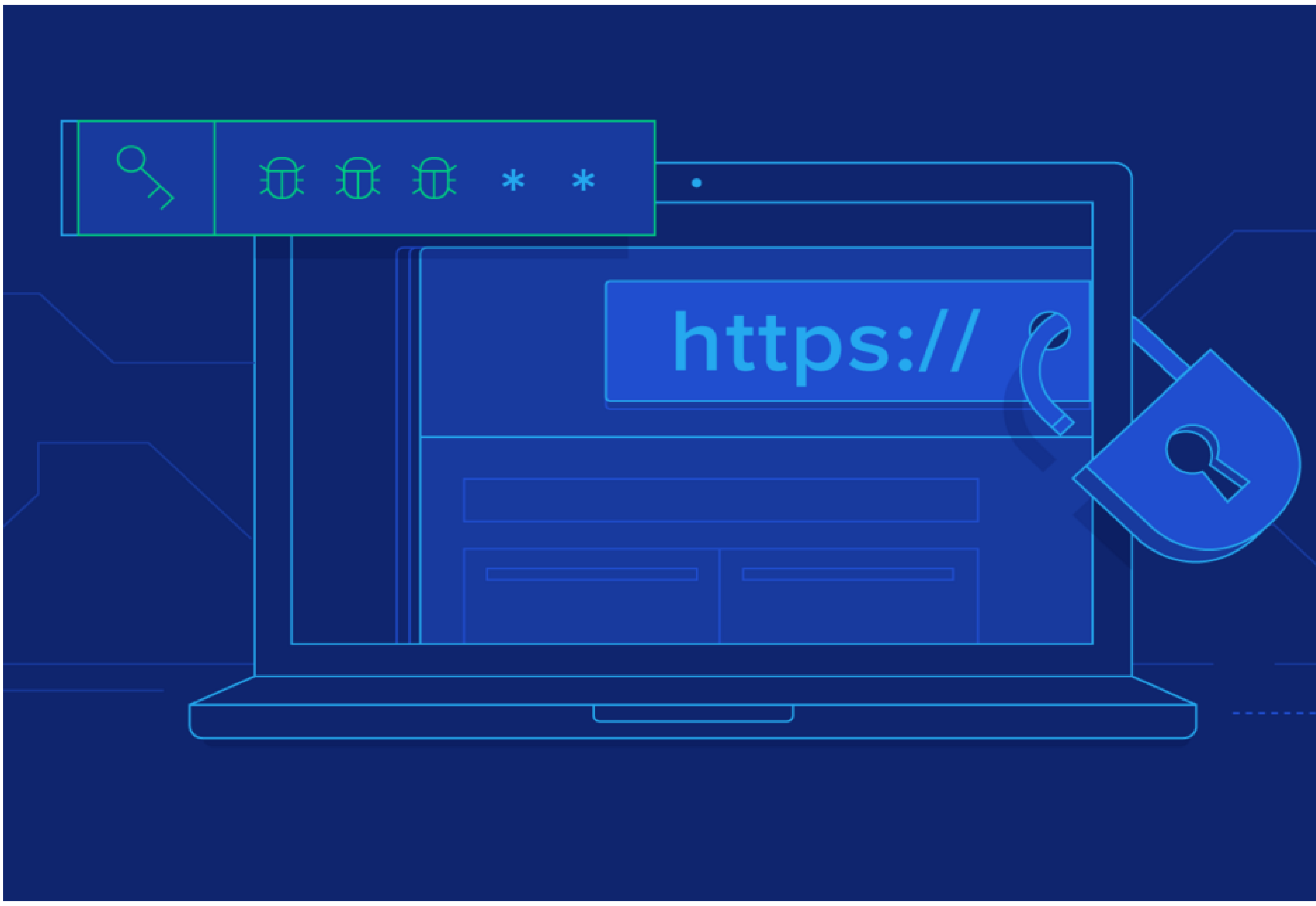
Web vulnerability

By- Legion of geeks



Index

#A word from creators

We are Legion of geeks. A learning circle basically, But more than that we are intending to build a community of tech enthusiast. Our moto is to spread mulearn and coding culture throughout our campus which is College of engineering,Perumon.Our team comprises of four members currently Ayush. S(lead),Gandharv C,Hari nattinpuram,Arsha Maria joji

Hari was the one who took all the initiative throughout the project as he had better understanding about the vulnerabilities and given instruction to fulfil the task

Gandharv is the one took in charge to detect cross site scripting and presented the report comprising of all possible details

Arsha as the Female code ninja of the group she was assigned of the task to detect CSRF vulnerability and submit the report

Ayush was assigned of SQL injection vulnerability and present a report comprising the code and its working

