



Application Vulnerability Assessment

Helping reveal the sorry state of application security



Author:

Mushtaq Ahmed, Senior Security Assurance Analyst in Emirates Group IT.



Abstract

In the current age every business or person is moving towards e-commerce and the success or failure of a business is dictated by its internet presence. Many of the developed applications, including e-commerce applications that deal with buying and selling of products and services online, lack the basic security controls. This makes them prone to attacks and exploits that could result in getting the products or services for FREE. There are a few such vulnerabilities which expose the applications to such a threat. Here, we choose to discuss two such common issues which deal with improper session management and URL manipulation. We also discuss how these security issues can be eliminated.



Application Vulnerability Assessment

Helping reveal the sorry state of application security

The arms race has been changing since the dawn of time, the chances of survival for any species has always been dictated by the knowledge that they hold about the enemy and what it is capable of doing. If the prey is big, the hunter wanted to become bigger and if the hunter became bigger then the prey had to discover new ways to overcome the threat. This has been happening not only between hunter and prey but also in every walk of life. That could be between cities, states, countries, armies, businesses, corporations or even for that matter in computing.

Now when we talk about computers you might be wondering how an innocent looking computer can become smart, intelligent and secure. The key word here is secure and when we are talking about computers and arms races we are really referring to the people who are controlling these powerful machines and how they can be used to attack other innocent users, sites, businesses, and governments.

Imagine the day you get up and find out that your bank account is cleaned out, or perhaps your personal information has been revealed over the internet or a headline appears in the paper or a in a blog stating that another massive breach reveals the sorry state of IT security of your company or you see the Top 5 security breaches of this year and your company's name is present in that list. These are the scenarios which make you shiver.

In this race between the hackers and the hacked, do you intend to be the one on the receiving end or are you going to stand up and give them a strong fight? As per a popular web site, the term vulnerability means "a weakness which allows an attacker to reduce a system's Information Assurance". Vulnerability is a mix of three elements: a system having a flaw, attacker having access to the flaw, and attacker's capability to exploit the flaw. To be vulnerable, an attacker must have at least one applicable tool or technique that can exploit the system weakness. In this frame, vulnerability is also known as the attack surface.

In this race between the hackers and the hacked, will you like to be the one on the receiving end or are you going to stand up and give them a strong fight.

In this paper, we are going to explore two such attack surfaces that could be introduced by not following secure coding practices or by using flawed architecture in developing an e-commerce site. Exploiting a weakness or compromise of an e-commerce site of a company can be harmful in many ways and a few key ones are the serious impact on the brand image of the organization, loss of revenue and loss of customers.

Paying without a credit card

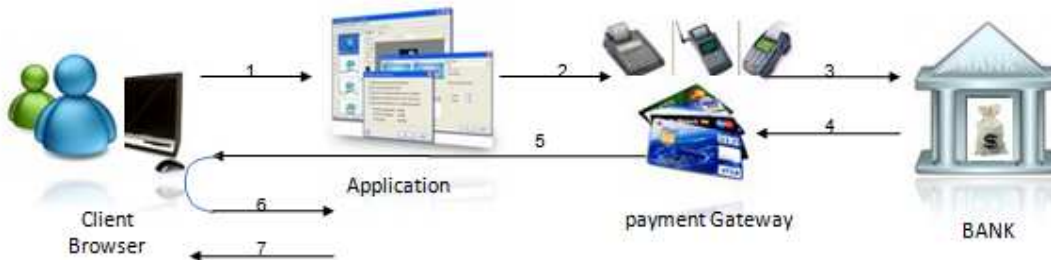
By response URL manipulation

How nice would it be to buy a product without having to pay for it! We can call it the buyer's paradise. There are many ways to develop an e-commerce application and integrate it with a payment gateway. If this is not done thoughtfully, this could introduce vulnerabilities that can be exploited to compromise the application, we have taken one such example and explored it below:

Let's talk about an instance where an e-commerce customer buys a product on the internet. While paying through his credit card he observes that the URL flips or changes for a moment before the payment confirmation page is displayed to him stating his credit card status is accepted or rejected.



If you notice carefully there is some action happening in between which tells the application that the credit card status is accepted or rejected. The actual flow of the transaction is illustrated in the following diagram:



The steps are:

1. User selects the product and pays through a credit card: The application server takes the users request for a product and presents the user with a payment page.
2. User enters the credit card number and submits this number to the payment page. The payment page takes the credit card number, expiry date and CVC number to the payment gateway.
3. The payment gateway will submit the credit card details to the bank for processing the payment.
4. The bank verifies the credit card details and sends the status as AUTHORIZED or REJECT to the payment gateway to progress the processing of the credit card payment.
5. The payment gateway sends the status to the application through the “User’s browser” as AUTHORIZED or REJECT.
6. At this point of time, a quick flip is witnessed in the user’s browser where the credit card status is returned back to the application through the user’s browser.
7. Application will check the status message and if it is AUTHORIZED then it will issue the product, otherwise it rejects the request and asks the user to try with a different card.

The deciding factor of this transaction is the AUTHORIZED or REJECT status which is returned from the bank and the payment page. If we look at steps 5 and 6 in the above diagram you will notice that the information about the payment status is sent through the browser and if a malicious user stops the transaction in case of a REJECT status and changes the REJECT status to AUTHORIZED then a fake credit card can be used to purchase a product for free.

You might wonder how we can change the URL status in case of a GET method and the data in the POST methods. This can be achieved by stopping the URL and changing the URL content in GET and in case of a POST methods using the browser utilities that can change the web content passing from one page to another.

