Dynamic Analysis for Tunestore

Dynamic Analysis for Tunestore Report

Amanuel Haile

November 21st, 2024

VULNERABILITY ASSESSMENT AND SYSTEMS ASSURANCE REPORT

TABLE OF CONTENTS

1.0 General Information

1.1 Purpose

The purpose of this Dynamic Analysis for Dynamic Analysis for Tunestore report is to examine all vulnerabilities present in the XSS Demos webpage that have a severity level of low or higher, according to ZAP. Each of these vulnerabilities will be categorized as one of the following: true positive or false positive. Additionally, false negatives, which are vulnerabilities that were found in my own pentesting, but not discovered by ZAP, will be discussed.

1.2 Overview

After running a ZAP automated scan on the XSS demos application, a total of 12 vulnerabilities were found to have a security level of low or higher. The following is a list of all vulnerabilities that met this criteria:

- SQL Injection - (High)
- Cross-Site Scripting (Reflected) - (High)
- Cross-Site Scripting (Persistent) - (High)
- Session ID in URL Rewrite - (Medium)
- X-Frame-Options Header Not Set - (Medium)
- Buffer Overflow - (Medium)
- X-Content-Type-Options Header Missing - (Low)
- Absence of Anti-CSRF Tokens - (Low)
- Application Error Disclosure - (Low)

2.0 SQL Injection

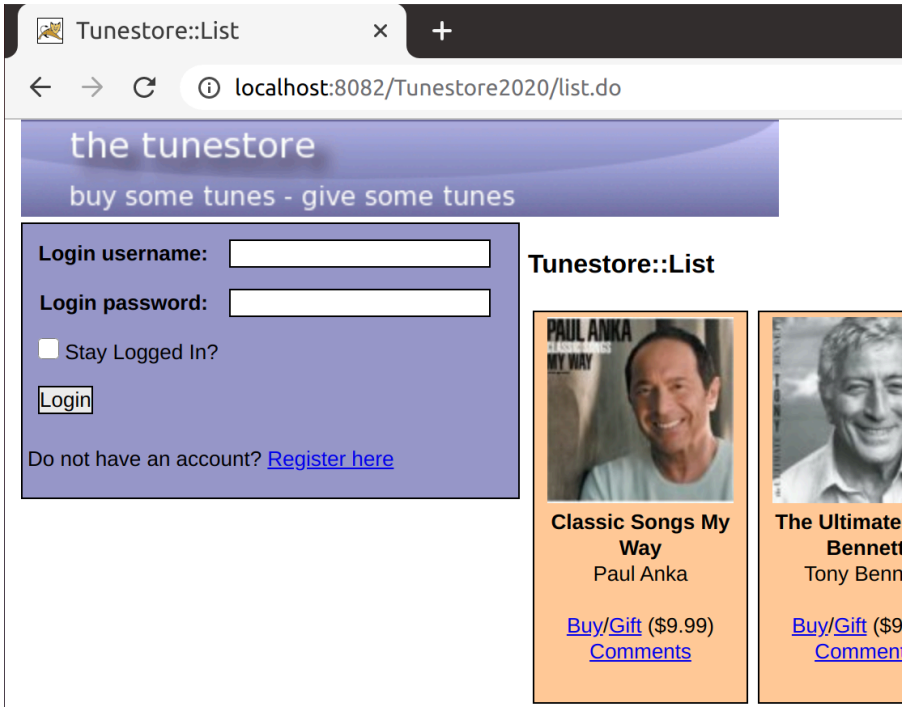
The first vulnerability that ZAP discovered was an SQL Injection vulnerability. Below is a screenshot of the ZAP Scanning Report that contains the information regarding the SQL Injection vulnerability that may be present on the Tunestore webpage:

High (Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://localhost:8082/Tunestore2020/login.do;jsessionid=82B3363823DFD1597C99EAAA4475B92B?username=amanuel%27+AND+%271%27%3D%271%27+--+&password=12345&stayLogged=true
Method	GET
Parameter	username
Attack	amanuel' AND '1'='1' --
URL	http://localhost:8082/Tunestore2020/login.do;jsessionid=82B3363823DFD1597C99EAAA4475B92B?username=amanuel&password=12345%27+AND+%271%27%3D%271&stayLogged=true
Method	GET
Parameter	password
Attack	12345' AND '1'='1

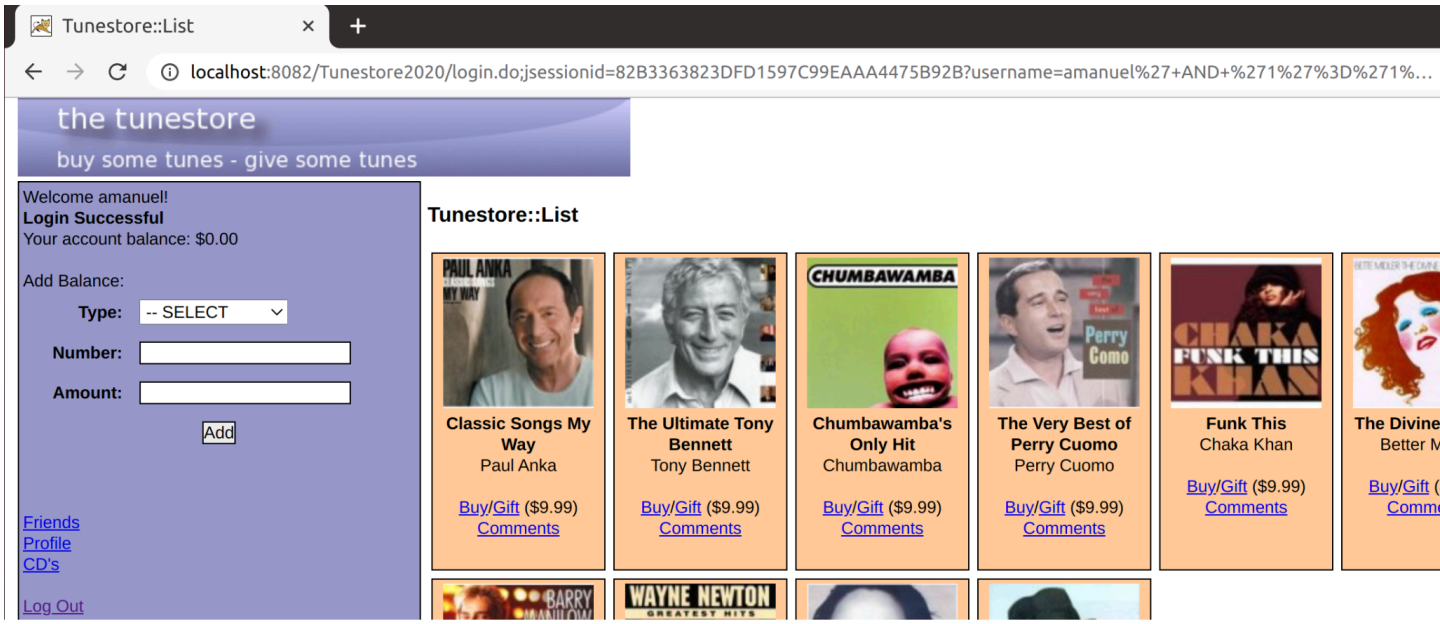
The reflected SQL Injection vulnerability is a true positive and it can be exploited by using this URL which was given by the ZAP scan:
http://localhost:8082/Tunestore2020/login.do;jsessionid=82B3363823DFD1597C99EAAA4475B92B?username=amanuel%27+AND+%271%27%3D%271%27+--+&password=12345&stayLogged=true
From:

URL	http://localhost:8082/Tunestore2020/login.do;jsessionid=82B3363823DFD1597C99EAAA4475B92B?username=amanuel%27+AND+%271%27%3D%271%27+--+&password=12345&stayLogged=true
Method	GET
Parameter	username
Attack	amanuel' AND '1'='1' --

Before the attack:



After the attack:



Entering the SQL payload bypassed authentication, granting access to a user's account in the application. This indicates that user inputs are directly passed to the database without proper sanitization.

