Secrets of Google VRP

The bug hunter's guide to sending great bugs

Krzysztof Kotowicz, Google Security Team

Introduction

Who are we?

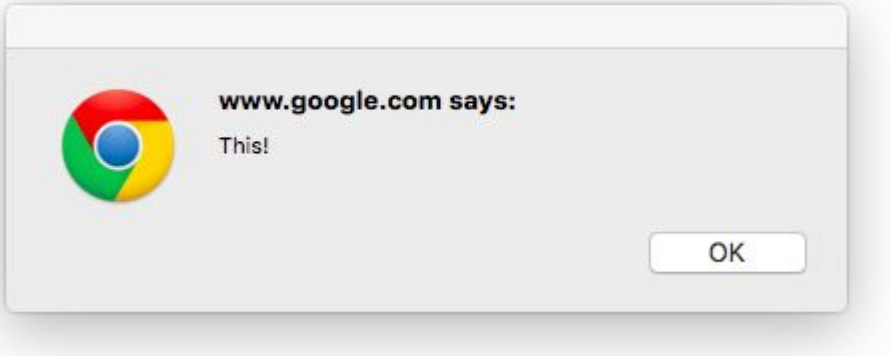
Google Security Team - we handle the submissions to Google VRP

What makes us happy?

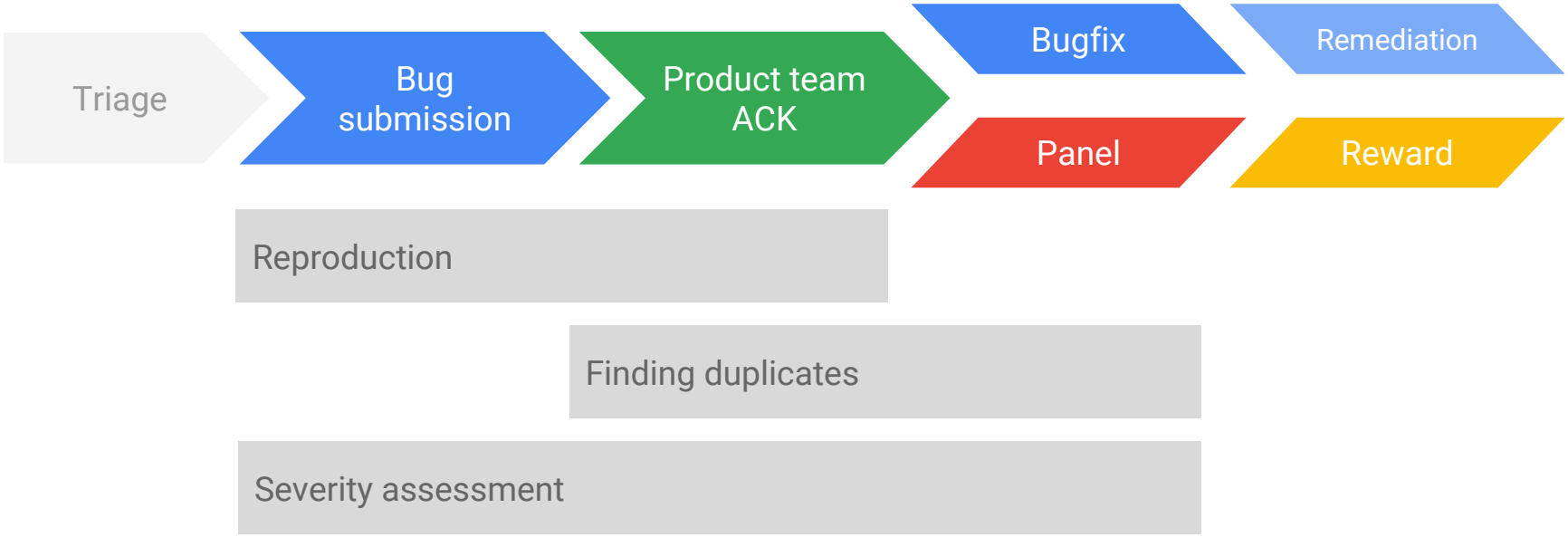
Who are we?

Google Security Team - we handle the submissions to Google VRP

What makes us happy?



