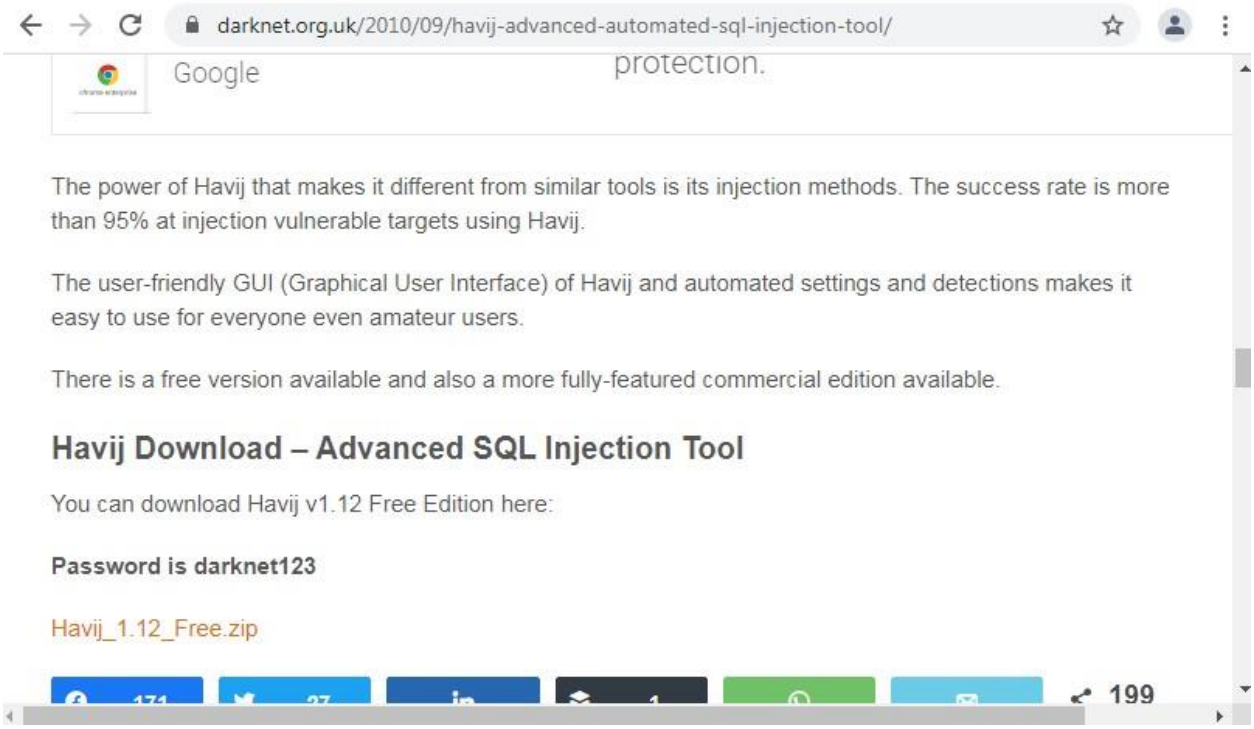
Task#5

Perform SQL Injection using Windows 7 as a host.

