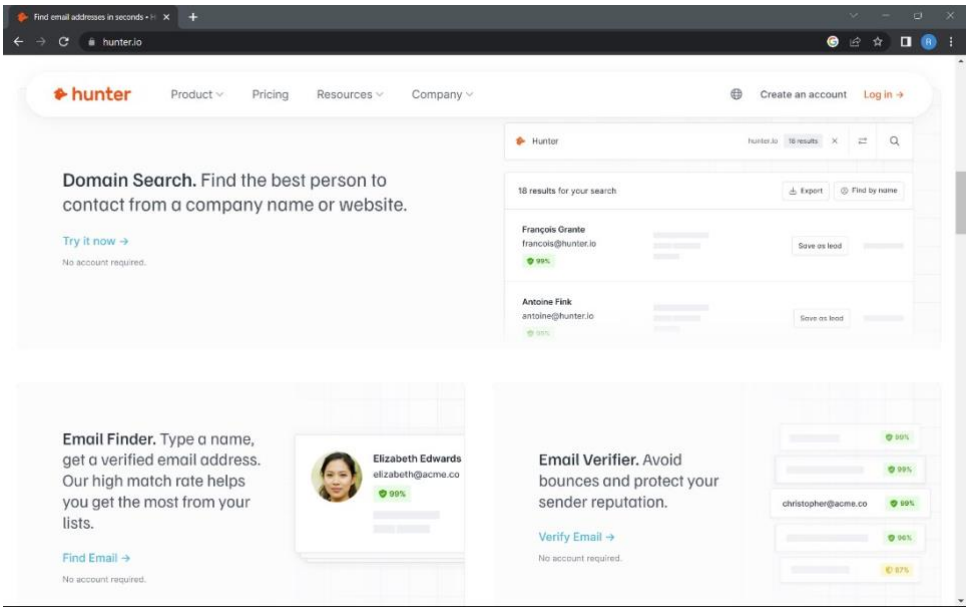
Practical 1: OSINT (Open source Intelligence)

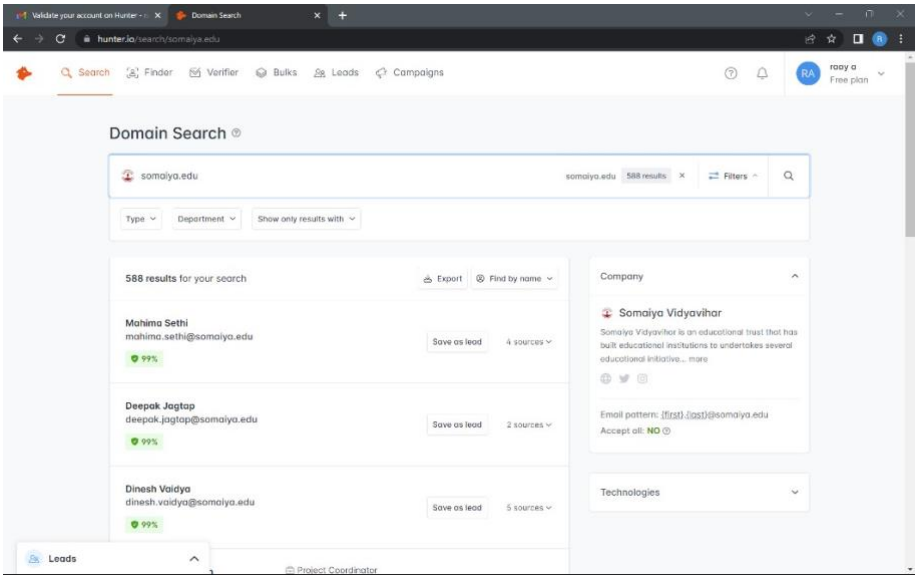
The OSINT framework is a methodology that integrates data, processes, methods, tools and techniques to help the security team identify information about an adversary or their actions quickly and accurately. An OSINT framework can be used to: Establish the digital footprint of a known threat.

OSINT framework focused on gathering information from free tools or resources. The intention is to help people find free OSINT resources. Some of the sites included might require registration or offer more data for $$$, but you should be able to get at least a portion of the available information for no cost.

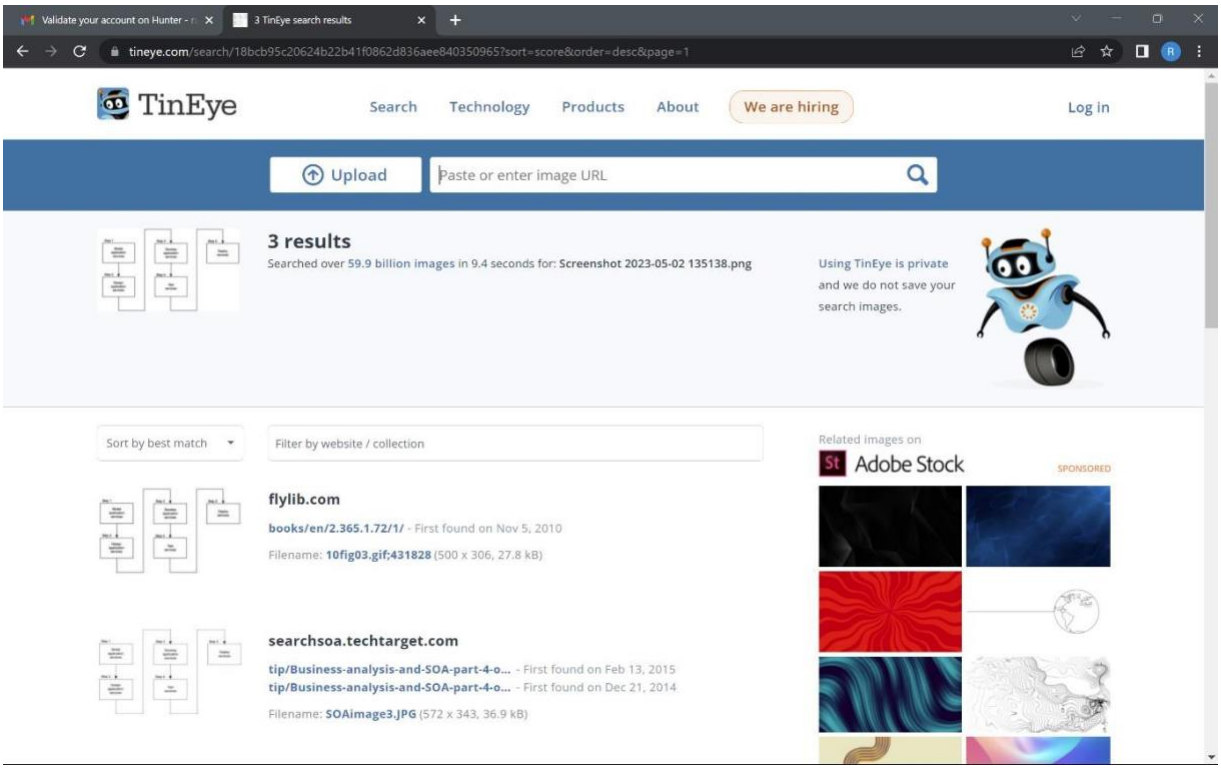
Examples for OSINT:

1. Hunter.io



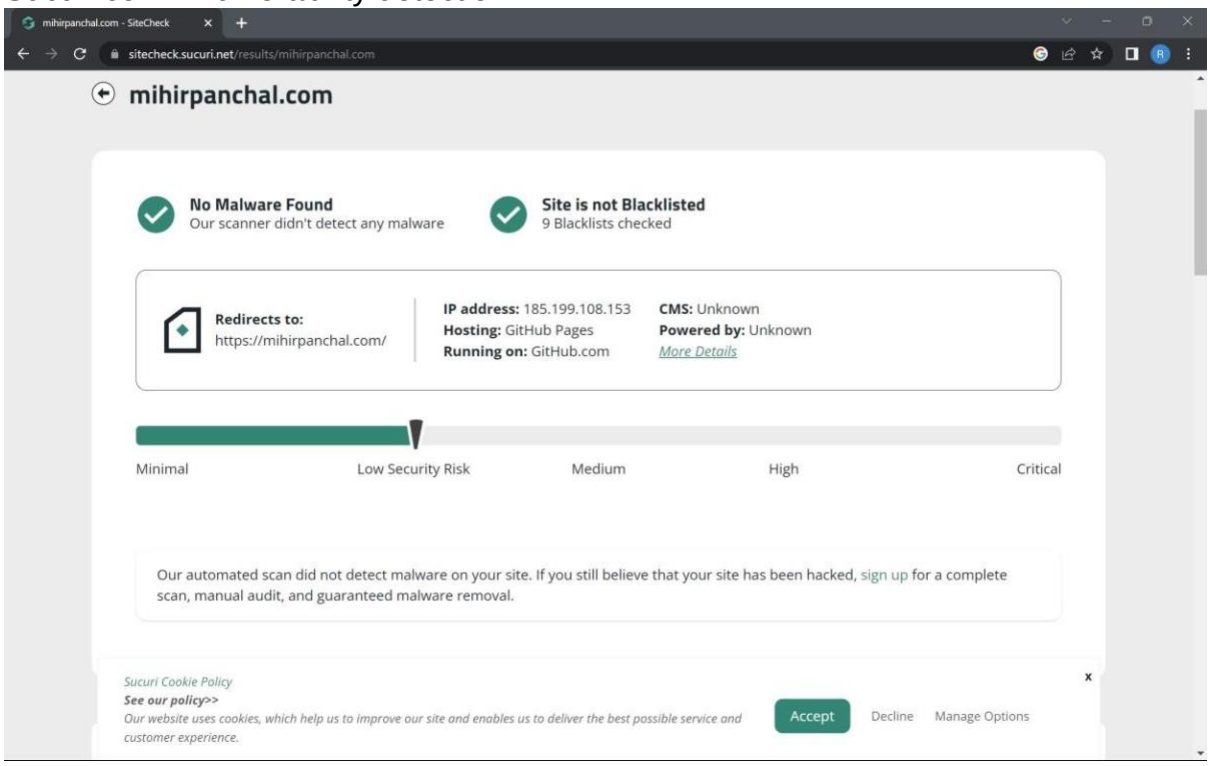


2. Tineye.com
   Used for reverse search of images

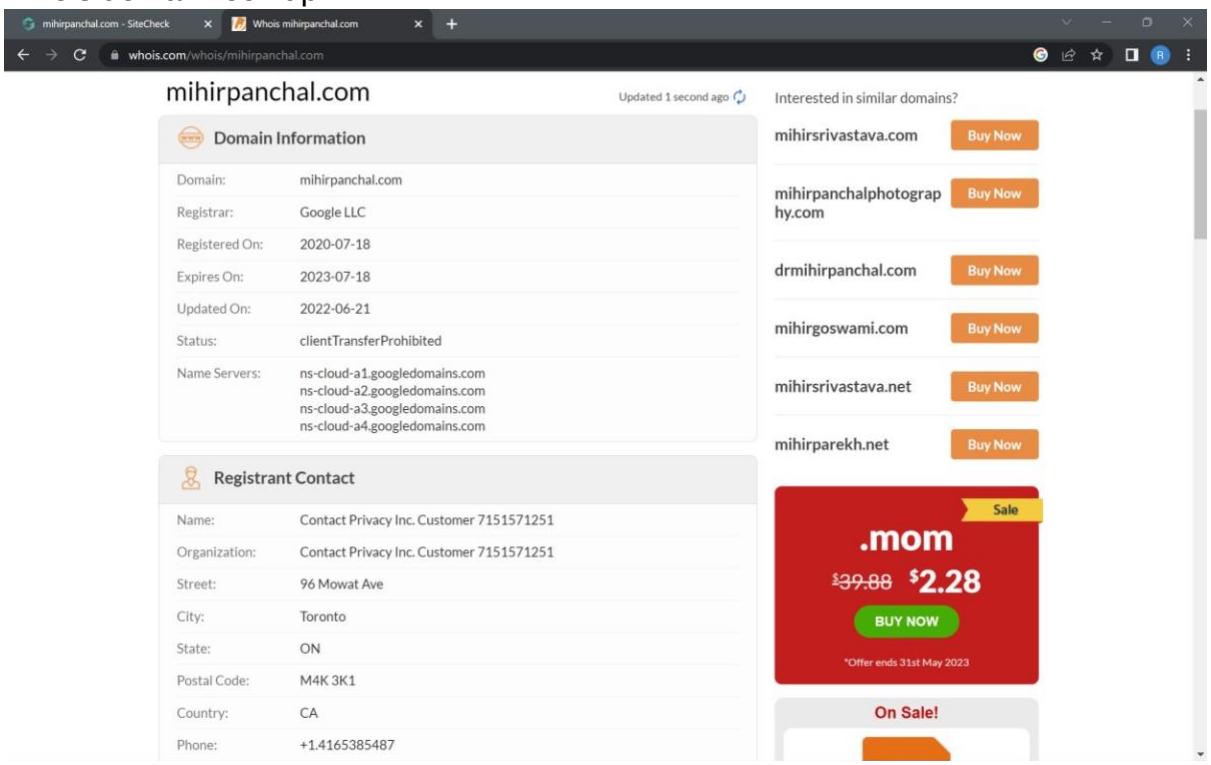


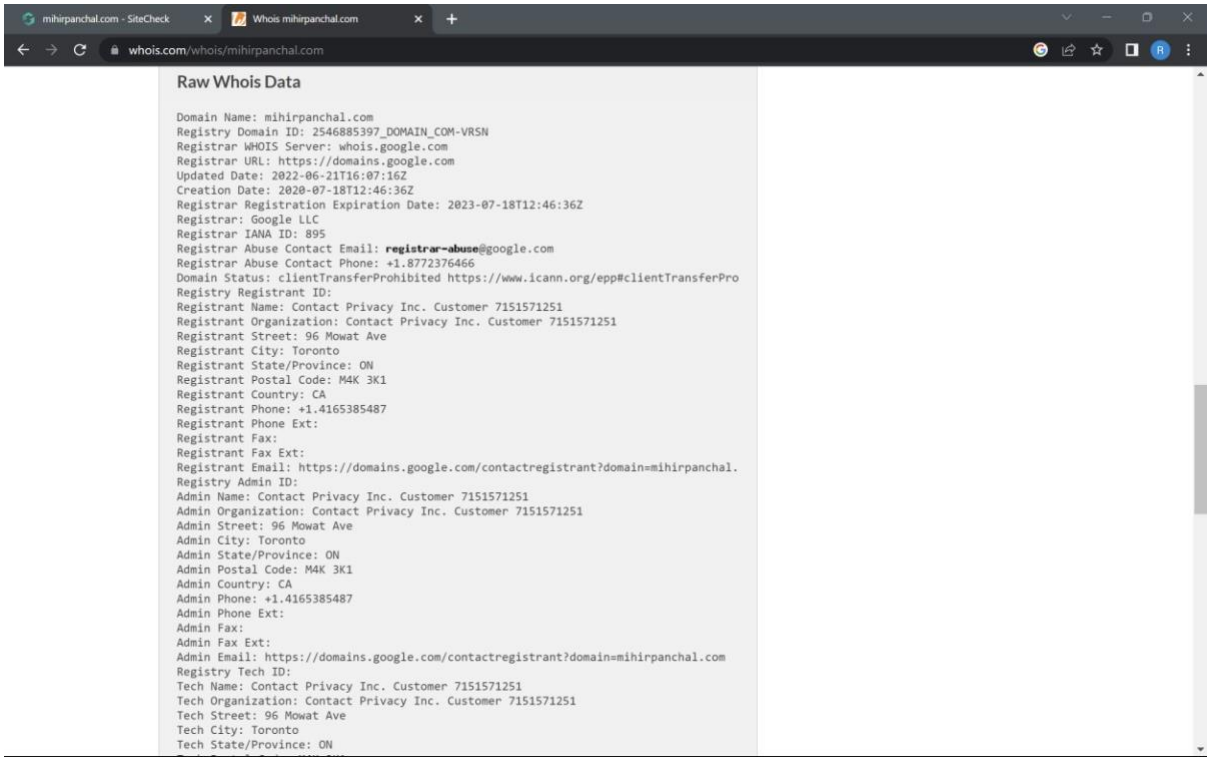
3. Pimeyes.com
   Used for facial reverse search of images

4. Sucuri.com – vulnerability detection