Life of a VRP report



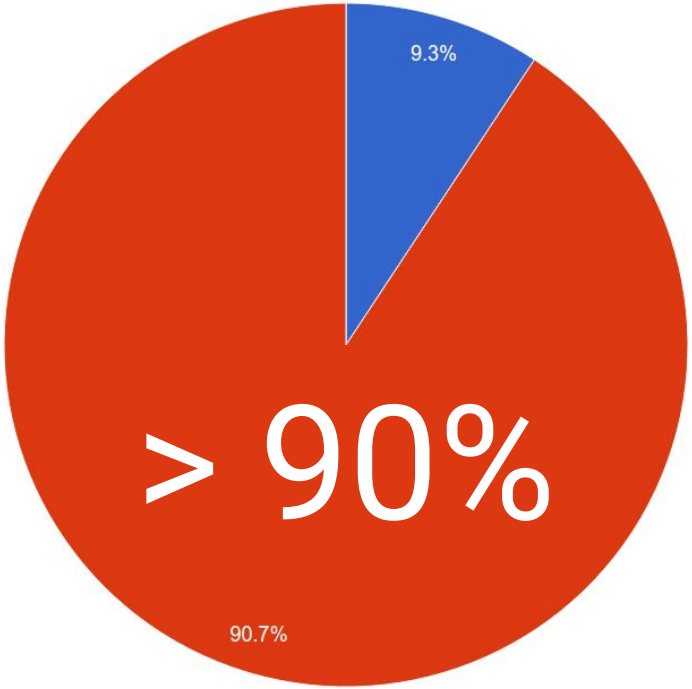
Life of a VRP report

We handle >150 vulnerability reports every week:

- Read
- Understand (Is it in scope? What's the impact?)
- Reproduce (find the application, set up the accounts)
- Create a proof-of-concept
- File the bug and explain it to the product team
- Check if this is duplicate
- Communicate with the bughunter

6 min 15s

per incoming report



do not result in bugs

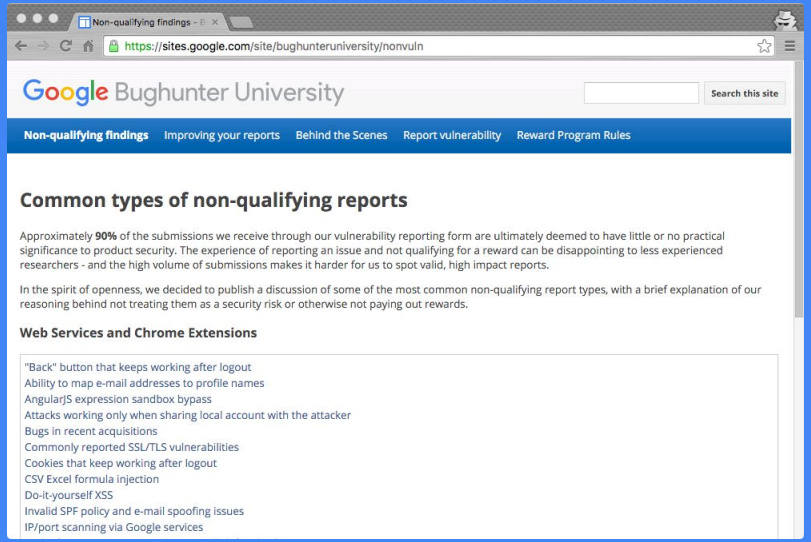
Most popular issues

- foobar@gmail.com receives e-mails to foo.bar@gmail.com
- XSS in translate.googleusercontent.com

Both are invalid :/

Visit the **Bughunter University** to check other common non-vulnerabilities!

https://g.co/bughunteruniversity



The greater picture

How much do we pay for an XSS?

It varies:

- The panel evaluates the **impact** first and finds the worst-case scenario
- We look for the root cause, so report may turn out to be a duplicate
- We might pay more for unusually clever or severe vulnerabilities
- We might pay less for unusual user interaction required
- There's a bug chain bonus

Might be **$5000** for an XSS on www.google.com or a **$100** in an acquisition

Which bug has more impact?

Mixed content

Remote code execution

Which bug has more impact?

Mixed content
on accounts.google.com

Remote code execution
on blog.fooacquisition.co.uk

Vulnerability impact

What can the attacker do having exploited the vulnerability?