3.0 Cross-Site Scripting (Reflected)

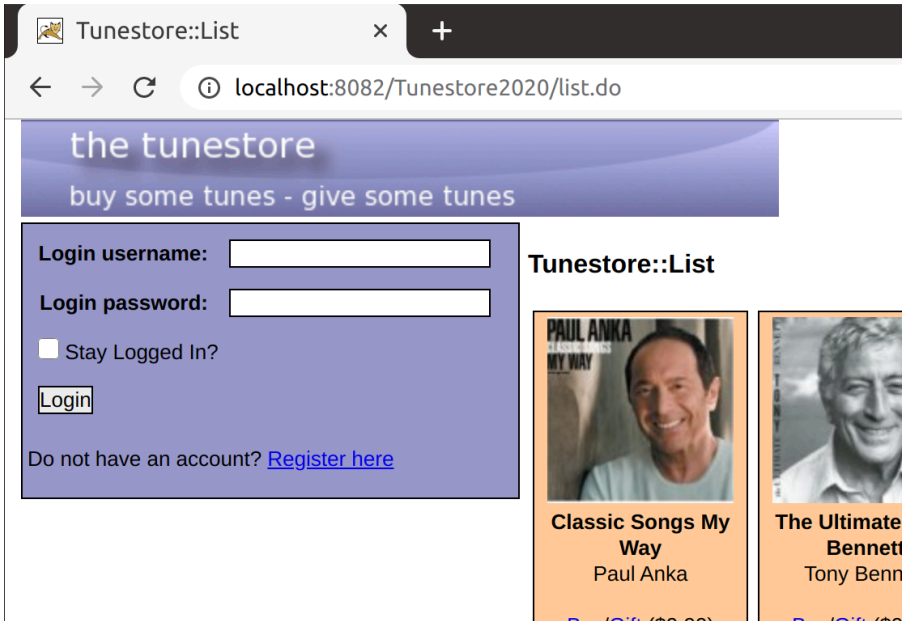
The second vulnerability that ZAP discovered was a reflected Cross-Site Scripting vulnerability. Below is a screenshot of the ZAP Scanning Report that contains the information regarding the reflected Cross-Site Scripting vulnerability that may be present on the Tunestore webpage:

High (Medium)	Cross Site Scripting (Reflected)
Description	Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML/JavaScript, but may also extend to VBScript, ActiveX, Java, Flash, or any other browser-supported technology. When an attacker gets a user's browser to execute his/her code, the code will run within the security context (or zone) of the hosting web site. With this level of privilege, the code has the ability to read, modify and transmit any sensitive data accessible by the browser. A Cross-site Scripted user could have his/her account hijacked (cookie theft), their browser redirected to another location, or possibly shown fraudulent content delivered by the web site they are visiting. Cross-site Scripting attacks essentially compromise the trust relationship between a user and the web site. Applications utilizing browser object instances which load content from the file system may execute code under the local machine zone allowing for system compromise. There are three types of Cross-site Scripting attacks: non-persistent, persistent and DOM-based. Non-persistent attacks and DOM-based attacks require a user to either visit a specially crafted link laced with malicious code, or visit a malicious web page containing a web form, which when posted to the vulnerable site, will mount the attack. Using a malicious form will oftentimes take place when the vulnerable resource only accepts HTTP POST requests. In such a case, the form can be submitted automatically, without the victim's knowledge (e.g. by using JavaScript). Upon clicking on the malicious link or submitting the malicious form, the XSS payload will get echoed back and will get interpreted by the user's browser and execute. Another technique to send almost arbitrary requests (GET and POST) is by using an embedded client, such as Adobe Flash. Persistent attacks occur when the malicious code is submitted to a web site where it's stored for a period of time. Examples of an attacker's favorite targets often include message board posts, web mail messages, and web chat software. The unsuspecting user is not required to interact with any additional site/link (e.g. an attacker site or a malicious link sent via email), just simply view the web page containing the code.
URL	http://localhost:8082/Tunestore2020/login.do;jsessionid=AEA38D74D0D65AAA983FCAB8F09E67BC?password=ZAP&stayLogged=true&username=%3C%2Fspan%3E%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E%3Cspan%3E
Method	GET
Parameter	username
Attack	<script>alert(1);</script>
Evidence	<script>alert(1);</script>

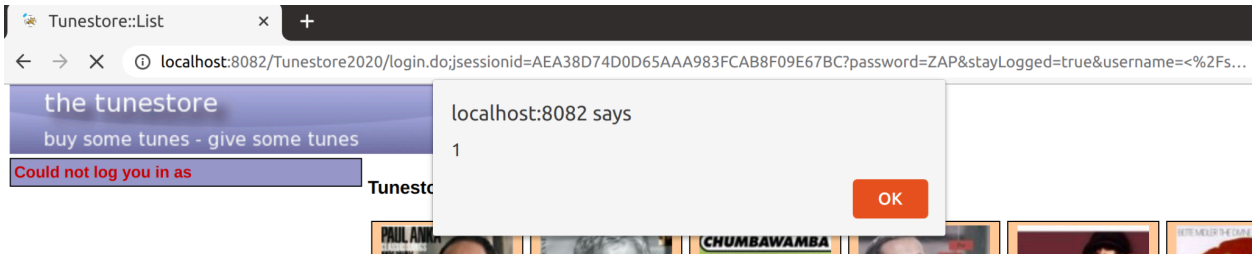
The Reflected XSS Scripting vulnerability is a true positive and it can be exploited by using this url which was given by the ZAP scan:
http://localhost:8082/Tunestore2020/login.do;jsessionid=AEA38D74D0D65AAA983FCAB8F09E67BC?password=ZAP&stayLogged=true&username=%3C%2Fspan%3E%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E%3Cspan%3E
From:

URL	http://localhost:8082/Tunestore2020/login.do;jsessionid=AEA38D74D0D65AAA983FCAB8F09E67BC?password=ZAP&stayLogged=true&username=%3C%2Fspan%3E%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E%3Cspan%3E
Method	GET
Parameter	username
Attack	<script>alert(1);</script>
Evidence	<script>alert(1);</script>

Before the attack:



After the attack:



The search query parameter is not properly sanitized, allowing JavaScript to execute. Testing this vulnerability by manually inputting the payload resulted in a browser alert, confirming the vulnerability.

4.0 Cross-Site Scripting (Persistent)

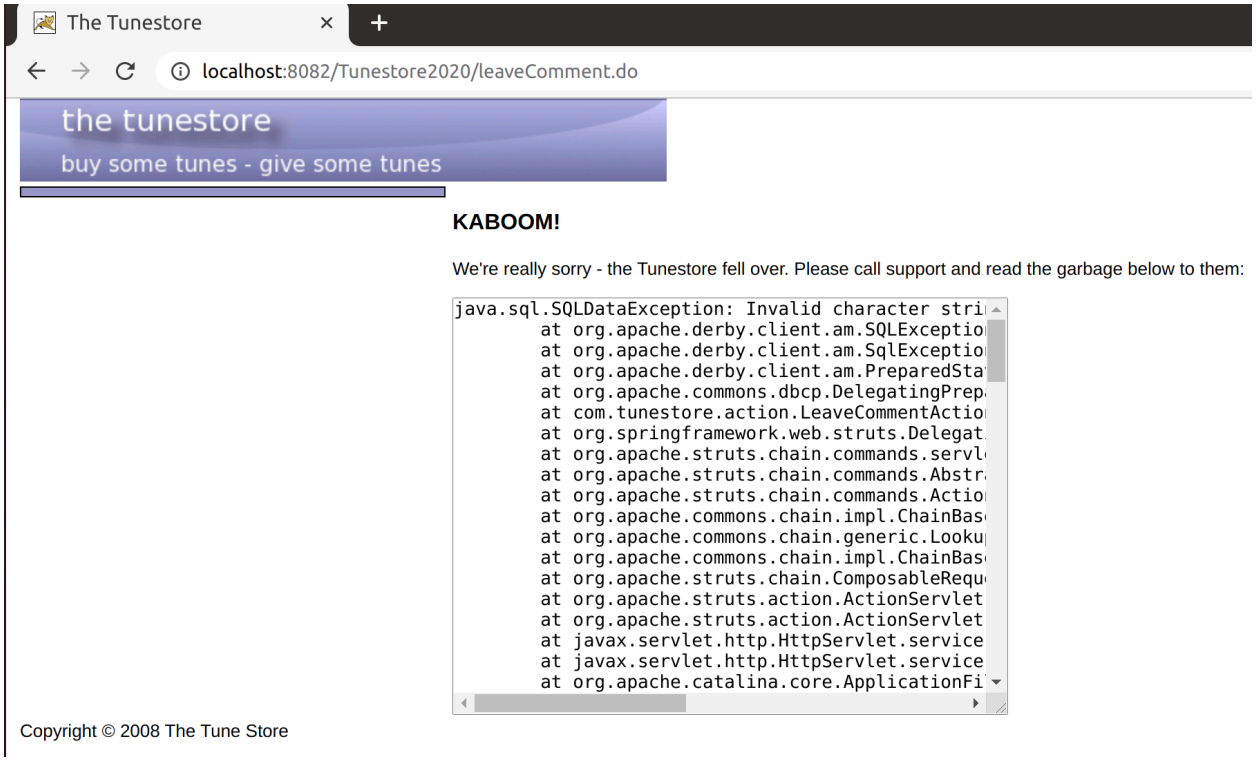
The third vulnerability that ZAP discovered was a persistent Cross-Site Scripting vulnerability. Below is a screenshot of the ZAP Scanning Report that contains the information regarding the persistent Cross Site Scripting vulnerability that may be present on the Tunestore webpage:

High (Medium)	Cross Site Scripting (Persistent)
Description	Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML/JavaScript, but may also extend to VBScript, ActiveX, Java, Flash, or any other browser-supported technology. When an attacker gets a user's browser to execute his/her code, the code will run within the security context (or zone) of the hosting web site. With this level of privilege, the code has the ability to read, modify and transmit any sensitive data accessible by the browser. A Cross-site Scripted user could have his/her account hijacked (cookie theft), their browser redirected to another location, or possibly shown fraudulent content delivered by the web site they are visiting. Cross-site Scripting attacks essentially compromise the trust relationship between a user and the web site. Applications utilizing browser object instances which load content from the file system may execute code under the local machine zone allowing for system compromise. There are three types of Cross-site Scripting attacks: non-persistent, persistent and DOM-based. Non-persistent attacks and DOM-based attacks require a user to either visit a specially crafted link laced with malicious code, or visit a malicious web page containing a web form, which when posted to the vulnerable site, will mount the attack. Using a malicious form will oftentimes take place when the vulnerable resource only accepts HTTP POST requests. In such a case, the form can be submitted automatically, without the victim's knowledge (e.g. by using JavaScript). Upon clicking on the malicious link or submitting the malicious form, the XSS payload will get echoed back and will get interpreted by the user's browser and execute. Another technique to send almost arbitrary requests (GET and POST) is by using an embedded client, such as Adobe Flash. Persistent attacks occur when the malicious code is submitted to a web site where it's stored for a period of time. Examples of an attacker's favorite targets often include message board posts, web mail messages, and web chat software. The unsuspecting user is not required to interact with any additional site/link (e.g. an attacker site or a malicious link sent via email), just simply view the web page containing the code.
URL	http://localhost:8082/Tunestore2020/leaveComment.do
Method	POST
Parameter	comment
Attack	</blockquote><script>alert(1);</script><blockquote>

The Persistent XSS Scripting vulnerability is a false positive and it can't be exploited by using this URL which was given by the ZAP scan:
http://localhost:8082/Tunestore2020/leaveComment.do

URL	http://localhost:8082/Tunestore2020/leaveComment.do
Method	POST
Parameter	comment
Attack	</blockquote><script>alert(1);</script><blockquote>

This is what shows up when the link is used:



Stored data is sanitized and validated on both input and output using a secure encoding library. This suggests that the flagged vulnerability is not exploitable.

5.0 Session ID in URL Rewrite

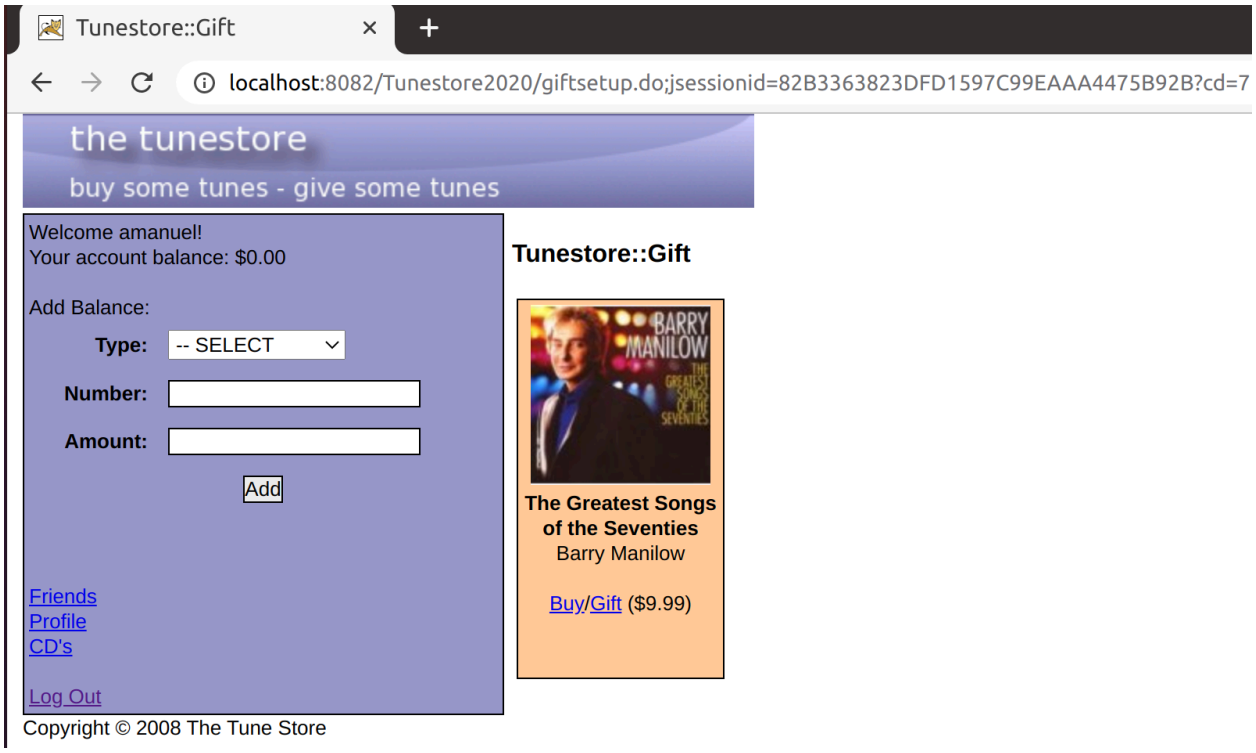
The fourth vulnerability that ZAP discovered was a Session ID in URL Rewrite vulnerability. Below is a screenshot of the ZAP Scanning Report that contains the information regarding this vulnerability that may be present on the Tunestore webpage:

Medium (High)	Session ID in URL Rewrite
Description	URL rewrite is used to track user session ID. The session ID may be disclosed via cross-site referer header. In addition, the session ID might be stored in browser history or server logs.
URL	http://localhost:8082/Tunestore2020/giftsetup.do;jsessionid=82B3363823DFD1597C99EAAA4475B92B?cd=7
Method	GET
Evidence	jsessionid=82B3363823DFD1597C99EAAA4475B92B

The Session ID in URL Rewrite vulnerability is a false positive and it can't be exploited by using this URL when logged in, which was given by the ZAP scan: http://localhost:8082/Tunestore2020/giftsetup.do;jsessionid=82B3363823DFD1597C99EAAA4475B92B?cd=7