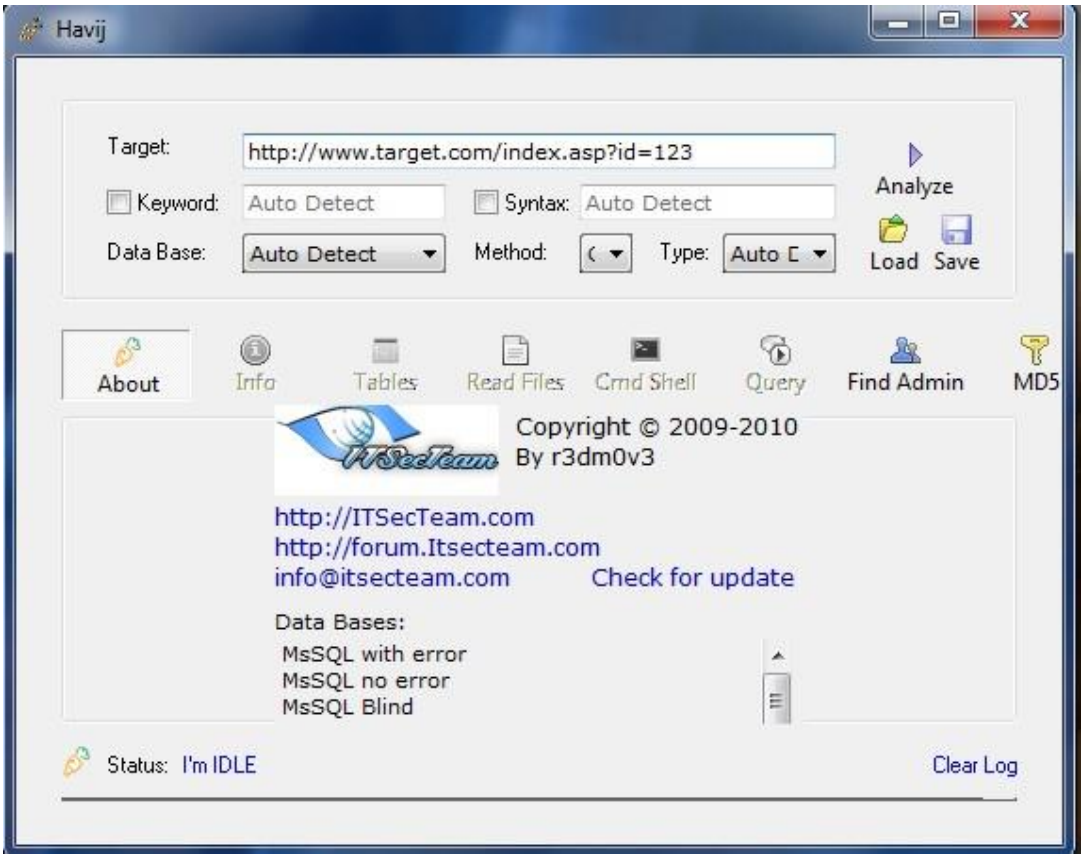
Download Havij from internet.



Run and install Havij.exe file.