- Type of the application
- Number of users affected
- Difficulty of exploitation
- Requirements for the attack
  - Victim interaction?
  - Shared computer use?
- Effect after the attack

Vulnerability impact

Look beyond the vulnerability types!

- Not every XSS, CSRF is the same
- Not every application is equally important
- Is the attack useful for the attacker or is is just annoying for the victim?

Testing on a small scale

Some heuristics are tuned to a different adversary

- Botnet owner hijacking accounts vs. your sister
- Tradeoffs are unavoidable
- Mitigations might not trigger when pentesting - it works as intended
- Common examples:
  - Account recovery of test accounts
  - reCAPTCHA bypass
- If in doubt, send it and we'll investigate
- Most of the times it turns out it's not a bug

Choose your target

Google product

- Bugs are harder to find
- Unique technology stacks
- Defense in depth
- There's a lot of products!
- Bigger payouts
  - Big impact
  - A lot of users
- Tips:
  - Don't go after the obvious
  - Look at older apps & rare features

Acquisition

- Bugs are easier to find
- Standard software stacks
  - Wordpress, Apache, PHP, Rails
- A lot of duplicate issues
- Much lower rewards
- Often does not meet the bar for credit
  - Not integrated
  - Few users
  - Non-sensitive websites
  - Soon to-be-deprecated

In-scope or out of scope?

Spam / phishing / social engineering vectors - out of scope

- We know it's possible. We know phishing works!
- Text injection / Content spoofing
- Sending e-mails from *noreply@google.com*
- Open redirect to a phishing site
- RFD, Make your victim execute the downloaded file
- SPF / DKIM - we know and are working on it. Stay tuned.

In-scope or out of scope?

Local attacker (sharing OS user account) - out of scope

- It's impossible to protect!
- If the attacker uses your browser, it's game over anyway
- Cached content after logout, recovering passwords from Password Manager, cookies working for a brief time after logout, reusing tokens from HTTPS streams, malicious files downloaded, backdoored browser extensions...

In-scope or out of scope?

Domain names, acquisitions, vendor websites - check whois, DNS. **Ask** if in doubt.

```
$ whois nest.com | grep -i google
Registrant Organization: Google Inc.
Registrant Email: dns-admin@google.com
```

```
$ whois community.nest.com | grep -i google
```

```
$ host connect.googleforwork.com
connect.googleforwork.com is an alias for google-partner-customer.hosted.jivesoftware.com.
google-partner-customer.hosted.jivesoftware.com has address 204.93.79.169
google-partner-customer.hosted.jivesoftware.com mail is handled by 10 mx1-in.phx1.jivehosted.com.
$ whois 204.93.79.169 | grep -i google
```