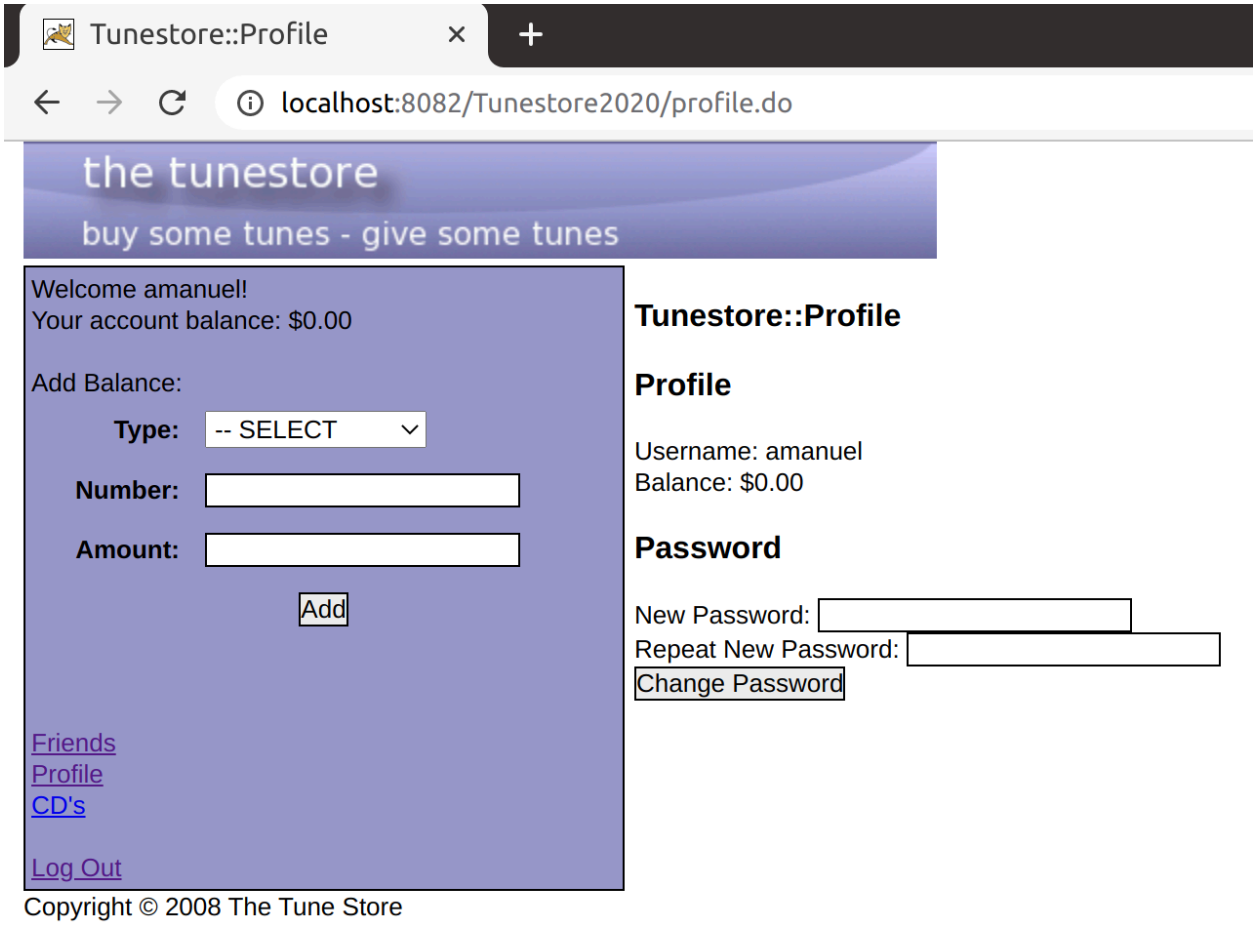
URL	http://localhost:8082/Tunestore2020/giftsetup.do;jsessionid=82B3363823DFD1597C99EAAA4475B92B?cd=7
Method	GET
Evidence	jsessionid=82B3363823DFD1597C99EAAA4475B92B

When logged in, this is what shows up when the link is used:



However, if the user decides to click on any link on the page, it ignores the session-id in the URL, and still keeps the current actual session for the user.

(Clicking on the profile link):



6.0 X-Frame-Options Header Not Set

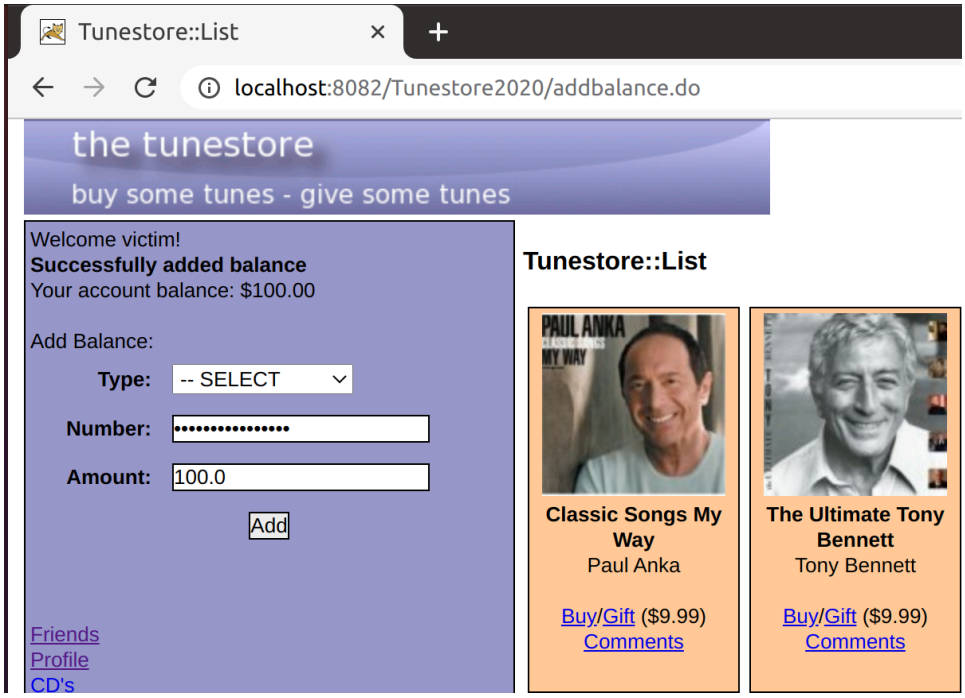
The fifth vulnerability that ZAP discovered was that some responses lacked the X-Frame-Options header, which could allow clickjacking attacks. Below is a screenshot of the ZAP Scanning Report that contains the information regarding this vulnerability that may be present on the Tunestore webpage:

Medium (Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://localhost:8082/Tunestore2020/giftsetup.do;jsessionid=82B3363823DFD1597C99EAAA4475B92B?cd=3
Method	GET
Parameter	X-Frame-Options
URL	http://localhost:8082/Tunestore2020/logout.do
Method	GET
Parameter	X-Frame-Options
URL	http://localhost:8082/Tunestore2020/profile.do;jsessionid=82B3363823DFD1597C99EAAA4475B92B
Method	GET
Parameter	X-Frame-Options
URL	http://localhost:8082/Tunestore2020/buy.do;jsessionid=A3AEEC08821DA0B02D320C18C20A301B?cd=1
Method	GET
Parameter	X-Frame-Options

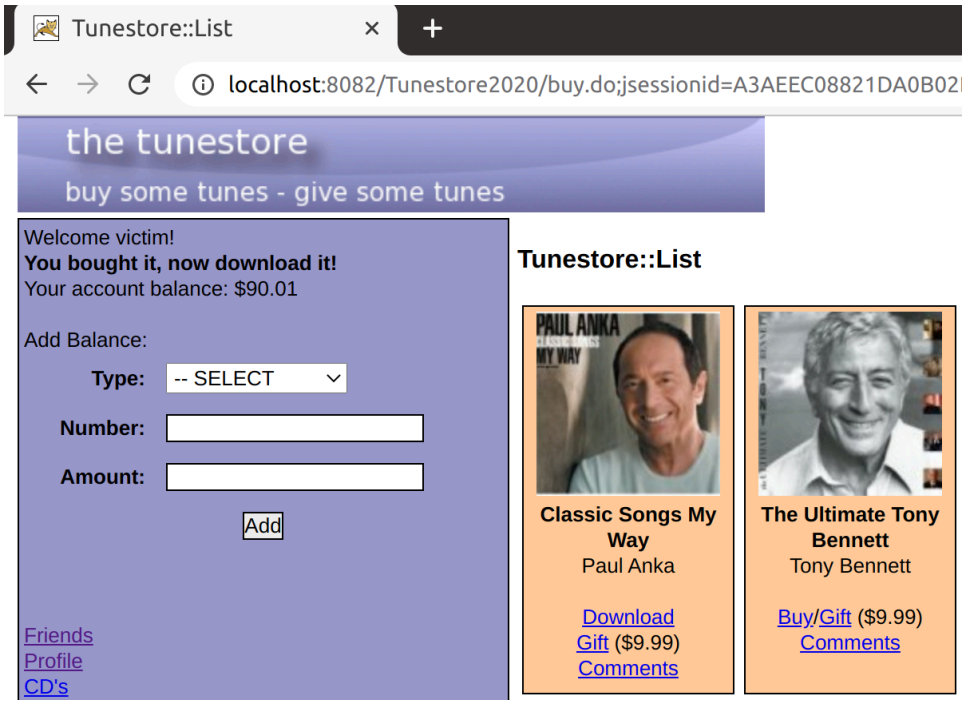
This vulnerability is a true positive and it can be exploited by using this url which was given by the ZAP scan: http://localhost:8082/Tunestore2020/buy.do;jsessionid=A3AEEC08821DA0B02D320C18C20A301B?cd=1

