**PROBLEM STASTEMENT:**

5. Perform SQL injection Manually on http://testphp.vulnweb.com Write a report along with screenshots and mention preventive steps to avoid SQL injections .

SOLUTION:

WHAT IS SQL INJECTION:

SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve. This might include data belonging to other users, or any other data that the application itself is able to access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior.

In some situations, an attacker can escalate an SQL injection attack to compromise the underlying server or other back-end infrastructure, or perform a denial-of-service attack.

WHAT IS HAVIJ TOOL:

”Havij is an automated SQL Injection tool that helps **penetration testers** to find and exploit SQL Injection vulnerabilities on a web page. It can take advantage of a vulnerable web application. By using this software, user can perform back-end database fingerprinting, retrieve DBMS login names and password hashes, dump tables and columns, fetch data from the database, execute SQL statements against the server, and even access the underlying file system and execute operating system shell commands. ”

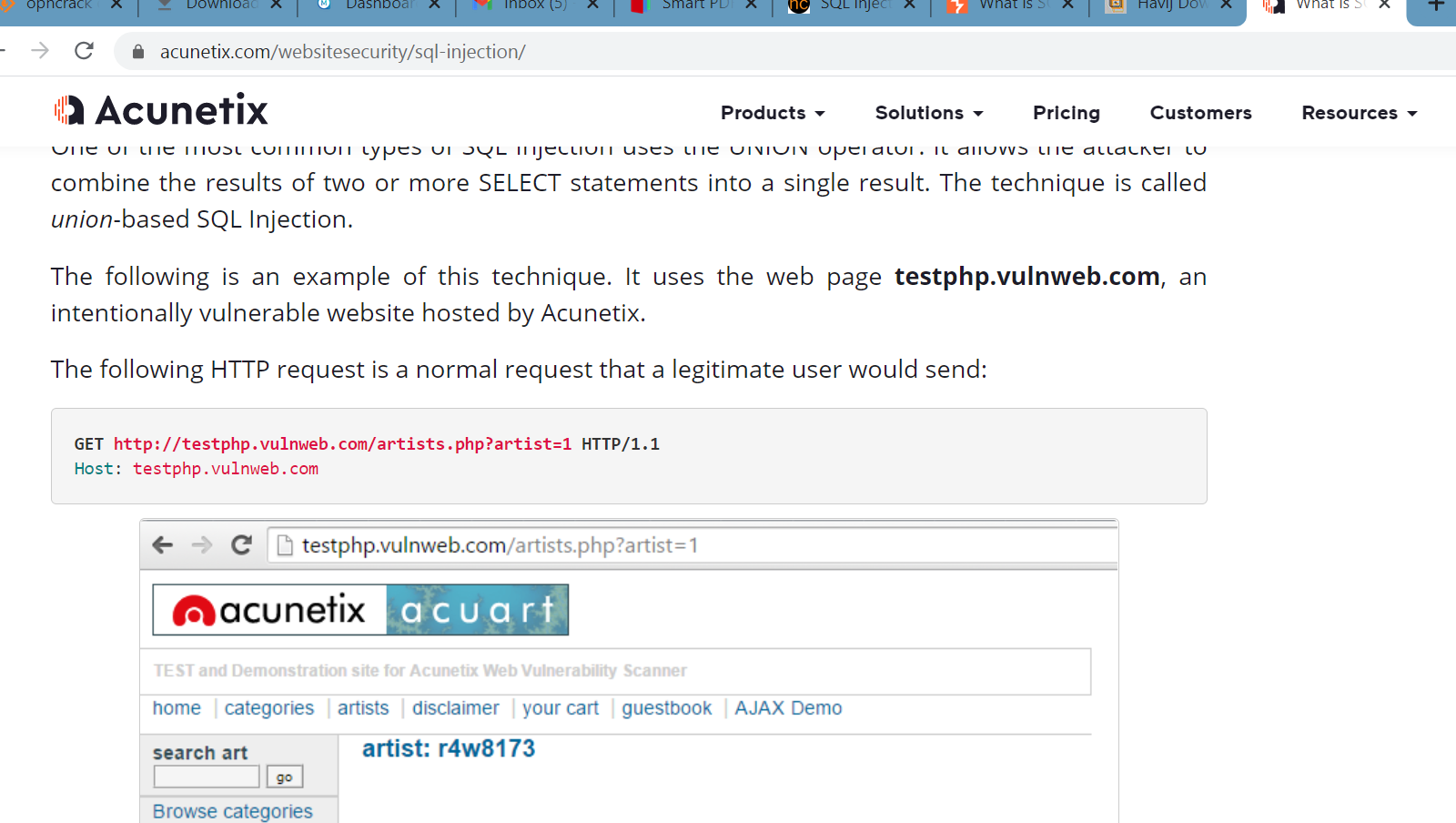
It is available both in free and commercial versions. Today we are going to see how to dump the contents of a database using Havij. For this I am going to use the free version. First download Havij from[here](http://www.itsecteam.com/) and install it. Then open it and enter the vulnerable page url in the target column ( for this tut I am using my own vulnerable webpage ).