5. Whois domain look up



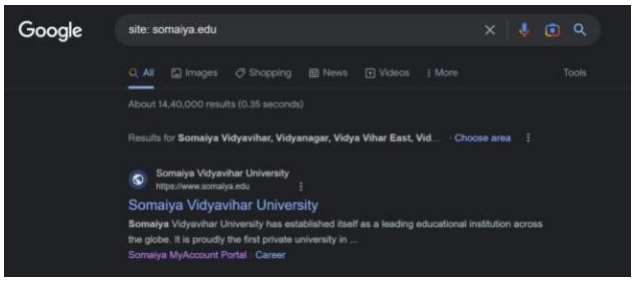


Practical 2: To obtain specific results using google dorks

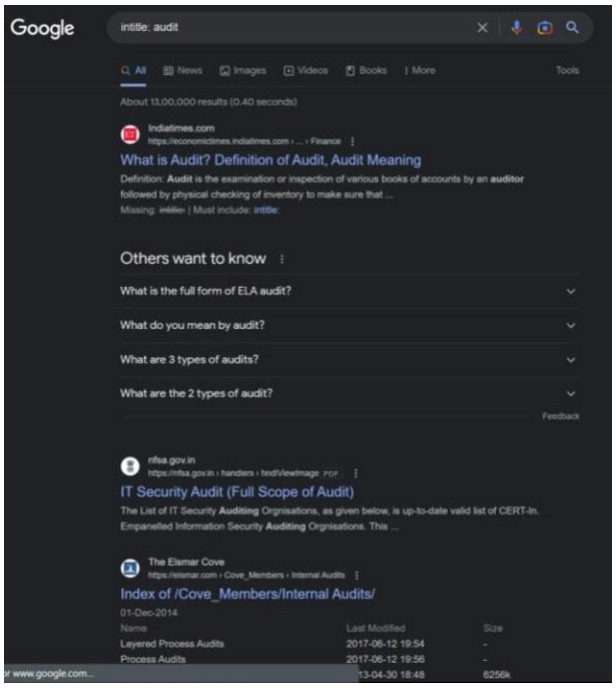
Google dorking is a passive attack or hacking method involving the use of a custom query.