Task#5

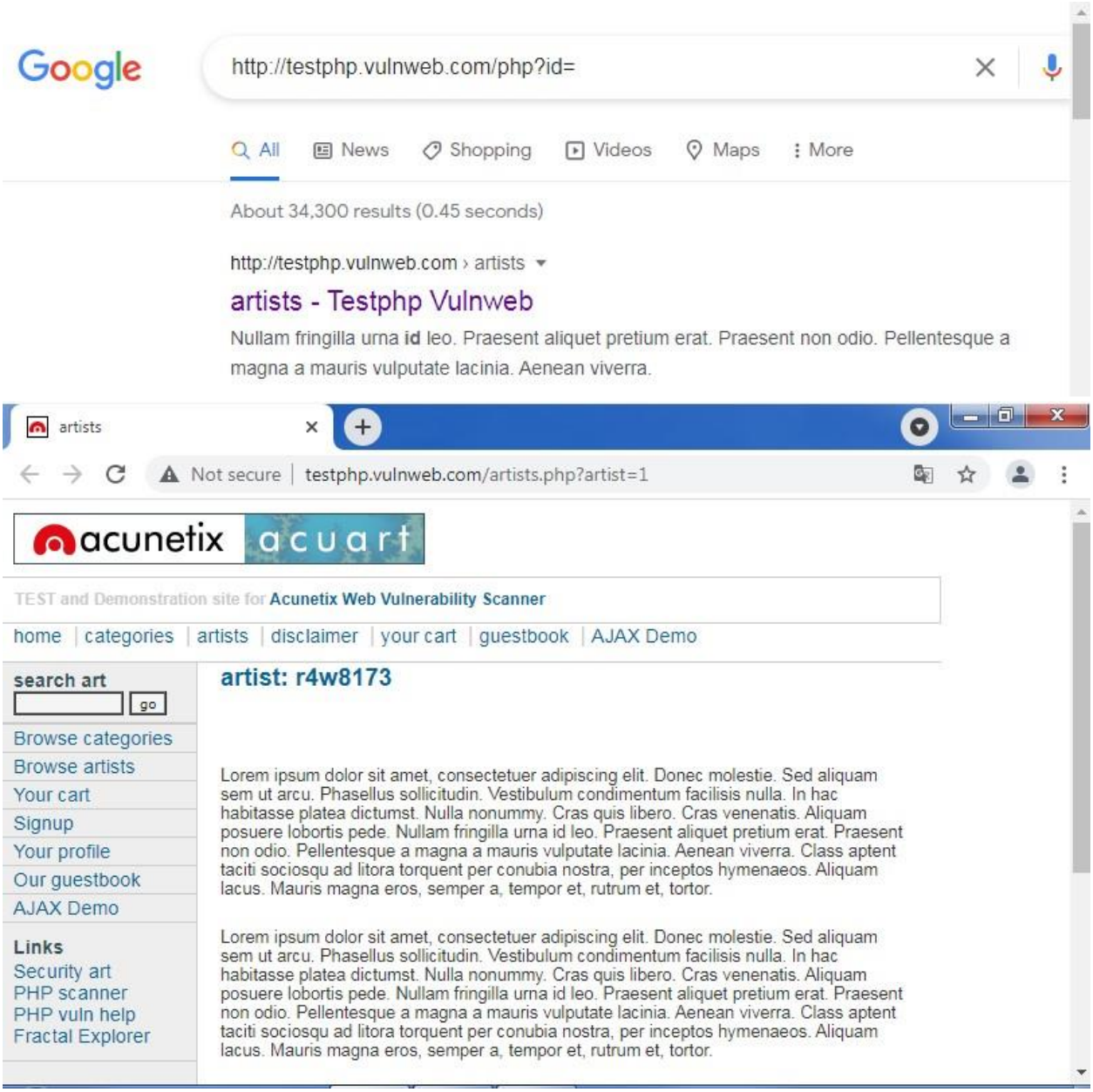
Open installed Havij



Get the target url in mentioned format “http://www.target.com/index.asp?id=123”.

Task#5

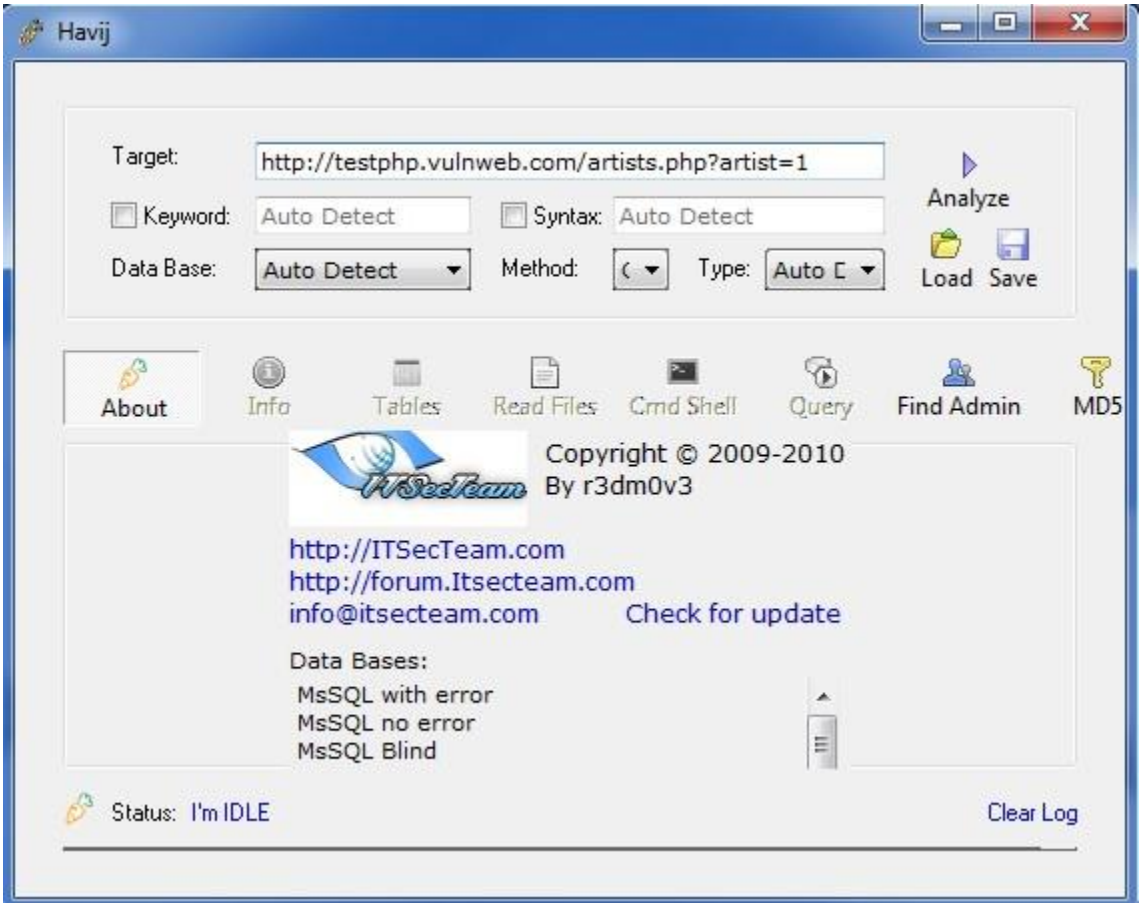
Search for the target url on search engine.