Common out of scope - **zagat.com, polarb.com, community.nest.com**

Creating the vulnerability report

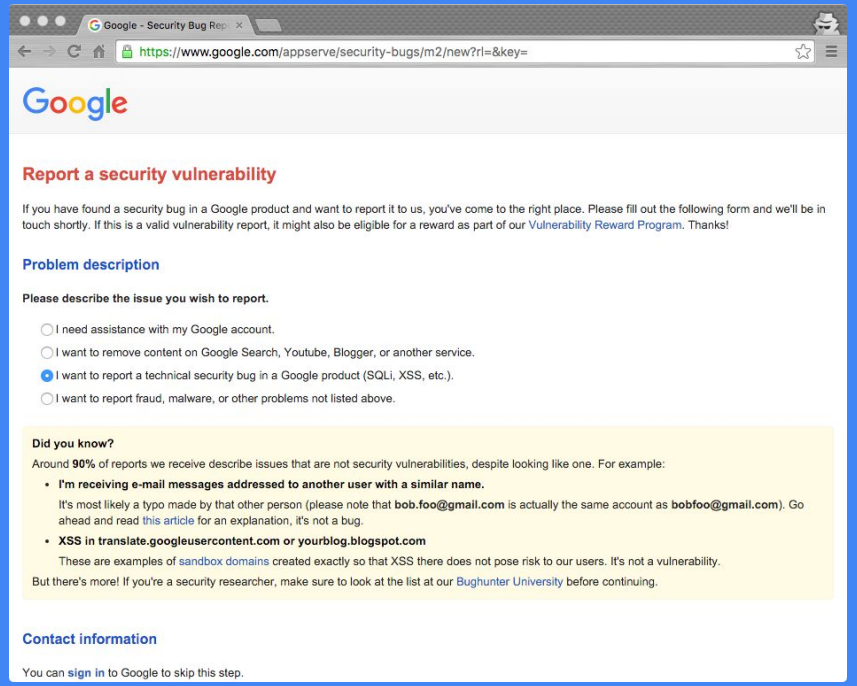
Submission form

https://goo.gl/vulnz

Faster than security@google.com

Tips for the first-timers

After the submission, we'll talk over e-mail



Reproduction steps

- Google launches new products or features daily.
  You might know the application better than the Security Team!
- Specify the target
  - URL, application name, Android package name
  - Specify the elements to click
  - Essential for more complex issues
- If multiple users involved in the attack, please say so.

Reproduction steps

I just found a big clickjacking bug on your website google.com, your data can be fully modified without any permissions!

If you really want to know please give a website link, I will show you how it affects you.

1. Go to www.google.com and sign in.

2. Browse to https://www.google.com/preferences?igu=2

3. Note that the server does not issue an X-Frame-Options header when the igu=2 parameter is present.

4. Also note that this page displays a Sign In button, but any changes made to these settings will affect the signed-in user's account.

Attack scenario

Explain **who** can exploit the vulnerability, **what is required** to trigger it, and **what the attacker gains** afterwards.

"With this critical vulnerability I can execute code."

"A regular user of Google Foobar may escalate privileges to become a Google Foobar group administrator. The other Foobar group administrator needs to click the link given by the attacker."

Videos and screenshots

- We prefer text, but if it helps - go ahead
- Not a replacement for written **reproduction steps**
- Make it short
  - Once, we got a >1hr video. Hillarious, but not a bug :/
- Don't write in Notepad, keep the writeup separate
  - Hard to follow, slow and impossible to copy-paste
- Upload an **unlisted** video, share the URL

OWASP top 10 and beyond

A1 - Injection

What are you injecting? Where are you injecting it?

- Text snippet - usually not a problem
- XML - XXE attack
  - Billion laughs attack.
  - Verify - sometimes it's not a problem (hard limit for entity expansion - e.g. up to 10000)
- SQL injection
  - Google products rarely use SQL databases
  - Automated tools (sqlmap) are often wrong
  - Verify the issue manually, especially a blind SQLi
- LDAP injection
  - Highly unlikely
- Think outside the box!

A1 - Injection

I used a tool that detected a blind SQL injection in http://foobar.google.com/citations?view_op=new_profile&hl=%Inject_Here%en

List of tables extracted from the DB:

sb_host_adminAffichage1name Affichage1name yhmm yhmAffichage1 sb_host_adminAffichage1 ststaffFusion Fusion8 Cold ColdFusion ColdFusion8 slserver server bayview bayviewpath edu mac windows pc win Affichage1Affichage1edu pierre liste Lieux "Lieu de la manifestation" Pr�f�rencesUtilisateur "Nombre total de membres"

If a tool found the vulnerability - **verify it**!

Blind SQL exploitation tools often cause false positives.

The table list looks random - it's a dictionary from sqlmap.

It's highly improbable there's a "Coldfusion8", "Lieu de la manifestation", and "Nombre total de membres" columns in the table.

A1 - Injection

Command injection

We can include any command in the below URL format:
https://console.cloud.google.com/home/dashboard/project=;sudo%20id

Let's assume the attacker sends the above URL to the victim. Once the victim accesses the URL and he clicks "Activate cloud shell", the command executes.

Found by S.Venkatesh, @pranavvenkats

- Clear reproduction steps
- Simple attack scenario
- Bash script created client-side in JS
- Requires user interaction (click)
  - we found a better vector internally

$6000
(5000 + 1000 for thinking outside the box)

A2 - Broken authentication & session management

- Token timeouts
- Token invalidation upon password change
- User enumeration attacks
  - To some extent - unavoidable
- Cookies working after logout
  - A known issue
- Account recovery
  - It's hard to test on a small scale
  - Attempts done on test accounts created from the same location
  - Often turns out to be invalid

A2 - Broken authentication & session management

The session cookies are not expired when the user is logged out of his account. This will lead to Session Misconfiguration. The Attacker will be able to use the user cookies/sessions to log into his account whenever he wants.