Some examples of commonly used dorks are:

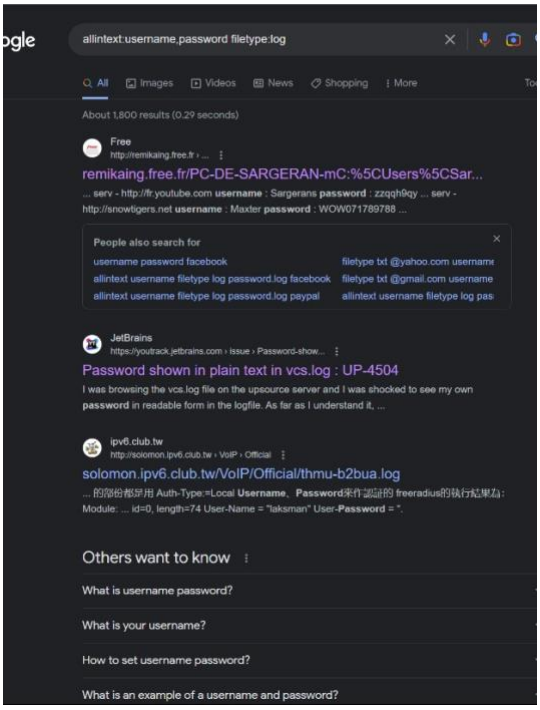
a. Site:____(site name)



b. Intitle:

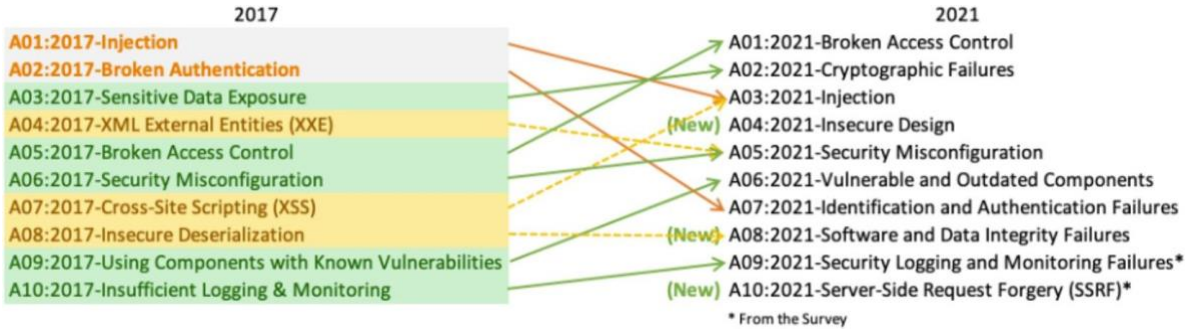


c. allintext: password filetype:log after:2020



Practical 3: OWASP Top 10

The OWASP Top 10 is a non-profit organization whose work is focused on improving software security. It finds most dangerous security threats to web applications and rank them according to their frequency of occurrence and severity of impact. These are the OWASP top 10 vulnerabilities that every developer should look out before proceeding with the development.



1. Broken access control:
   Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits.

2. Cryptographic failures:
   A cryptographic failure is a critical web application security vulnerability that exposes sensitive application data on a weak or non-existent cryptographic algorithm. Those can be passwords, patient health records, business secrets, credit card information, email addresses, or other personal user information.

3. Injection:
   Some of the more common injections are SQL, NoSQL, OS command, Object Relational Mapping (ORM), LDAP, and Expression Language (EL) or Object Graph Navigation Library (OGNL) injection. The concept is identical among all interpreters. Source code review is the best method of detecting if applications are vulnerable to injections. Automated testing of all parameters, headers, URL, cookies, JSON, SOAP, and XML data inputs is strongly encouraged. Organizations can include static (SAST), dynamic (DAST), and interactive (IAST) application security testing tools into the CI/CD pipeline to identify introduced injection flaws before production deployment.

4. Insecure design:
   Insecure design is a broad category representing different weaknesses, expressed as “missing or ineffective control design.” Insecure design is not the source for all other Top 10 risk categories. There is a difference between insecure design and insecure implementation. We differentiate between design flaws and implementation defects for a reason, they have different root causes and remediation. A secure design can still have implementation

defects leading to vulnerabilities that may be exploited. An insecure design cannot be fixed by a perfect implementation as by definition, needed security controls were never created to defend against specific attacks. One of the factors that contribute to insecure design is the lack of business risk profiling inherent in the software or system being developed, and thus the failure to determine what level of security design is required.

5. Security misconfiguration:
   Security misconfigurations are security controls that are inaccurately configured or left insecure, putting your systems and data at risk. Basically, any poorly documented configuration changes, default settings, or a technical issue across any component in your endpoints could lead to a misconfiguration.

6. Vulnerable and outdated components:
   If the software is vulnerable, unsupported, or out of date it is vulnerable. This includes the OS, web/application server, database management system (DBMS), applications, APIs and all components, runtime environments, and libraries.

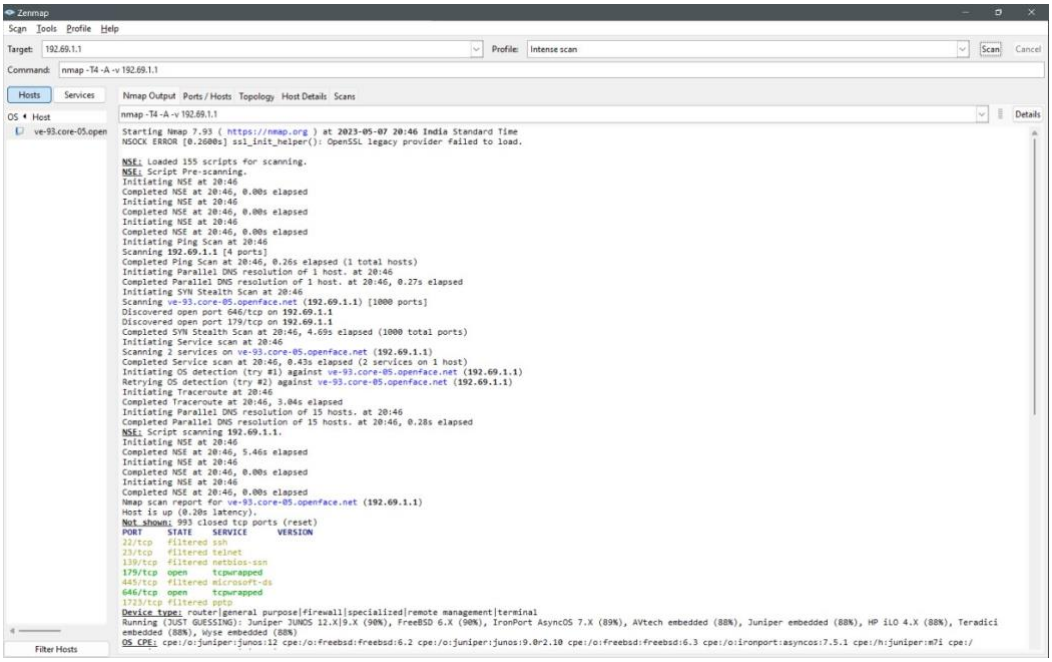
7. Identification and authentication failures:
   Confirmation of the user's identity, authentication, and session management is critical to protect against authentication-related attacks. There may be authentication weaknesses if the application;
   - Permits automated attacks such as credential stuffing, where the attacker has a list of valid usernames and passwords.
   - Permits brute force or other automated attacks.
   - Permits default, weak, or well-known passwords, such as "Password1" or "admin/admin".