URL	http://localhost:8082/Tunestore2020/buy.do;jsessionid=A3AEEC08821DA0B02D320C18C20A301B?cd=1
Method	GET
Parameter	X-Frame-Options

If a user is currently logged in and they click on this link, it immediately buys the album using the users funds:



After the link is visited:



7.0 Buffer Overflow

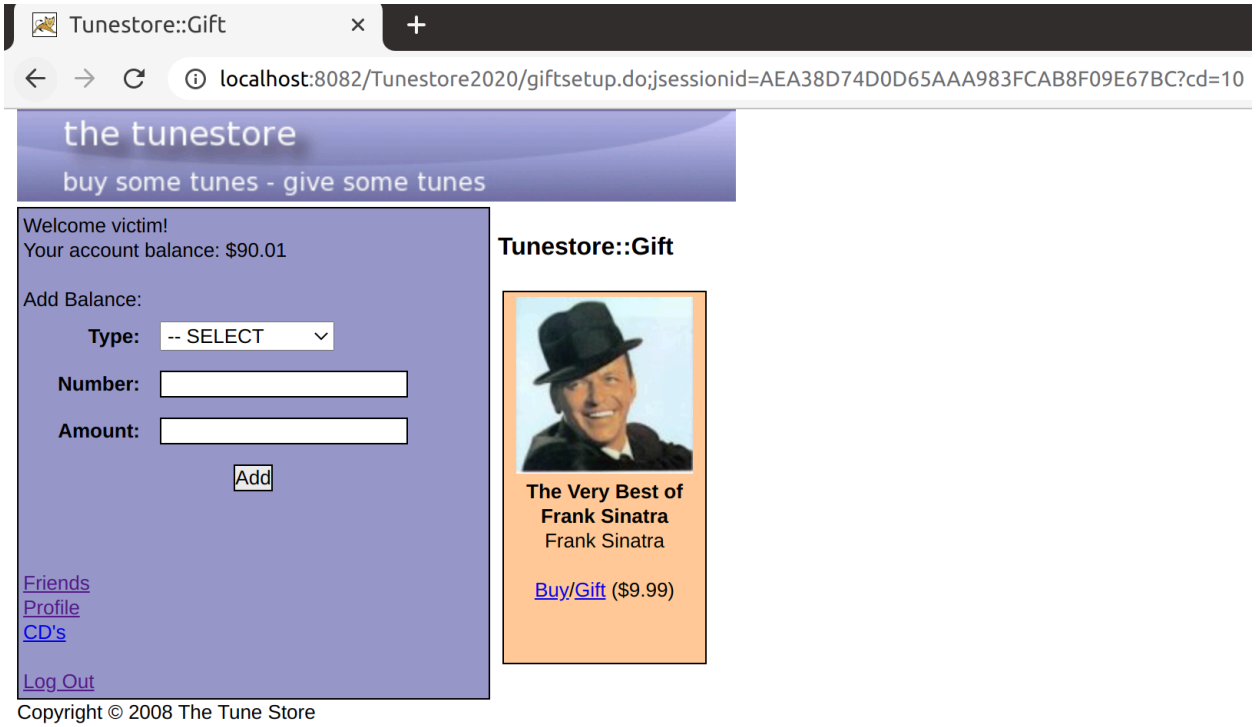
The sixth vulnerability that ZAP discovered was that some input fields were potentially vulnerable to buffer overflow attacks when large inputs were submitted. Below is a screenshot of the ZAP Scanning Report that contains the information regarding the Buffer Overflow vulnerability that may be present on the Tunestore webpage:

Medium (Medium)	Buffer Overflow
Description	Buffer overflow errors are characterized by the overwriting of memory spaces of the background web process, which should have never been modified intentionally or unintentionally. Overwriting values of the IP (Instruction Pointer), BP (Base Pointer) and other registers causes exceptions, segmentation faults, and other process errors to occur. Usually these errors end execution of the application in an unexpected way.
URL	http://localhost:8082/Tunestore2020/giftsetup.do;jsessionid=AEA38D74D0D65AAA983FCAB8F09E67BC?cd=10
Method	GET
Parameter	cd
Evidence	GET http://localhost:8082/Tunestore2020 /giftsetup.do;jsessionid=AEA38D74D0D65AAA983FCAB8F09E67BC?cd=SlvFXynPTmENYmWYfaxiCQYPrNpLrWoAtSfwjPegytBnBCbEMkMBZvsmjTnrWFwZTA rJXHMadrAmJMAUeqLDCQsBjWLVPHxtVgdskDBDHuhKrIShUjyQNpAWwvjPfxHOPdbnlsyiWiNliLoUdjcjNYpLkkufslmenhlTgGlCcNQcRRWEDCNEDobXVxPnJN nQpcqncrXruQgRcfNoENpjELSlExEEgIsvMEpycObHiKgRpFnpgwRJatFkgbfqwgHUFLgYPNZyirJRgARiNWkuhWmlnWOjYinnJbilNaKKxNJwIUgQqrDFrkZGuapF NnjVorQrmcpShqEDUbNKJPLaUUjVvjWEPJfPUxTtPTpeYBQWhIAjcZedTCXIvupsoKTvJiRXrlfjMjopcLaCyDpCrAWSHiikilhDoNhxBdXRDDOxdEOotTwjgdupxvhk DvmulhvaauaabnuDdRiybhCqxADeRVQevmToPlopGZWwHklJXwkvdvXhiRjyAXueMIjEyLkeYgXGEaWQieqxGDpOSKNOcZgSZaTCXLvUNIXiJsxpYEVyYuHncxo JLaxBlVSSOSdVCnQwhNJEXKMMZKrXXikPjEFZfiRkjnfHeCXvUZlJeKJmcykUDVPYKtApyGjfSSYpQYfauygSGAruIGvxDIhgllUSZFMoNFoZOHktdYwjVkNrQlKlVg guemYOvvJkGJAaUDKCGkJuWllbemgKBnMydDqqrJmZSKCQnKwCJQODacYWDYZcNoaIUcnRexFFfvhRtnWsOHwMhkgKRTVLfYiDrshjhKtxAMOLeRxKlmZPlX kkaLwhVgyVHQVKHGSEUBnWdXgtMpWUjCYqRmJShXZBiPufmaQGXKKHOkOYiEOaLYxaJmLZtMRDEfqQtgTZqZveaqIjNcfHiUwQHiCykXyXfgktPudXASLLjTh hrMlHCZcADuSaOZWoTWpgPTCamtTlipZAXVPPcPfIUJvJvuRtxhVAUpGKYXfXYvoOWvisglFJECetDQWEDjLblUMXSZgntnbHuOYThKNtBJWmOTtckMWPTNkR CTiHcVDYSdaXqVFcAsUvrIhpQKckhrfPXfFqSajlETDxHVTsyOjgdSsLBPhvAGKCFuBmfEbNeFHwVkesIHDVPckgSPiLNMEcohEVIWxkipDetmRYIFgLisaPcgQVv CtJAohwqfraZVFBpxcmCRhbOnvsFbbLCEUKeNORcUjAvZEHvfIZEuQjlYdHNuLZSwqYfSArnffbsFVZcJpZVYnPgppIKIFKHVxIsLnTijkNePyfMANampIDGvHbhGQ oOlisqyWhSmmVwOFgUnNHBuTHrTSiuMfiWmqNATZNuGaKONoeoFkBQNkJfDHBgSQIyemcVLqITAkQGLcAmjpSWkZGJKSJlgWKOcSiFcFdvSITZoFkfpOavcA uGIvLOxSbtYmSheXJXqBFQvlZWPoZbZDjxAwkIKbHeIgcoWQSPYtugXATuuojCIpcCrjxbAyIPKgJOGRnNmjoweUkclkMcstpbduLyePAIDADUOWZmqVDWcYNkB arwrfeTdWmwVUKPPyRuWJBgpHAyuKAIZPvkWCTSxjtQDuxHbrqRXKnsNdIkikJVOpCRNpLDfVrWjDTACXktvbdraJQbdaAOkAbEaZOBFeSTKmtFBXSKbTkXoQ MeTZMSredZJPZJFjPcGmeuLAUVXAdFQrDqkjDdWUwshYDTXaysMcfeKRndUvRCIZMZAMjdDLiZpjWusgpAjiikTOBdKZrBQITfgPJkZoAibovTvQdYBeMueqROg oyyFhmpoyCYsSgbnFcdiWdgrrIDILtQdbdPrcOjskEdhqOyEuHWXuqpuNfEdvnxgybnwKytyvcSbumTknbjrUxhXINbtuNadbHjUXYJZIvteyQhfHEbnbFEWSEuaUFrKI ndUfTPXnMAjwANKeIryuAVcJugdPkBvmvpraMkFOJMJbKvlMTBogKQwLulOWdihlplqBtNuhesibHwtbsYKRiQxAETyGrDCOEOpxeGCfVOtgJvqVMixKgZgnGpoux hXgVocQbxYAbTQdaHswlNmHeNlligViCjtoqViDpogjNouAvhRTc HTTP/1.1 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:71.0) Gecko/20100101 Firefox/71.0 Pragma: no-cache Cache-Control: no-cache Content-Length: 0 Cookie: JSESSIONID=5E91B5BC263B1974BA06B704F9DEFE07 Host: localhost:8082