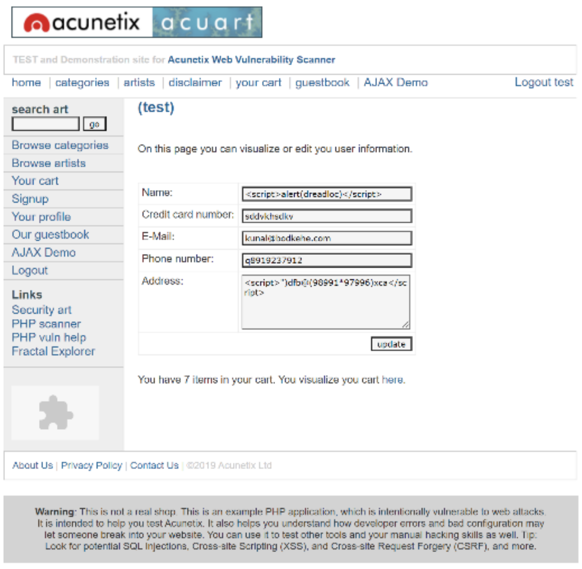
Introduction

Vulnerability refers to a weakness or flaw in a system's design, implementation, or operation that could be exploited to compromise the system's security. These weaknesses can exist in software, hardware, or even human factors, making it crucial to identify and address them to prevent potential threats and attacks. Certainly. Vulnerabilities can manifest in various forms, such as coding errors, misconfigurations, or outdated software. They create potential entry points for attackers to exploit, leading to unauthorized access, data breaches, or service disruptions. The process of identifying and fixing vulnerabilities is crucial for maintaining the integrity, confidentiality, and availability of systems, ultimately safeguarding against cyber threats. Regular security assessments, patches, and robust security practices are essential components of managing and mitigating vulnerabilities in any technological environment.

SQL INJECTION

SQL injection is a type of cyber attack that occurs when an attacker inserts or manipulates malicious SQL (Structured Query Language) code into input fields of a web application.In a SQL injection attack, the attacker exploits vulnerabilities in the application's input validation procedures to inject malicious SQL code into the queries that the application sends to its database. This can allow the attacker to execute arbitrary SQL commands, potentially gaining unauthorized access to the database, retrieving, modifying, or deleting data, and even taking control of the entire application or server.

Here in the given webpage vulnerability via SQL injection can generated by a malicious code 'OR 1=1--(is space)


acunetix acuart
TEST and Demonstration site for Acunetix Web Vulnerability Scanner
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
Logout test
search art
go
Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Logout
Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer
update
About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd

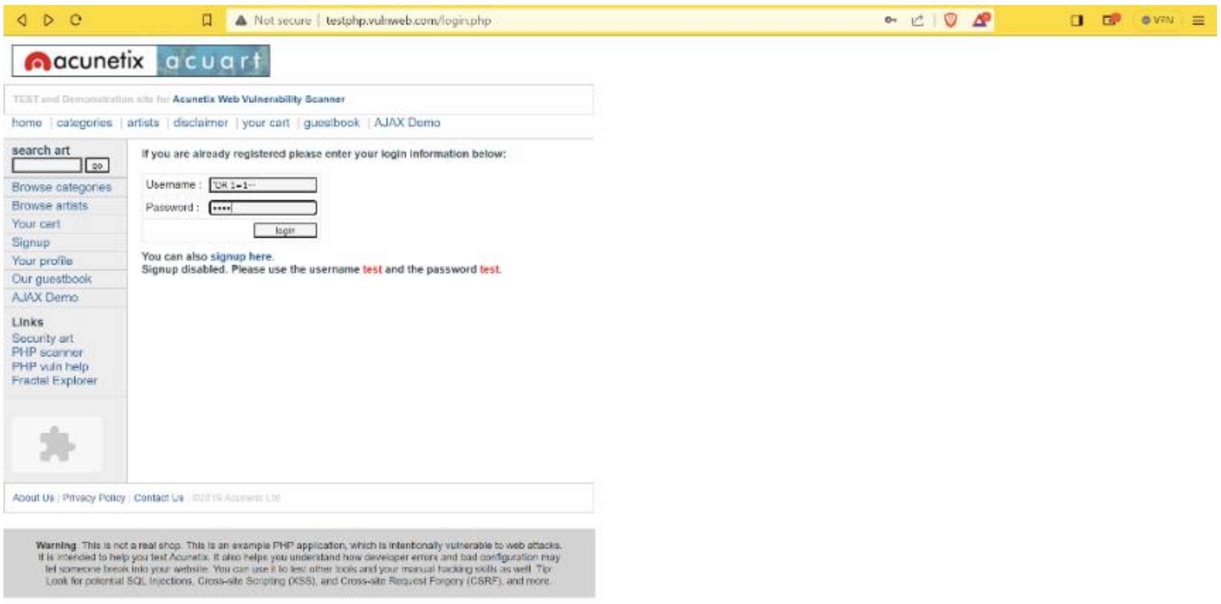
(test)

On this page you can visualize or edit you user information.