8. Software and data integrity failures:
   Software and data integrity failures relate to code and infrastructure that does not protect against integrity violations. An example of this is where an application relies upon plugins, libraries, or modules from untrusted sources, repositories, and content delivery networks (CDNs). An insecure CI/CD pipeline can introduce the potential for unauthorized access, malicious code, or system compromise. Lastly, many applications now include auto-update functionality, where updates are downloaded without sufficient integrity verification and applied to the previously trusted application. Attackers could potentially upload their own updates to be distributed and run on all installations. Another example is where objects or data are encoded or serialized into a structure that an attacker can see and modify is vulnerable to insecure deserialization.

9. Server-side request forgery:
   SSRF flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL. It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or another type of network access control list (ACL).
10. Security logging and monitoring failures:
    This is to help detect, escalate, and respond to active breaches. Without logging and monitoring, breaches cannot be detected. Insufficient logging, detection, monitoring, and active response occurs any time.

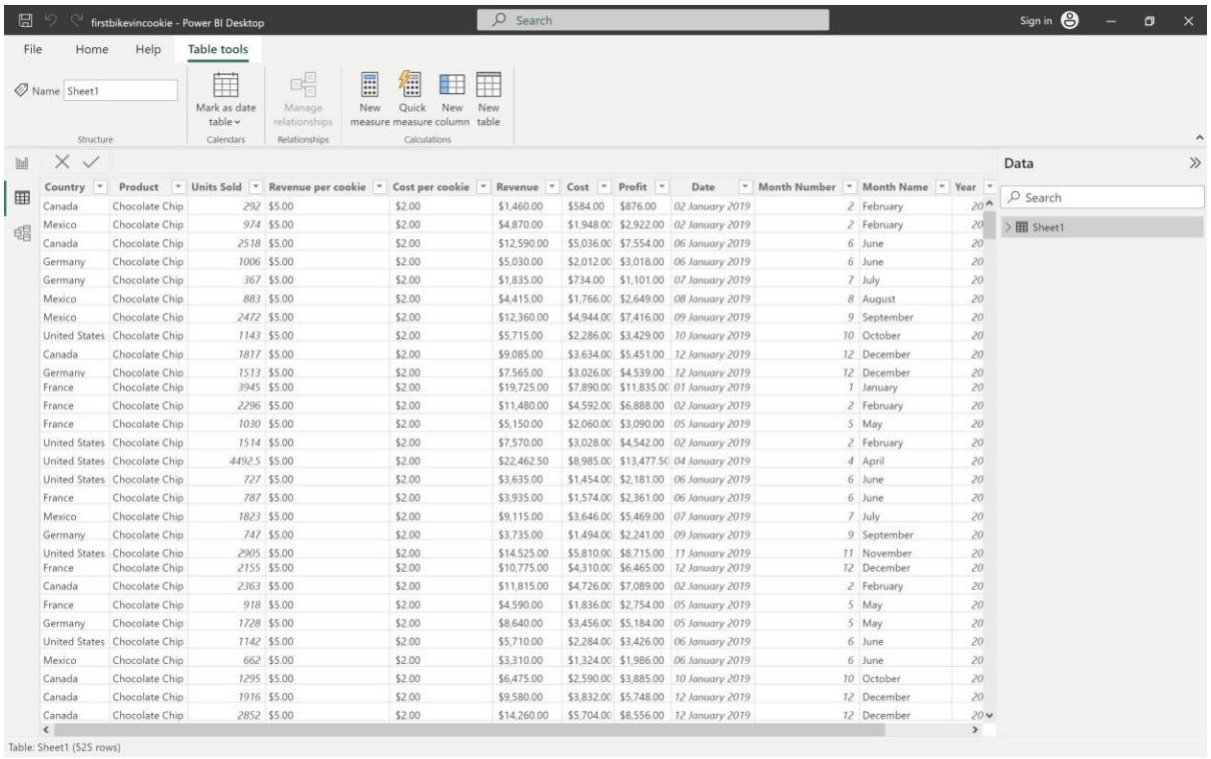
Practical 4: Network scanner

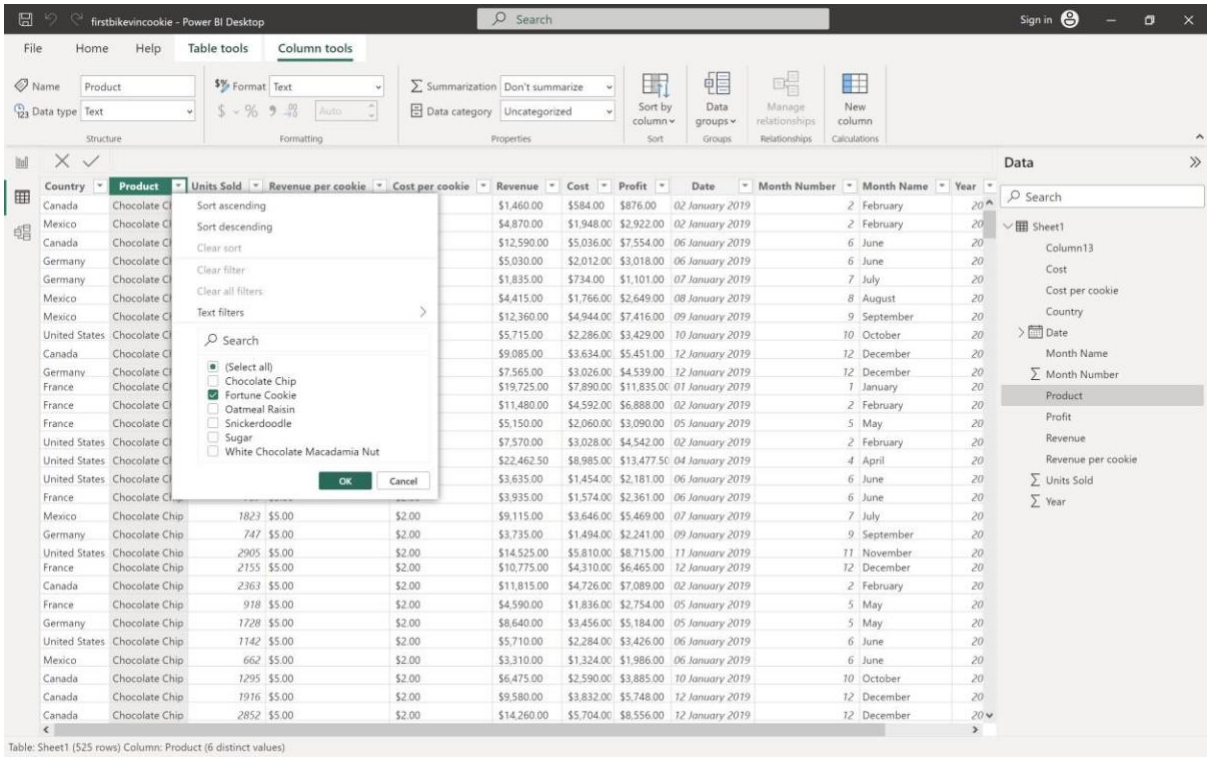
Nmap: Nmap is a network scanner created by Gordon Lyon. Nmap is used to discover hosts and services on a computer network by sending packets and analyzing the responses. Nmap provides a number of features for probing computer networks, including host discovery and service and operating system detection. It is used for network exploration, host discovery, and security auditin