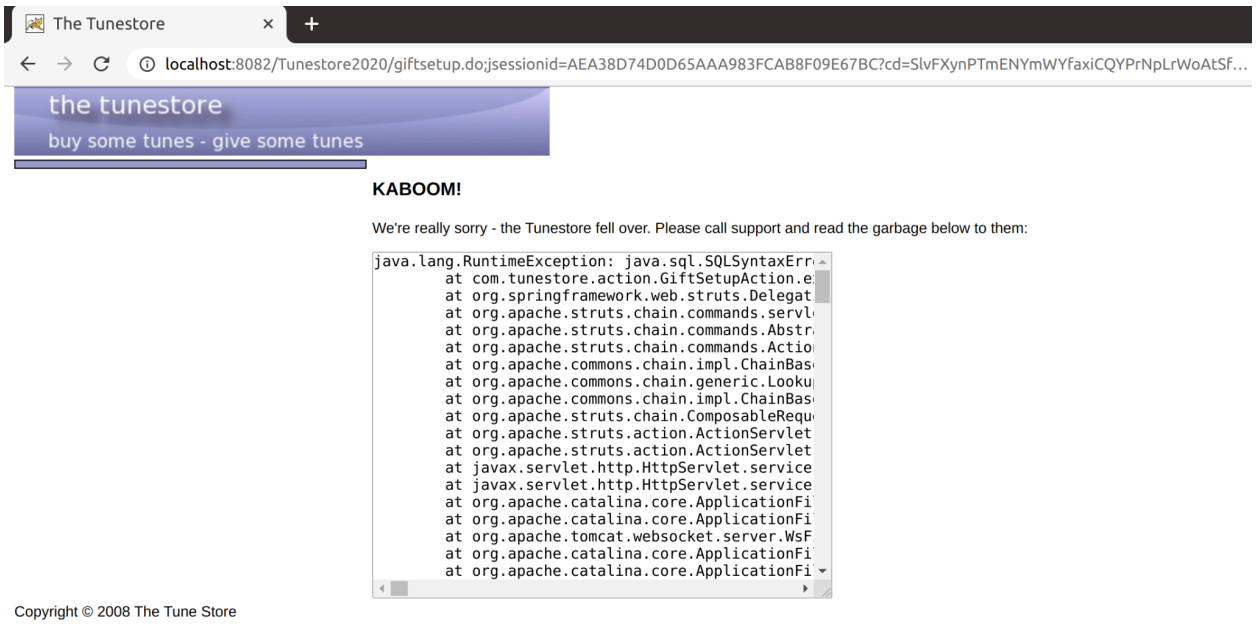
This vulnerability is a true positive and it can be exploited by using this url which was given by the ZAP scan:

URL	http://localhost:8082/Tunestore2020/giftsetup.do;jsessionid=AEA38D74D0D65AAA983FCAB8F09E67BC?cd=10
Method	GET
Parameter	cd
Evidence	GET http://localhost:8082/Tunestore2020 /giftsetup.do;jsessionid=AEA38D74D0D65AAA983FCAB8F09E67BC?cd=SlvFXynPTmENYmWYfaxiCQYPrNpLrWoAtSfwjPegytBnBCbEMkMBZvsmjTnrWFwZTA rJXHMadrAmJMAUeqLDCQsBjWLVPHxtVgdskDBDHuhKrIShUjyQNpAWwvjPfxHOPdbnlsyiWiNliLoUdjcjNYpLkkufslmenhlTgGlCcNQcRRWEDCNEDobXVxPnJN nQpcqncrXruQgRcfNoENpjELSlExEEgIsvMEpycObHiKgRpFnpgwRJatFkgbfqwgHUFLgYPNZyirJRgARiNWkuhWmlnWOjYinnJbilNaKKxNJwIUgQqrDFrkZGuapF NnjVorQrmcpShqEDUbNKJPLaUUjVvjWEPJfPUxTtPTpeYBQWhIAjcZedTCXIvupsoKTvJiRXrlfjMjopcLaCyDpCrAWSHiikilhDoNhxBdXRDDOxdEOotTwjgdupxvhk DvmulhvaauaabnuDdRiybhCqxADeRVQevmToPlopGZWwHklJXwkvdvXhiRjyAXueMIjEyLkeYgXGEaWQieqxGDpOSKNOcZgSZaTCXLvUNIXiJsxpYEVyYuHncxo JLaxBlVSSOSdVCnQwhNJEXKMMZKrXXikPjEFZfiRkjnfHeCXvUZlJeKJmcykUDVPYKtApyGjfSSYpQYfauygSGAruIGvxDIhgllUSZFMoNFoZOHktdYwjVkNrQlKlVg guemYOvvJkGJAaUDKCGkJuWllbemgKBnMydDqqrJmZSKCQnKwCJQODacYWDYZcNoaIUcnRexFFfvhRtnWsOHwMhkgKRTVLfYiDrshjhKtxAMOLeRxKlmZPlX kkaLwhVgyVHQVKHGSEUBnWdXgtMpWUjCYqRmJShXZBiPufmaQGXKKHOkOYiEOaLYxaJmLZtMRDEfqQtgTZqZveaqIjNcfHiUwQHiCykXyXfgktPudXASLLjTh hrMlHCZcADuSaOZWoTWpgPTCamtTlipZAXVPPcPfIUJvJvuRtxhVAUpGKYXfXYvoOWvisglFJECetDQWEDjLblUMXSZgntnbHuOYThKNtBJWmOTtckMWPTNkR CTiHcVDYSdaXqVFcAsUvrIhpQKckhrfPXfFqSajlETDxHVTsyOjgdSsLBPhvAGKCFuBmfEbNeFHwVkesIHDVPckgSPiLNMEcohEVIWxkipDetmRYIFgLisaPcgQVv CtJAohwqfraZVFBpxcmCRhbOnvsFbbLCEUKeNORcUjAvZEHvfIZEuQjlYdHNuLZSwqYfSArnffbsFVZcJpZVYnPgppIKIFKHVxIsLnTijkNePyfMANampIDGvHbhGQ oOlisqyWhSmmVwOFgUnNHBuTHrTSiuMfiWmqNATZNuGaKONoeoFkBQNkJfDHBgSQIyemcVLqITAkQGLcAmjpSWkZGJKSJlgWKOcSiFcFdvSITZoFkfpOavcA uGIvLOxSbtYmSheXJXqBFQvlZWPoZbZDjxAwkIKbHeIgcoWQSPYtugXATuuojCIpcCrjxbAyIPKgJOGRnNmjoweUkclkMcstpbduLyePAIDADUOWZmqVDWcYNkB arwrfeTdWmwVUKPPyRuWJBgpHAyuKAIZPvkWCTSxjtQDuxHbrqRXKnsNdIkikJVOpCRNpLDfVrWjDTACXktvbdraJQbdaAOkAbEaZOBFeSTKmtFBXSKbTkXoQ MeTZMSredZJPZJFjPcGmeuLAUVXAdFQrDqkjDdWUwshYDTXaysMcfeKRndUvRCIZMZAMjdDLiZpjWusgpAjiikTOBdKZrBQITfgPJkZoAibovTvQdYBeMueqROg oyyFhmpoyCYsSgbnFcdiWdgrrIDILtQdbdPrcOjskEdhqOyEuHWXuqpuNfEdvnxgybnwKytyvcSbumTknbjrUxhXINbtuNadbHjUXYJZIvteyQhfHEbnbFEWSEuaUFrKI ndUfTPXnMAjwANKeIryuAVcJugdPkBvmvpraMkFOJMJbKvlMTBogKQwLulOWdihlplqBtNuhesibHwtbsYKRiQxAETyGrDCOEOpxeGCfVOtgJvqVMixKgZgnGpoux hXgVocQbxYAbTQdaHswlNmHeNlligViCjtoqViDpogjNouAvhRTc HTTP/1.1 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:71.0) Gecko/20100101 Firefox/71.0 Pragma: no-cache Cache-Control: no-cache Content-Length: 0 Cookie: JSESSIONID=5E91B5BC263B1974BA06B704F9DEFE07 Host: localhost:8082