The cookies/session must expire when the user is logged out of his account and every time new cookie/session will be provided to the user logging in.

We are aware of this issue. Session cookies are valid for a varying amount of time after the logout.

It's *not a vuln* - in order to access the cookies, the attacker needs **physical access** to the targeted login session.

A3 - XSS

- Where does it fire?
  - Use *alert(document.domain)* instead of *alert(1)*
- Is the domain sandboxed?
  - See the sandbox domain list on BHU list
  - Are there interesting cookies there?
- Is user interaction required?
- Are users typing the payload themselves?
- Angular sandbox, CSP, browser XSS filters
  - they are not **security boundaries**
  - Assume there is a CSP and Angular sandbox bypass and no in-browser XSS filter

A3 - XSS

1. Go to Youtube and create an annotation with a link on a video. Link to any Google Plus-page.

2. Capture the request, use the following payload for the URL:

javascript://plus.google.com/?%0aalert%28document.domain%29//

3. Save and look at the video. Then press the annotation.

XSS on www.youtube.com

Found by Frans Rosén, @fransrosen

Full writeup

Clear reproduction steps!

Regex patterns validated server side

For plus.google.com URLs the protocol was not validated :)

//plus.google.com?alert(document.domain)

A4 - Insecure direct object reference

- What is the data you're getting via IDOR?
  - Is it public information?
- How long is the reference in the parameter?
  - Can you bruteforce it before the heat death of the sun?
- Is the parameter random, or sequential?
- Sometimes it’s by design
  - E.g. unguessable URLs
  - Mitigated by the parameter length and randomness
- You need to trust the user who you're sharing with
  - Whoever you're sharing the URL with, can just copy the contents

A4 - Insecure direct object reference

There's an IDOR vulnerability on:

https://android-review.googlesource.com/changes/201130/detail?O=10004

If you change the O parameter value you'll receive e-mail and names of other users - e.g.

```
"owner": {
   "_account_id": 1084268,
   "name": "Shubham Ajmera",
   "email": "shubhamajmera@google.com",
 }
```

That's a list of authors committing to an **open-source** project. That information is **public**.

A list of those can be easily extracted off git:

```
$ git clone https://android.googlesource.com/platform/libcore
$ cd libcore
$ git shortlog -sne
```

A5 - Security misconfiguration

- Directory listing
  - Depends on what's in the directory
- Ancient, not exploitable issues - not a vuln
  - TRACE method enabled / XST
- Information leak
  - Point out what information - credentials, debug information, cookie values (HTTPOnly bypass?)
- Various defense in depth mechanisms (CSP, HSTS, public key pinning, ...)
  - We know and are getting there (and trust us, it IS complex)
  - Lack of e.g. CSP header is usually not a vuln in itself
  - Unless we e.g. advertise it or there's an evident flaw/bypass

A5 - Security misconfiguration

Debug page is accessible at https://booboo.google.com/debug. It allows the visitor to see the service credentials:
- login + password to the API server
- error log details

The screenshot of the page is attached below.

Clear mention of what data is available

Easily reproductible

Screenshot to quickly determine if it's valid

A5 - Security misconfiguration

Directory listing is enabled at www.example.com/wp-content/images

Screenshot with a list of jpg, gif, etc. files.

No security impact - these are just static files, publicly accessible.

Missing viable attack scenario.

Would be valid if there are log files, database backup files available.

A6 - Sensitive data exposure

- Most common issue? TLS configuration on www.google.com
  - BEAST, POODLE, CRIME, DROWN
  - Weak ciphers
- We have countermeasures. Most of the times, it's just a bad report from an automated tool
- Not every website is HTTPS yet. We're working on it.
- Referrer leaks, sensitive data on HTTP
  - Rare, but they happen
  - Mostly in acquisitions

A6 - Sensitive data exposure

www.google.com and www.youtube.com still support SSLv3, making the website vulnerable to POODLE SSL attack.

You announced SSLv3 and RC4 deprecation and then you still offer them!

We are aware of it. We can't simply switch SSLv3 off support overnight.

Google has to support a wide range of clients, like old mobile phones, so big changes like this require careful roll-out. Meanwhile, we will never offer SSLv3 or RC4 to a client if more secure options are available.