If a malicious user stops the transaction in case of a REJECT status and changes the REJECT status to AUTHORIZED, then a fake credit card can be used to purchase a product for “FREE”



As you might have realized this compromise was possible only because the approval communication from the payment gateway was routed to the application through the client browser. This situation could have been avoided if the application would have been architected in such a way that the application server and the payment gateway shared the credit card status directly and not through the user's browser. Always make sure that the payment gateway and the application talks to each other and sends the status of the credit card status directly and only at that point the application should produce the payment successful message to the internet buyer. Further it is wiser to use the post methods then the get method, this ensures that the sensitive card data is not sent through the URL.

Payment Compromise

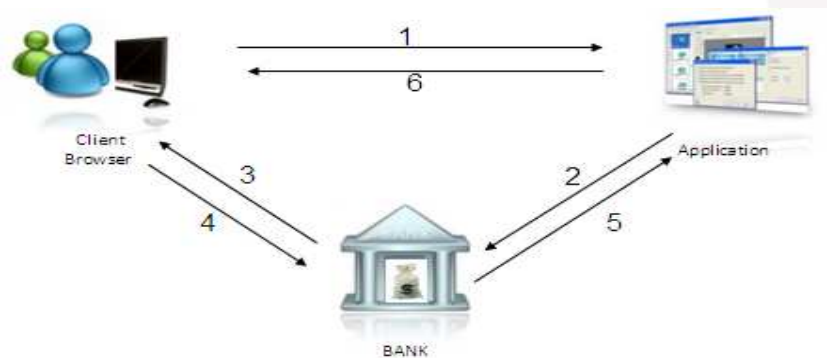
Information from browser history and session management

What is a session? A session permits an internet user from requiring to authenticate themselves repeatedly during a transaction. The example below is another scenario where common mistakes made in developing e-commerce applications can result in getting the services or products for free. This example exploits the session management vulnerability present on the web site, coupled with the saved history cached files and the above mentioned architectural issue of approval communication being routed through user's browser.

The example describes two different users buying two different products from the same e-commerce web site using the same computer. One of the users pays with his credit card and the other exploits the application and gets the desired product without paying for the transaction.

The 2 transaction are shown below:

First transaction:

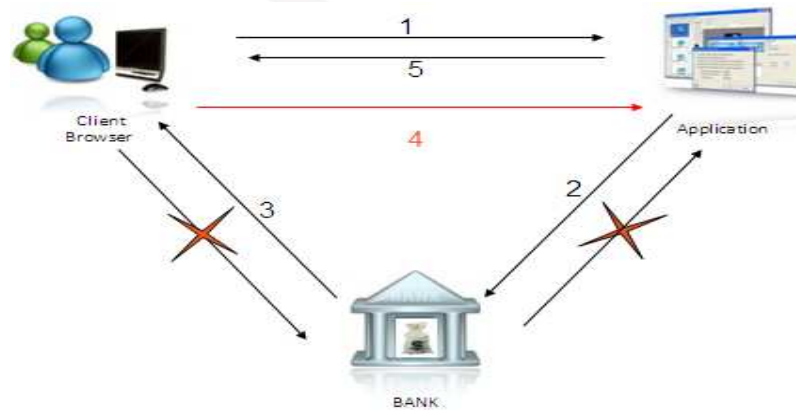


1. User "X" selects a product on the e-commerce web application and sends the request to buy the product.
2. The server sends this information to the payment gateway (bank) to send a payment page to the user.
3. The payment page is sent and displayed to the user by bank.
4. The user enters the credit card information and submits to the payment gateway (bank).
5. Once the payment is confirmed the payment gateway (bank) sends the confirmation page to the application that the payment is successful. This is sent through the client browser.



- The application sends a payment receipt to the user as the confirmation stating that the payment is accepted and the product will be sent to the address provided.

Second transaction:



- User "Y" selects a product from the same e-commerce web application on the same computer which was used by the user "X" and sends the request to buy the product.
- The server sends this information to the payment gateway (Bank) to send a payment page to the user.
- The payment page is sent to the user from the bank.
- The User "Y" does not enter the credit card information, instead he goes to browser history ("ctrl + h") and selects the confirmation page presented to the earlier user "X" and sends it to the server.
- The application assumes that this information is sent from the payment page and sends the product confirmation page to the user "Y" stating that the product is confirmed and his payment is successfully processed.

This transaction completes without User "Y" actually paying for the transaction.

The above scenario was possible due to a few coding weaknesses as well as an architectural flaw. It could have been prevented if the following practices were observed:

- The application should not store sensitive information like the transaction number or transaction code or session id in the URL and in the browser history.
- The application should use POST methods instead of GET methods which would eliminate the possibility of URLs having the details in the history.
- Once the transaction has ended then the session should expire so that the URL cannot be reused again.
- Further as highlighted in the earlier example, the payment gateway and the application should interact with each other directly instead of it being routed through the user's browser.
- Ideally the URL data should be exchanged in encrypted manner so that the possibility of manipulating the data is eliminated.



The above highlighted issues are just the tip of the iceberg. These issues could have been eliminated if proper steps would have been taken during the appropriate stages of the software development lifecycle and vulnerability assessment and penetration testing would definitely help reveal the security posture of the application. The fundamental gap is the lack of awareness about security threats and vulnerabilities amongst the application development community.

The value of security is generally not known till a breach occurs. It is therefore crucial we provide the necessary security awareness, guidance and information to the application developers and architects and work towards adoption of security in the entire software development life cycle.