If you were to visit the shorter url, you would be brought to this page which clearly doesn't contain the presence of a buffer overflow:



But visiting the longer url would bring you this, which shows that the Session ID was clearly too long:



8.0 X-Content-Type-Options Header Missing

The seventh vulnerability that ZAP discovered was an X-Content-Type-Options Header Missing vulnerability. Below is a screenshot of the ZAP Scanning Report that contains the information regarding this vulnerability that may be present on the Tunestore webpage:

Low (Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://localhost:8082/Tunestore2020/giftsetup.do;jsessionid=82B3363823DFD1597C99EAAA4475B92B?cd=2
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8082/Tunestore2020/list.do;jsessionid=82B3363823DFD1597C99EAAA4475B92B
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8082/Tunestore2020/images/bennett.jpg;jsessionid=A3AEEC08821DA0B02D320C18C20A301B
Method	GET
Parameter	X-Content-Type-Options

The flagged URL (http://localhost:8082/Tunestore2020/giftsetup.do) and its corresponding parameter (X-Content-Type-Option) are not associated with sensitive or executable data. The absence of the X-Content-Type-Options header in the Tunestore app does not pose a meaningful security risk. The resources on this site are served with proper Content-Type headers, and modern browser behavior mitigates the need for the no-sniff directive. This would make it a false positive.

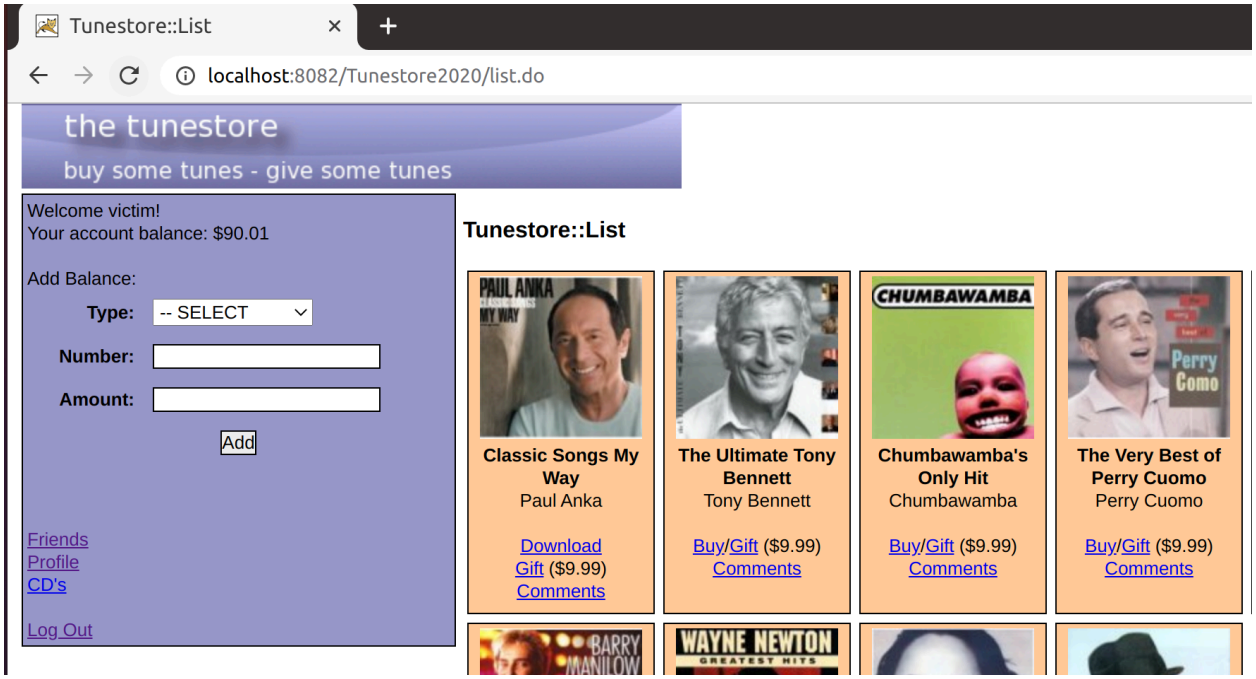
9.0 X-Content-Type-Options Header Missing

The eighth vulnerability that ZAP discovered was an X-Content-Type-Options Header Missing vulnerability. Below is a screenshot of the ZAP Scanning Report that contains the information regarding this vulnerability that may be present on the Tunestore webpage:

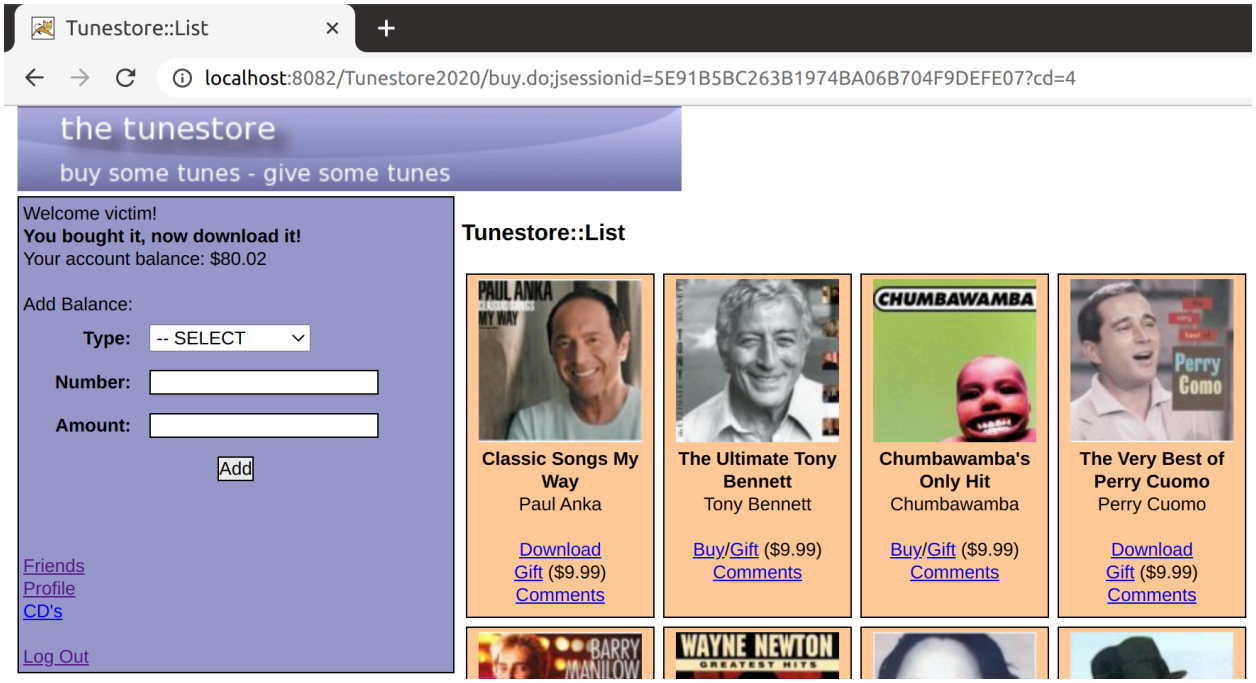
Low (Medium)	Absence of Anti-CSRF Tokens
Description	No Anti-CSRF tokens were found in a HTML submission form. A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf. CSRF attacks are effective in a number of situations, including: * The victim has an active session on the target site. * The victim is authenticated via HTTP auth on the target site. * The victim is on the same local network as the target site. CSRF has primarily been used to perform an action against a target site using the victim's privileges, but recent techniques have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-origin policy.
URL	http://localhost:8082/Tunestore2020/buy.do;jsessionid=5E91B5BC263B1974BA06B704F9DEFE07?cd=4
Method	GET
Evidence	<form name="loginForm" method="get" action="/Tunestore2020/login.do">