Name:	<script>alert(dreadloc)</script>
Credit card number:	sddvkhsdkv
E-Mail:	kunal@hodkehe.com
Phone number:	q8919237912
Address:	<script>")dfb@(98991*97996)xca</script>

You have 7 items in your cart. You visualize you cart here.

Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.


Not secure | testphp.vulnweb.com/login.php
acunetix acuart
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
search art
go
Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer
login
About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

If you are already registered please enter your login information below:

Username : 'OR 1=1--

Password : ••••

You can also signup here.
Signup disabled. Please use the username test and the password test.

Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.

This malicious code is used where login is made

To prevent SQL injection attacks, developers should use parameterized queries or prepared statements, which separate SQL code from user input, making it much more difficult for attackers to inject malicious code. Additionally, input validation and proper encoding techniques should be employed to ensure that user input is treated as data, not executable code. Regular security audits and testing are also important to identify and fix potential vulnerabilities in web applications.

CROSS SITE SCRIPTING(XSS)

Cross-Site Scripting (XSS) is a prevalent web application security vulnerability that occurs when an application includes untrusted data in a web page without proper validation or escaping. This allows attackers to inject malicious scripts into web pages that are then viewed by other users. XSS can have various forms:

1. *Stored XSS:* Malicious scripts are permanently stored on the target server, affecting every user who views the compromised page.

2. *Reflected XSS:* The injected script is embedded in a URL, and the server reflects it back to the user's browser. This type often relies on social engineering to trick users into clicking on a manipulated link.

3. *DOM-based XSS:* The attack occurs on the client side, manipulating the Document Object Model (DOM) of a web page. The injected script alters the structure of the page, leading to security issues.

The potential consequences of XSS attacks are severe, ranging from the theft of sensitive information to session hijacking and defacement of websites. Mitigating XSS involves input validation, output encoding, and employing security mechanisms like Content Security Policy (CSP) to restrict the execution of scripts.

Developers must sanitize user inputs, validate and escape output, and adopt secure coding practices to prevent XSS. Regular security audits and staying informed about emerging threats are vital for maintaining robust defenses against this widespread vulnerability.

CSRF(Cross-Site Request Forgery)

Cross-Site Request Forgery (CSRF) is a security vulnerability that exploits the trust a website has in a user's browser. In a CSRF attack, an attacker tricks a victim into unknowingly submitting a request on a web application where the victim is authenticated. This unauthorized action can lead to unintended consequences, such as changing account settings, making financial transactions, or modifying data.

Key aspects of CSRF:

1. *Exploitation Mechanism:* Attackers typically craft malicious links or use social engineering techniques to induce users to click on manipulated content, which triggers unintended actions on a target site where the victim is authenticated.

2. *Preventing CSRF:* Developers can implement countermeasures like anti-CSRF tokens. These tokens are unique, unpredictable values embedded in forms or requests, validating that the request is legitimate. Verifying the presence and correctness of these tokens helps mitigate CSRF attacks.

3. *Same-Origin Policy:* Browsers enforce the Same-Origin Policy, which restricts web pages from making requests to a different domain than the one that served the page. However, CSRF exploits the fact that browsers automatically include authentication cookies with requests to the target domain.

4. *Security Best Practices:* Developers should follow secure coding practices, validate and sanitize inputs, use session management best practices, and employ secure design principles to minimize the risk of CSRF vulnerabilities.

CSRF attacks pose a significant threat to web applications, and mitigating this risk requires a combination of secure coding practices, user education, and the implementation of effective security mechanisms within the application itself. Regular security audits and staying informed about evolving security standards are crucial for maintaining a robust defense against

CONCLUSION

Best defence is a great offense. First, organizations have to identify potential vulnerabilities and threats using the appropriate tools and processes like vulnerability scanners and threat detection technology. It’s also important to prioritize vulnerabilities and threats once they’ve been identified so that they are eliminated or mitigated in order of importance. When it comes to protecting against cyber attacks, the best defense is a great offense. First, organizations have to identify potential vulnerabilities and threats using the appropriate tools and processes like vulnerability scanners and threat detection technology. It’s also important to prioritize vulnerabilities and threats once they’ve been identified so that they are eliminated or mitigated in order of importance.

After finding the vulnerabilities and threats, some of the most common fixes are:

Using antivirus software and other endpoint protection measures

Regular operating system patch updates

Implementing Wi-Fi security that secures and hides Wi-Fi networks

Installing or updating a firewall that monitors network traffic

Implementing and enforcing secure access through least privileges and user controls CSRF.