We got the target url “testphp.vulnweb.com/artists.php?artist=1”.

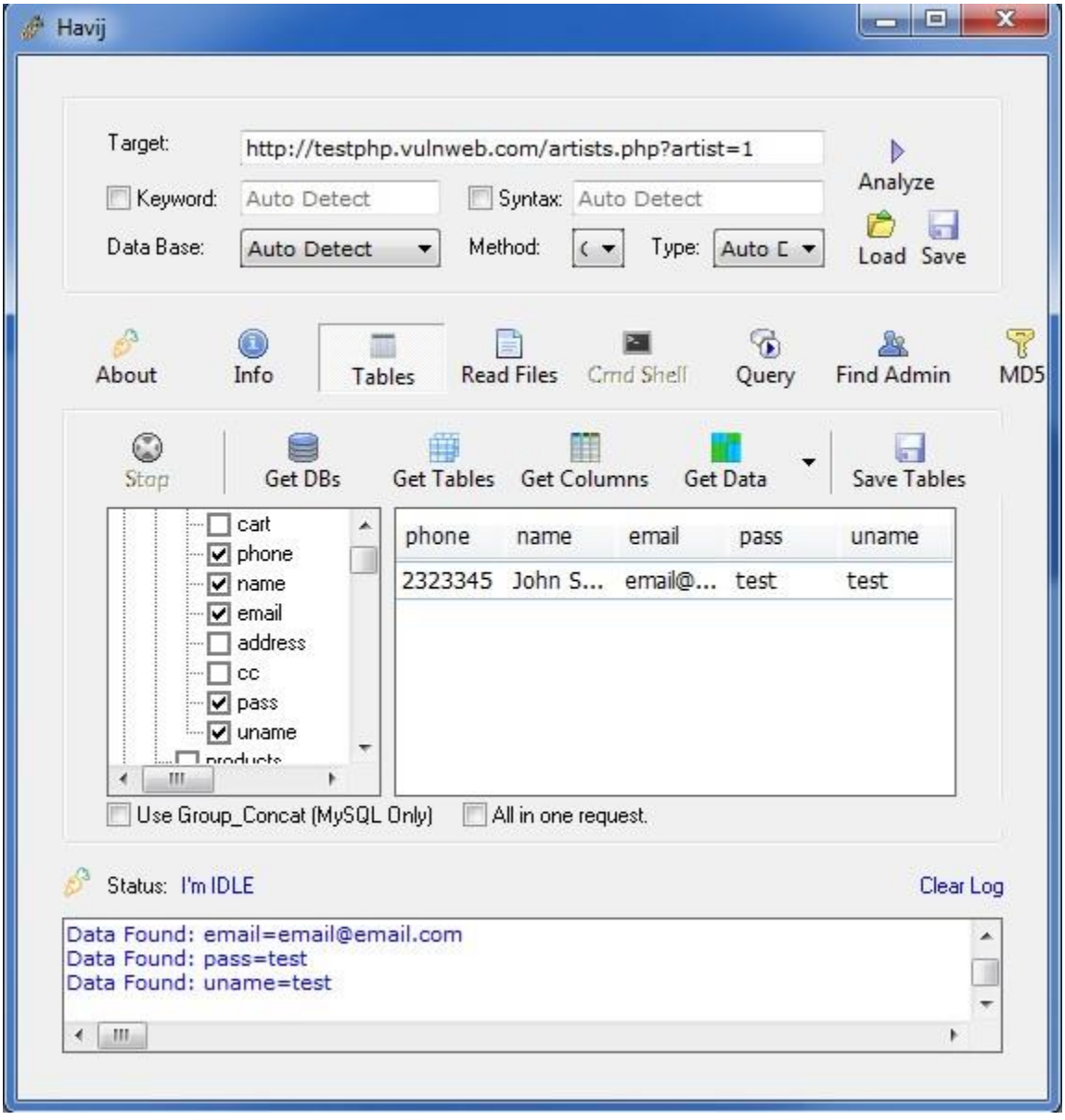
Task#5

Open Havij and paste the target url and click on Analyze.