A6 - Sensitive data exposure

www.example.com/reset_password loads external widgets - e.g. a Facebook like button:

<script src="https://facebook.com/widget.js"></script>

The password reset token is sent to those 3rd party websites in a Referrer header.

A valid bug. A sensitive token value should not be sent to external websites.

Hall Of Fame (it's an acquisition)

A7 - Missing function level access control

- Multiple roles in the application, one role allows for more than described
  - User Management Admin vs. Billing Admin vs Admin
  - Please link to the documentation where the roles are described
- When users are downgraded, their tokens are still valid
  - How long are the tokens valid for?
  - Please retry a few times
- How useful is it in an attack?
  - Changing a language
  - Re-adding yourself to the admin group
  - Accessing password recovery tokens of other users

A8 - Cross Site Request Forgery

- Make sure the action triggered is interesting from a security perspective
  - CSRF a contact support form
  - CSRF sharing a Google Doc
- Think like an attacker
- Token does not expire
  - If you can read the token, you can do so much more!
- Token does not change the value
  - If you can read the token, you can do so much more!
- CSRF logout - a known issue

A9 - Components with known vulnerabilities

- Out of date software
  - e.g. Wordpress plugins
  - Specify how did you fingerprint the version (e.g. a README file exposed)
  - Links to CVEs help
- Just disclosing a version number / server banner is not a bug
- Bugs exploitable only on old browsers
  - If the browser vendor deprecates it, there are no security patches
  - Instead of e.g. an XSS, there's probably a browser exploit available

A9 - Components with known vulnerabilities

Wordpress installation at http://acquisition.example.com/ uses an outdated plugin FooBar 1.47:
http://acquisition.example.com/wp-admin/plugins/foobar/README.txt

This plugin has an SQL injection vulnerability described here: http://blog.researcher.com (CVE-2015-0101) - it was patched in 1.51.

Actionable - we can confirm the version, see the vulnerability details and understand the impact.

No need for PoC if it requires you to have an account in Wordpress installation.

$100.00 / Hall Of Fame - depending on a site and the vuln details.

A9 - Components with known vulnerabilities

An attacker can run arbitrary code at www.foo.com.

An old Wordpress installation there is vulnerable to SQL injection. I extracted the password hashes and bruteforced the admin password - it's "IDDQD".
I logged to the admin panel and uploaded a PHP shell. Here is a zip file with other files on this server.

Stop at confirming the SQL injection, or even just give us the vulnerable version number and link to vuln descriptions.

Don't exploit the vulnerabilities without prior permissions from us!

It's a **valid bug**, we investigate and pay for maximum impact.

A10 - Unvalidated Redirects and Forwards

- Open redirect to **http[s]://** is not an issue
  - Caveat: OAuth
- It is useful for triggering other vulnerabilities
  - Angular ng-include
  - CSP bypass
  - Referrer check bypass
  - URL whitelists bypass
- Open redirect to **javascript:** or **data:** - report it
- Use it in an exploit chain!
  - https://www.google.com/url?sa=D&q=http://evil.com

A10 - Unvalidated Redirects and Forwards

The ServiceNotAllowed page appears when you attempt to access a Google app service that has not been configured for your domain. It requires that you are logged into at least two accounts and will give you a form to switch accounts. When you select an account via the bullet on the page, it executes JavaScript to redirect your browser.

https://admin.google.com/foobar.com/ServiceNotAllowed?service=grandcentral&continue=javascript:alert(document.cookie);//

Found by Brett Buerhaus.

Full writeup, including an impact and attack scenario.

Open-redirect resulting in **XSS on admin.google.com**

$5000

OWASP top 10

- This and other lists are a great resource for classifying vulnerabilities
- They don't tell the whole story
- The impact of a given vulnerability is application-specific
- Google VRP panel evaluates the impact

To profit from the VRP:

- Understand the application(s)
- Focus on finding vulnerabilities useful in an attack
- Don't blindly scan for **insert type here*** - think!

Summary

Summary

To submit great bugs:

1. Find a vulnerability - you all know how to do it :)
2. Evaluate it
   a. Visit **Bughunter University**
   b. Look **beyond the vulnerability type**
   c. Assess the **impact**
3. Report the bug at **goo.gl/vulnz**
   a. Think about a sensible **attack scenario**
   b. Prepare clear **reproduction steps**
4. Profit!

Good luck on your future bughunting!