STEPS :

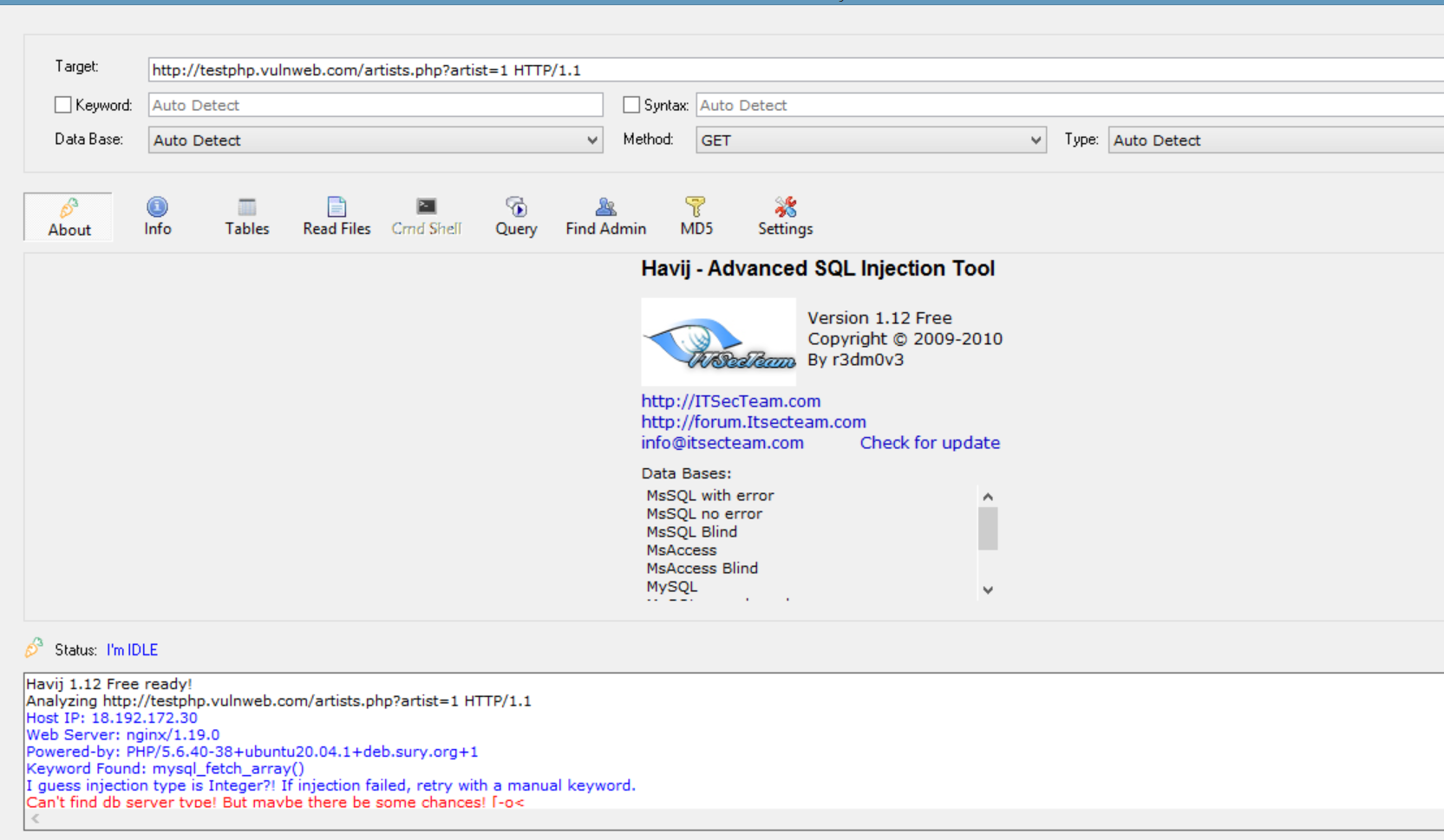
1. First we download the Havij Tool on internet.And install on your system.After installation it will show like this:

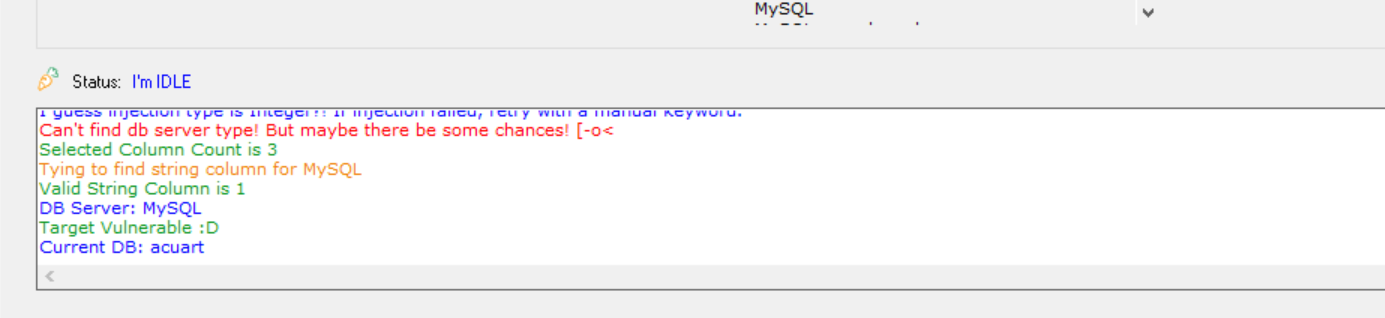


1. Given website loop hole to be searched.And indentify the SQL injection link.



3.copy the sql injction value,and paste it on the havij tool.it shows the host ip of the website.





