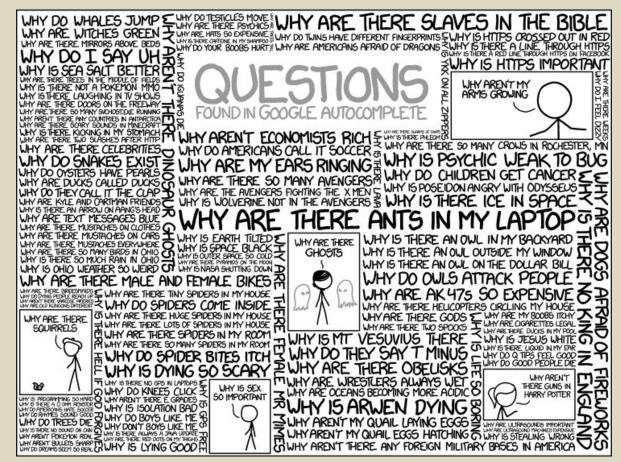


Source: https://www.canvaslms.com/security

tl;dr









Source: https://xkcd.com/1256/