

---

## AllPocsFromHackerOne

Contact me on



This script grabs public report from hacker one and download all JSON files to be greppable

The main goal is make easy categorize vulns by technique

**Would you have a suggestion?** Please open it on issues tab =)

I would love hear from you. # TOP 20 Weakness from HackerOne disclosed Reports ### From 9k disclosed reports

```
1    1019 Information Disclosure
2    915 Cross-site Scripting (XSS) - Generic
3    723 Violation of Secure Design Principles
4    610 Improper Authentication - Generic
5    416 Cross-Site Request Forgery (CSRF)
6    415 Cross-site Scripting (XSS) - Stored
7    357 Denial of Service
8    324 Cross-site Scripting (XSS) - Reflected
9    320 Privilege Escalation
10   314 Memory Corruption - Generic
11   293 Improper Access Control - Generic
12   261 Open Redirect
13   226 Code Injection
14   198 Business Logic Errors
15   197 SQL Injection
16   186 Command Injection - Generic
17   169 Insecure Direct Object Reference (IDOR)
18   165 Server-Side Request Forgery (SSRF)
19   165 Cryptographic Issues - Generic
20   130 Path Traversal
```

## All Categorized Vulns

Allocation of Resources Without Limits or Throttling

Array Index Underflow

Authentication Bypass Using an Alternate Path or Channel

Brute Force

Buffer Over-read

---

Buffer Underflow

Buffer Under-read

Business Logic Errors

Classic Buffer Overflow

Cleartext Storage of Sensitive Information

Cleartext Transmission of Sensitive Information

Client-Side Enforcement of Server-Side Security

Code Injection

Command Injection - Generic

CRLF Injection

Cross-Site Request Forgery (CSRF)

Cross-site Scripting (XSS) - DOM

Cross-site Scripting (XSS) - Generic

Cross-site Scripting (XSS) - Reflected

Cross-site Scripting (XSS) - Stored

Cryptographic Issues - Generic

Denial of Service

Deserialization of Untrusted Data

Double Free

Embedded Malicious Code

Execution with Unnecessary Privileges

Exposed Dangerous Method or Function

Externally Controlled Reference to a Resource in Another Sphere

Failure to Sanitize Special Elements into a Different Plane (Special Element Injection)

File and Directory Information Exposure

Forced Browsing

Heap Overflow

HTTP Request Smuggling

---

HTTP Response Splitting

Improper Access Control - Generic

Improper Authentication - Generic

Improper Authorization

Improper Certificate Validation

Improper Check or Handling of Exceptional Conditions

Improper Export of Android Application Components

Improper Handling of Insufficient Permissions or Privileges

Improper Handling of URL Encoding (Hex Encoding)

Improper Input Validation

Improper Neutralization of Escape, Meta, or Control Sequences

Improper Neutralization of HTTP Headers for Scripting Syntax

Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)

Improper Null Termination

Improper Privilege Management

Inadequate Encryption Strength

Incorrect Authorization

Incorrect Calculation of Buffer Size

Information Disclosure

Information Exposure Through an Error Message

Information Exposure Through Debug Information

Information Exposure Through Directory Listing

Information Exposure Through Sent Data

Insecure Direct Object Reference (IDOR)

Insecure Storage of Sensitive Information

Insecure Temporary File

Insufficiently Protected Credentials

Insufficient Session Expiration

---

---

Integer Overflow

Integer Underflow

Key Exchange without Entity Authentication

LDAP Injection

Malware

Man-in-the-Middle

Memory Corruption - Generic

Misconfiguration

Missing Authentication for Critical Function

Missing Authorization

Missing Encryption of Sensitive Data

Missing Required Cryptographic Step

Modification of Assumed-Immutable Data (MAID)

NULL Pointer Dereference

Off-by-one Error

Open Redirect

OS Command Injection

Out-of-bounds Read

Password in Configuration File

Path Traversal

Phishing

Plaintext Storage of a Password

Privacy Violation

Privilege Escalation

Reliance on Cookies without Validation and Integrity Checking in a Security Decision

Reliance on Reverse DNS Resolution for a Security-Critical Action

Reliance on Untrusted Inputs in a Security Decision

Remote File Inclusion

---

Resource Injection

Reusing a Nonce, Key Pair in Encryption

Security Through Obscurity

Server-Side Request Forgery (SSRF)

Session Fixation

SQL Injection

Stack Overflow

Time-of-check Time-of-use (TOCTOU) Race Condition

Type Confusion

UI Redressing (Clickjacking)

Unprotected Transport of Credentials

Unrestricted Upload of File with Dangerous Type

Unverified Password Change

Use After Free

Use of a Broken or Risky Cryptographic Algorithm

Use of a Key Past its Expiration Date

Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG)

Use of Externally-Controlled Format String

Use of Hard-coded Credentials

Use of Hard-coded Cryptographic Key

Use of Hard-coded Password

Use of Inherently Dangerous Function

User Interface (UI) Misrepresentation of Critical Information

Violation of Secure Design Principles

Weak Cryptography for Passwords

Weak Password Recovery Mechanism for Forgotten Password

Write-what-where Condition

XML Entity Expansion

---

XML External Entities (XXE)

XML Injection

## Requirements

### Gron

```
1 go get -u github.com/tomnomnom/gron
```

### JQ

```
1 apt install jq
```

### Tree

### Weakness

All weakness categorized ## jsonReports All json files from disclosed reports from hackerone. Already downloaded. ## reportLinksHackerOne file

All ids from hackerOne disclosed reports ## Utils Folder ### searchIntoJson.sh (gron required) Script helping you finding keys and values into JSON ### buildRepo.sh Do your own jsonReports folder, downloading all disclosed reports from hackerone