This vulnerability is a true positive and it can be exploited while the victim is logged in by using this url which was given by the ZAP scan: http://localhost:8082/Tunestore2020/buy.do;jsessionid=5E91B5BC263B1974BA06B704F9DEFE07?cd=4

This would buy the CD automatically when clicked on by the victim:
Before the link is clicked:



After the link is clicked:



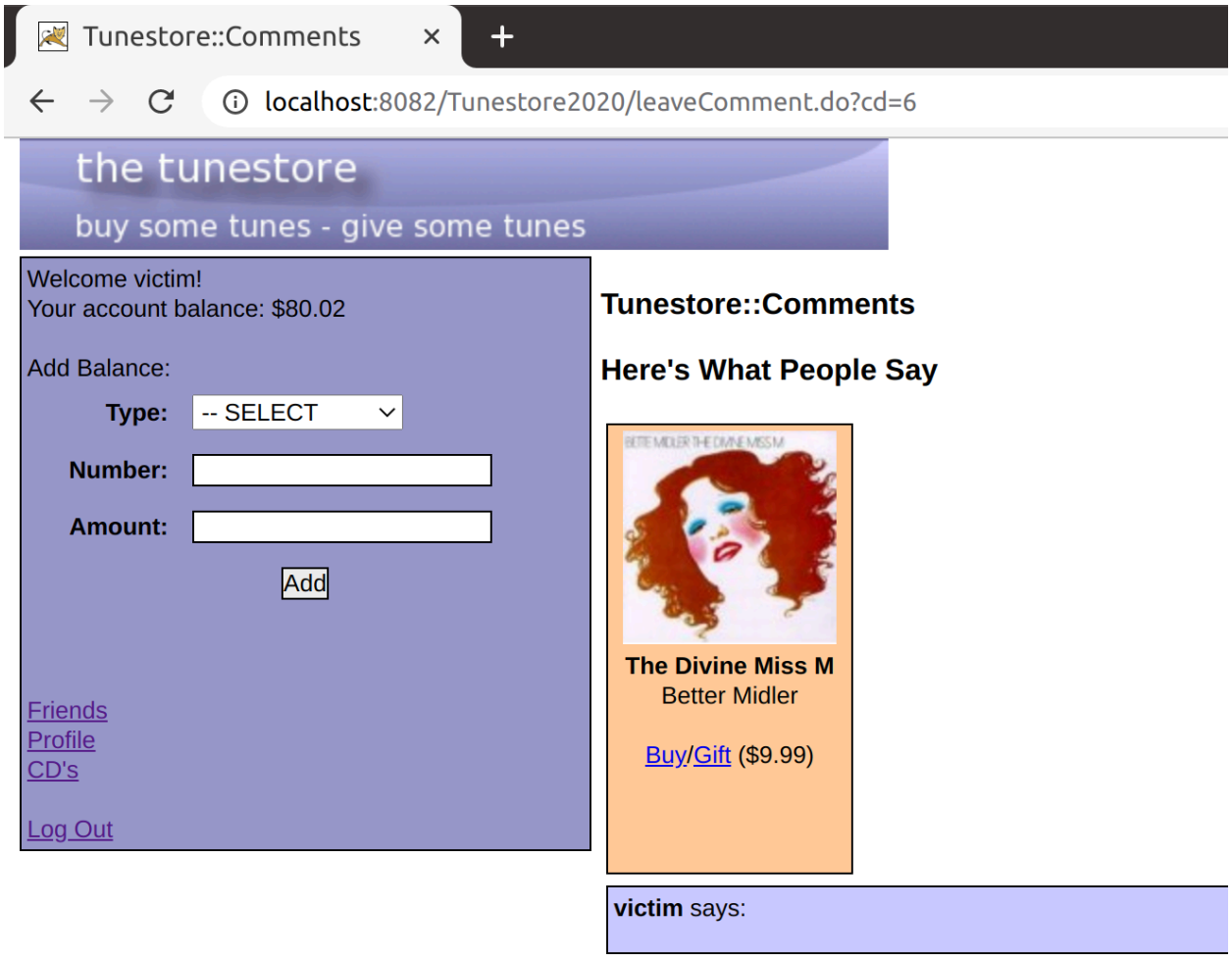
10.0 Application Error Disclosure

The ninth vulnerability that ZAP discovered was an Application Error Disclosure vulnerability. Below is a screenshot of the ZAP Scanning Report that contains the information regarding this vulnerability that may be present on the Tunestore webpage:

Low (Medium)	Application Error Disclosure
Description	This page contains an error/warning message that may disclose sensitive information like the location of the file that produced the unhandled exception. This information can be used to launch further attacks against the web application. The alert could be a false positive if the error message is found inside a documentation page.
URL	http://localhost:8082/Tunestore2020/leaveComment.do
Method	GET
Evidence	HTTP/1.1 500

The evidence in the report points to an HTTP 500 error (HTTP/1.1 500) on the page at leaveComment.do. However, there is no indication in the report that the error message discloses sensitive information. This means that it is a false positive.

Going to a link like: http://localhost:8082/Tunestore2020/leaveComment.do?cd=6, will just lead to an empty comment being made with no information being exposed:



11.0 False Negatives

1. Broken Access Control Vulnerability
Why it was missed: ZAP primarily focuses on common vulnerabilities like XSS, SQL injection, etc., and may not have deep visibility into the logic of role-based access controls or fine-grained access control mechanisms. It may not thoroughly test for privilege escalation or unauthorized access based on different user roles unless explicitly configured to do so (such as through authenticated crawling or user role simulation).
Reason for False Negative: ZAP might not have interacted with the application under different user roles or privileges.

2. DOM-XSS Vulnerability
Why it was missed: ZAP's automated scanning focuses on detecting reflected and stored XSS vulnerabilities, which are server-side vulnerabilities. However, DOM-XSS vulnerabilities are client-side issues. ZAP may not detect these because it doesn't simulate the client-side JavaScript environment in enough detail to detect DOM manipulation or issues triggered by user interaction.
Reason for False Negative: ZAP might not have fully executed or tested the JavaScript on the page, making it less likely to detect vulnerabilities that rely on client-side scripting behavior.

3. Clickjacking Vulnerability
Why it was missed: ZAP does not specifically focus on client-side UI vulnerabilities like clickjacking, especially if the site is not protected with security headers like X-Frame-Options or Content-Security-Policy (CSP) directives.
Reason for False Negative: ZAP's automated scan may not have triggered the attack vector because it does not always simulate user interactions like clicking within an iframe. Clickjacking vulnerabilities usually require manual testing.

4. Command Injection
Why it was missed: Command injection requires a specialized input processed by the system. ZAP usually does not test for command injection in detail unless the application explicitly sends user input to the system shell.
Reason for False Negative: ZAP primarily scans for web-based vulnerabilities (e.g., XSS, SQLi), and command injection vulnerabilities often require specific payloads and interactions that ZAP does not automatically test for.

5. Path Manipulation
Why it was missed: ZAP usually scans for well-known vulnerabilities like directory traversal but might not test for more subtle forms of path manipulation, such as those hidden in parameters or URL query strings. ZAP may not detect if the vulnerability requires more complex interaction with the server's file system.
Reason for False Negative: Path manipulation issues often require testing input and interactions with the file system, which may not be detected during an automated scan unless specific test cases are designed to trigger such flaws.

6. SMTP Vulnerability
Why it was missed: ZAP's automated scanning does not usually assess email functionality or test for vulnerabilities in email protocols like SMTP. Email-related vulnerabilities are often missed unless explicitly tested for.
Reason for False Negative: SMTP testing usually requires manual interaction or specialized email-based penetration testing tools, which go beyond what ZAP's automated scan checks for.

7. Handling Exceptions
Why it was missed: ZAP's automated scan does not specifically focus on error handling or exception management.
Reason for False Negative: This issue involves reviewing how the application handles exceptions at the code level, which is outside of ZAP's standard testing scope.

8. XPath Query Vulnerability
Why it was missed: ZAP may not detect XPath injection vulnerabilities unless the application directly exposes an XML API or query interface in a way that can be easily exploited. XPath injection often requires specific and structured inputs, which ZAP may not automatically test.
Reason for False Negative: ZAP is designed to detect more common vulnerabilities and may not specifically test for XPath injection. This vulnerability requires special tests for XML inputs and interactions specifically made for the app.