4.Read and modify the data.

preventive steps to avoid SQL injections :

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](https://www.esecurityplanet.com/trends/ibm-appscan-takes-aim-at-input-validation/) for all valid SQL statements, leaving unvalidated statements out of the query. This process is known as [input validation](https://www.esecurityplanet.com/trends/ibm-appscan-takes-aim-at-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.

### Sanitize Data By Limiting Special Characters:

Another component of safeguarding against SQL injection attacks is mitigating inadequate [data sanitization](https://www.esecurityplanet.com/endpoint/prevent-web-attacks-using-input-sanitization/). Because SQLi attackers can use unique character sequences to take advantage of a database, sanitizing data not to allow string concatenation is critical.

One way of doing this is configuring user inputs to a function such as MySQL’s mysql\_real\_escape\_string(). Doing this can ensure that any dangerous characters such as a single quote **‘**is not passed to a SQL query as instructions. A primary method of avoiding these unauthenticated queries is the use of prepared statements.

### 3. Enforce Prepared Statements And Parameterization

Sadly, input validation and data sanitization aren’t fix-alls. It’s critical organizations also use prepared statements with parameterized queries, also known as variable binding, for writing all database queries. By defining all SQL code involved with queries, or parameterization, you can distinguish between user input and code.

While dynamic SQL as a coding technique can offer more flexible application development, it can also mean SQLi vulnerabilities as accepted code instructions. By sticking with standard SQL, the database will treat malicious SQL statements inputted like data and not as a potential command.

### 4. 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](https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html) 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.

### 5. 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](https://www.esecurityplanet.com/threats/microsoft-patches-massive-remote-access-vulnerability/) 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.

### 6. Raise Virtual Or Physical Firewalls

We strongly recommend using a software or appliance-based [web application firewall (WAF)](https://www.esecurityplanet.com/network-security/application-firewalls.html) to help filter out malicious data.

Firewalls today, including [NGFW](https://www.esecurityplanet.com/products/top-ngfw/) and [FWaaS](https://www.esecurityplanet.com/cloud/firewalls-as-a-service-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](https://modsecurity.org/), 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.

### 7. Harden Your OS And Applications

This step goes beyond mitigating SQL injection attacks in ensuring your entire physical and virtual framework is working intentionally. With the big news of [supply chain compromises in 2020](https://www.esecurityplanet.com/threats/guarding-against-solorigate-ttps-solarwinds-hack/), many are looking to NIST and other industry-standard security checklists to harden operating systems and applications.

Adopting application vendor security guidelines can enhance an organization’s defensive posture and help identify and disable unnecessary applications and servers.

### 8. Reduce Your Attack Surface

In cybersecurity, an attack surface refers to the array of potential entry points for attackers. So in the context of SQLi attacks, this means disposing of any [database](https://www.esecurityplanet.com/products/database-security-tools/) functionalities that you don’t need or further safeguarding them.

One such example is the [xp\_cmdshell](https://docs.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/xp-cmdshell-transact-sql?view=sql-server-ver15) extended stored procedure in the Microsoft SQL Server. This procedure can spawn a Windows command shell and pass a string for execution. Because the Windows process generated by xp\_cmdshell has the same security privileges as the SQL Server service account, the attacker can cause severe damage.

### 9. Establish Appropriate Privileges And Strict Access

Given the power SQL database holds for an organization, it’s imperative to enforce [least privilege access policies](https://www.esecurityplanet.com/applications/privileged-access-management-pam/) 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.

### 10. Limit Read-Access

Connected to the principle of least privilege for SQL injection protection is configuring [read-access to the database](https://www.esecurityplanet.com/applications/tips-for-privileged-access-management-pam/). If your organization only requires active users employing read-access, it’s undoubtedly easier to adopt. Nevertheless, this added step is imperative for stopping attackers from altering stored information.

### 11. Encryption: Keep Your Secrets Secret

It’s best to assume internet-connected applications are not secure. Therefore [encryption](https://www.esecurityplanet.com/threats/tokenization-vs-encryption/) and hashing passwords, confidential data, and connection strings are of the utmost importance.

Encryption is almost universally employed as a data protection technique today and for a good reason. Without appropriate encryption and hashing policies, sensitive information could be in plain sight for an intruder. While only a part of the security checklist, [Microsoft notes](https://docs.microsoft.com/en-us/archive/msdn-magazine/2003/november/safeguarding-database-connection-strings-and-other-settings) encryption, “transforms the problem of protecting data into a problem of protecting cryptographic keys.”