Task#5

1. Click on Tables
2. Click on Get DBs
3. Select from left options and then click on Get Tables
4. Finally click on Get Columns



We got all the details available in target website.

Task#5

Steps to prevent SQL injection attacks

Though SQL injection attacks are still the most dangerous threat to web administrators, the good news is that there are plenty website owners can do to mitigate the danger.

1. Validate User Inputs

A common first step to preventing SQL injection attacks is validating user inputs. First, identify the essential SQL statements and establish a whitelist for all valid SQL statements, leaving unvalidated statements out of the query. This process is known as input validation or query redesign.

Additionally, you should configure inputs for user data by context. For example, input fields for email addresses can be filtered to allow only the characters in an email address, such as a required "@" character. Similarly, phone numbers and social security numbers should only be filtered to allow the specific number of digits for each.

While this action alone won't stop SQLi attackers, it is an added barrier to a common fact-finding tactic for SQL injection attacks.

2. Use Stored Procedures In The Database

Similar to parameterization, using stored procedures also requires variable binding. Unlike the prepared statements approach to mitigating SQLi, stored procedures reside in the database and are called from the web application. Stored procedures are also not immune to vulnerabilities if dynamic SQL generation is used.

Organizations like OWASP say only one of the parameterized approaches is necessary, but neither method is enough for optimal security. Crafting parameterized queries should be done in conjunction with our other recommendations.

3. Actively Manage Patches And Updates

Vulnerabilities in applications and databases that are exploitable using SQL injection are regularly discovered and publicly identified. Like so many cybersecurity threats, it's vital organizations stay in tune with the most recent news and apply patches and updates as soon as practical. For SQLi purposes, this means keeping all web application software components, including database server software, frameworks, libraries, plug-ins, and web server software, up to date.

4. Raise Virtual Or Physical Firewalls

We strongly recommend using a software or appliance-based web application firewall (WAF) to help filter out malicious data.

Firewalls today, including NGFW and FWaaS offerings, have both a comprehensive set of default rules and the ease to change configurations as needed. If a patch or update has yet to be released, WAFs can be handy.

A popular example is the free, open-source module ModSecurity, available for Apache, Microsoft IIS, and nginx web servers. ModSecurity provides a sophisticated and ever-evolving set of rules to filter

potentially dangerous web requests. Its SQL injection defenses can catch most attempts to sneak SQL through web channels.

5. Establish Appropriate Privileges And Strict Access

Given the power SQL database holds for an organization, it's imperative to enforce least privilege access policies with strict rules. If a website only requires the use of SELECT statements for a database, there's no reason it should have additional INSERT, UPDATE, or DELETE privileges.

Further, your database should only be accessed with admin-level privileges when necessary, nevermind granting others access. Using a limited access account is far safer for general activity and ultimately limits an attacker's access if the less-privileged credential is compromised.

6. Deny Extended URLs

Another tactic by SQLi attackers is sending excessively long URLs causing the server to fail at logging the complete request. In 2013, *eSecurityPlanet* reported on how attackers exploited Foxit by sending users long URLs that would trigger a stack-based buffer overflow.

Microsoft IIS, as another example, is built to process requests over 4096 bytes long. However, the web server software fails to place the contents of the request in the log files. Attackers can then go undetected while performing queries. To avoid this, set a limit of 2048 bytes for URLs.

7. Continuous Monitoring Of SQL Statements

Organizations or third-party vendors should continually monitor all SQL statements of database-connected applications for an application, including documenting all database accounts, prepared statements, and stored procedures. With visibility into how SQL statements function, it's much easier to identify rogue SQL statements and vulnerabilities. In this continued review, admins can delete and disable unnecessary accounts, prepared statements, and stored procedures.

Monitoring tools that utilize machine learning and behavioral analysis like PAM and SIEM can be excellent add-ons to your network security.

8. Perform Regular Auditing And Penetration Testing

Regular audits of your database and application security are becoming increasingly necessary, including auditing logs for suspicious activity, group and role memberships privileges, and variable binding terms.

Just as crucial as auditing for malicious behavior is conducting penetration tests to see how your defenses respond to an array of potential attacks, including SQLi. Most penetrating testing companies can find threats such as cross-site scripting, retired software, unpatched vulnerabilities, injections, and insecure passwords.