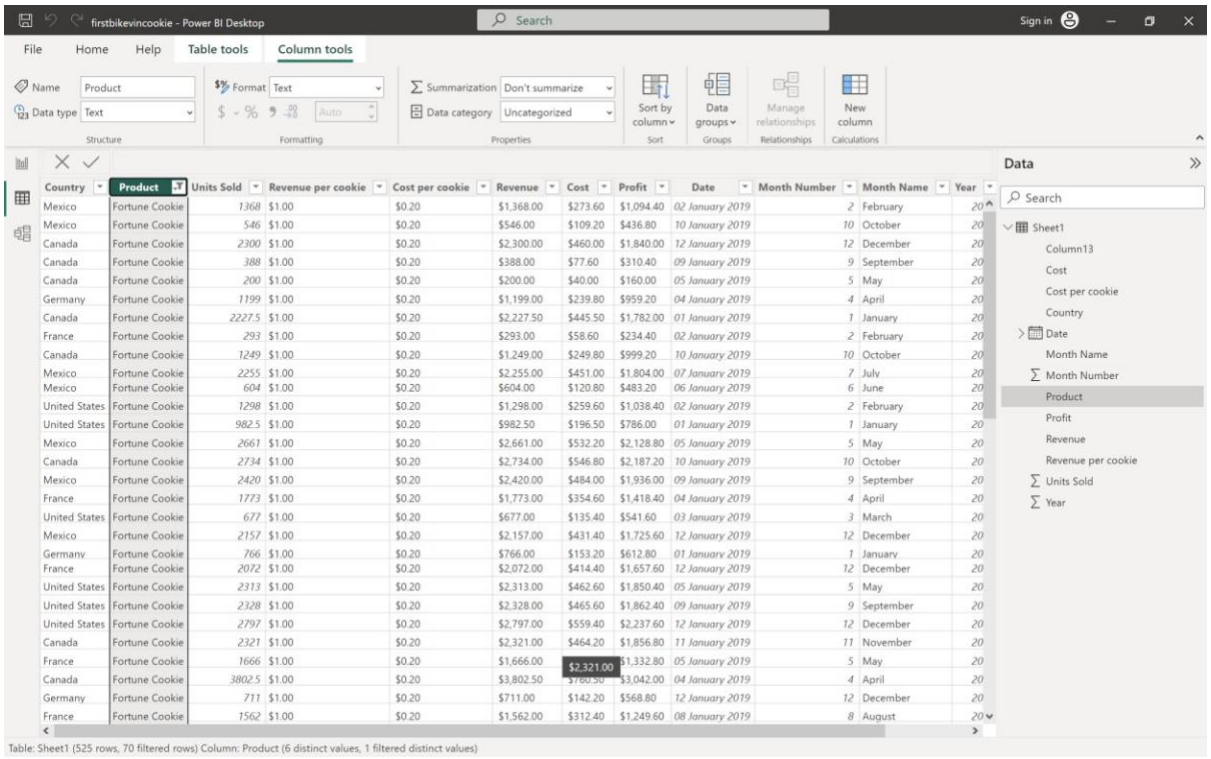


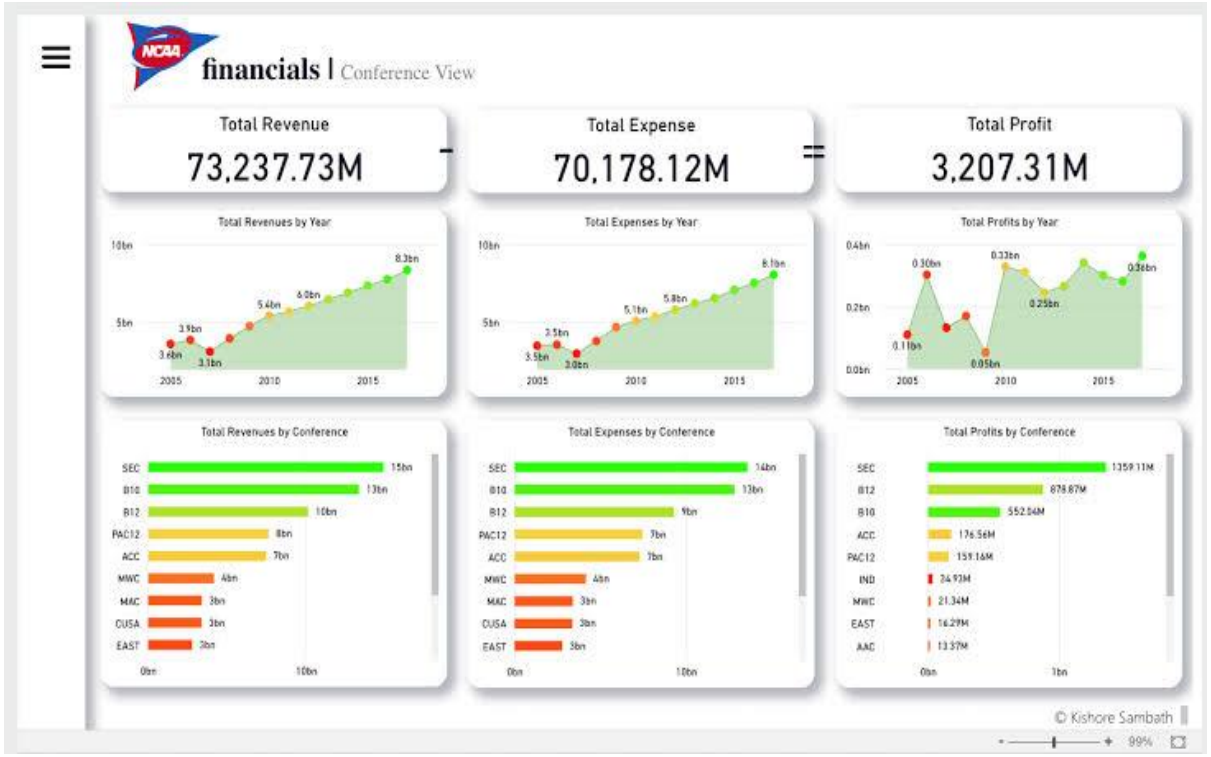
Practical 5: Power BI

Microsoft Power BI is an interactive data visualization software product developed by Microsoft with a primary focus on business intelligence. It is part of the Microsoft Power Platform.







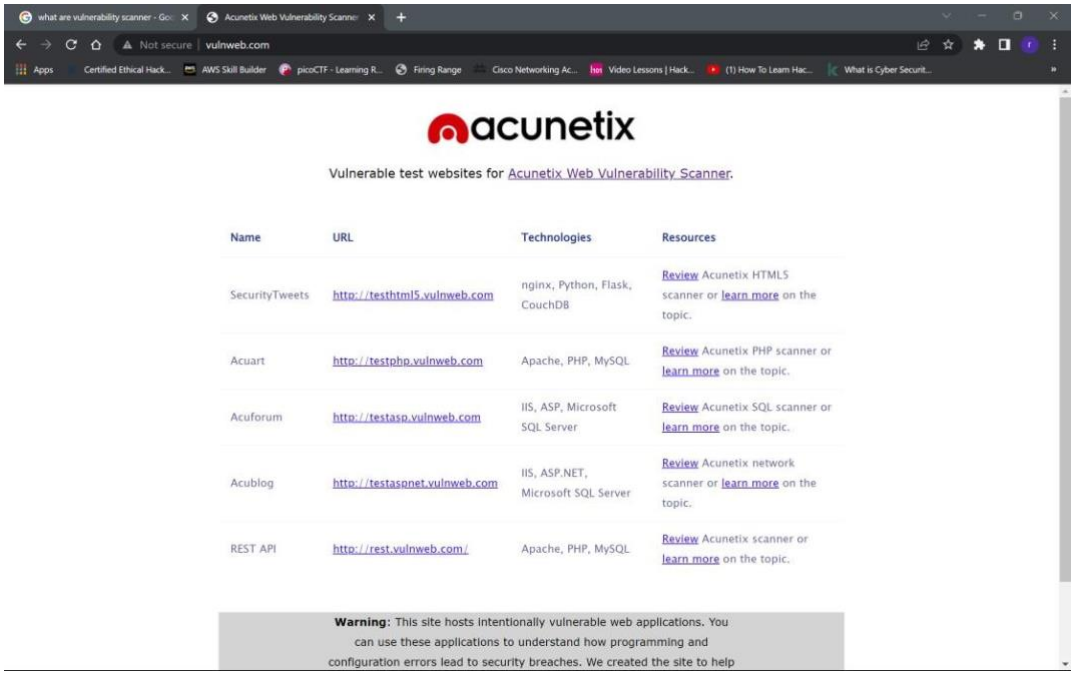


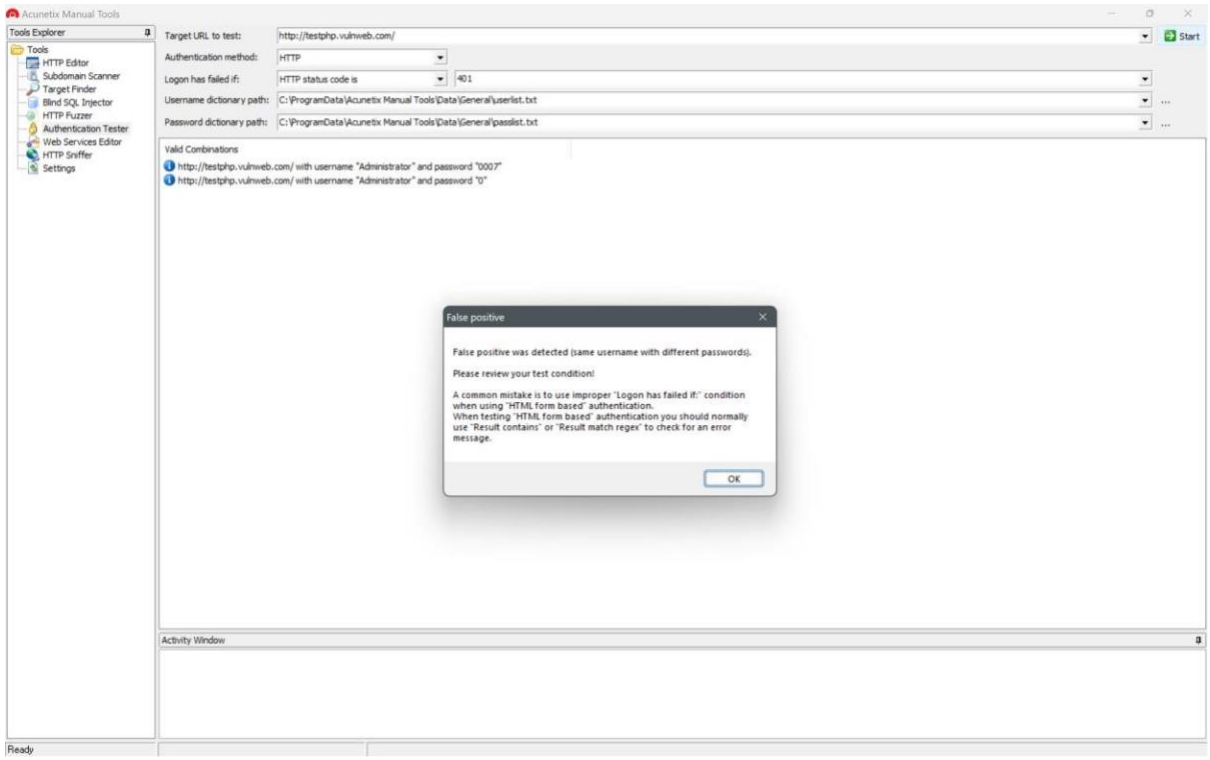
Practical 6: Vulnerability scanner

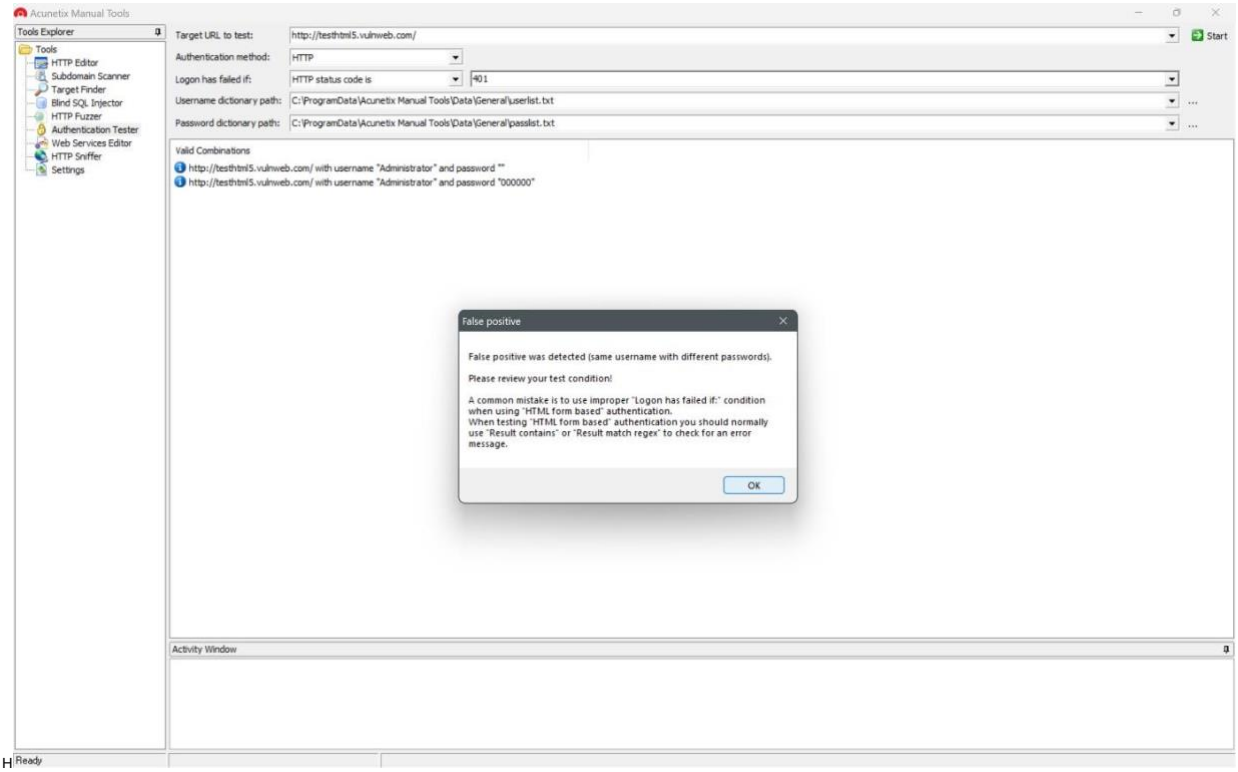
A vulnerability scanner is a computer program designed to assess computers, networks or applications for known weaknesses. These scanners are used to discover the weaknesses of a given system.

Acunetix is an automated web application security testing tool that audits your web applications by checking for vulnerabilities like SQL Injection, Cross site scripting

and other exploitable vulnerabilities.




Acunetix Manual Tools
Tools Explorer
Tools
HTTP Editor
Subdomain Scanner
Target Finder
Blind SQL Injector
HTTP Fuzzer
Authentication Tester
Web Services Editor
HTTP Sniffer
Settings
Target URL to test:
http://testphp.vulnweb.com/
Start
Authentication method:
HTTP
Logon has failed if:
HTTP status code is
401
Username dictionary path:
C:\ProgramData\Acunetix Manual Tools\Data\General\userlist.txt
Password dictionary path:
C:\ProgramData\Acunetix Manual Tools\Data\General\passlist.txt
Valid Combinations
http://testphp.vulnweb.com/ with username "Administrator" and password "0007"
http://testphp.vulnweb.com/ with username "Administrator" and password "0"
False positive
False positive was detected (same username with different passwords).
Please review your test condition!
A common mistake is to use improper "Logon has failed if:" condition when using "HTML form based" authentication.
When testing "HTML form based" authentication you should normally use "Result contains" or "Result match regex" to check for an error message.
OK
Activity Window
Ready


Acunetix Manual Tools
Tools Explorer
Tools
HTTP Editor
Subdomain Scanner
Target Finder
Blind SQL Injector
HTTP Fuzzer
Authentication Tester
Web Services Editor
HTTP Sniffer
Settings
Target URL to test:
http://testhtml5.vulnweb.com/
Start
Authentication method:
HTTP
Logon has failed if:
HTTP status code is
401
Username dictionary path:
C:\ProgramData\Acunetix Manual Tools\Data\General\userlist.txt
Password dictionary path:
C:\ProgramData\Acunetix Manual Tools\Data\General\passlist.txt
Valid Combinations
http://testhtml5.vulnweb.com/ with username "Administrator" and password ""
http://testhtml5.vulnweb.com/ with username "Administrator" and password "000000"
False positive
False positive was detected (same username with different passwords).
Please review your test condition!
A common mistake is to use improper "Logon has failed if:" condition when using "HTML form based" authentication.
When testing "HTML form based" authentication you should normally use "Result contains" or "Result match regex" to check for an error message.
OK
Activity Window
Ready

Practical 7: browser inspect get password

Convert the password from “ **** “ format to “text”
Go in inspect
Find the div of input type =password
Change it to ‘text’