

SSRF

A10:2021 – Falsificación de Solicitudes del Lado del Servidor (SSRF)





A10:2021 – Falsificación de Solicitudes del Lado del Servidor (SSRF)

Descripción

Las fallas de SSRF ocurren cuando una aplicación web está obteniendo un recurso remoto sin validar la URL proporcionada por el usuario. Permite que un atacante coaccione a la aplicación para que envíe una solicitud falsificada a un destino inesperado, incluso cuando está protegido por un firewall, VPN u otro tipo de lista de control de acceso a la red (ACL).

Dado que las aplicaciones web modernas brindan a los usuarios finales funciones convenientes, la búsqueda de una URL se convierte en un escenario común. Como resultado, la incidencia de SSRF está aumentando. Además, la gravedad de SSRF es cada vez mayor debido a los servicios en la nube y la complejidad de las arquitecturas.



SSRF

Ejemplos de escenarios de ataque

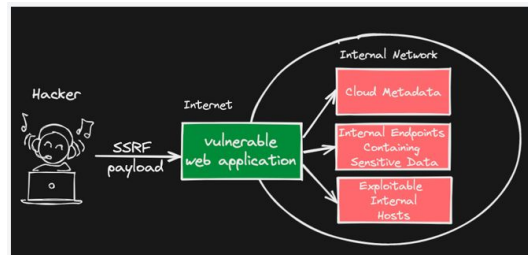
Los atacantes pueden usar SSRF para atacar sistemas protegidos detrás de firewalls de aplicaciones web, firewalls o ACLs de red, utilizando escenarios tales como:

Escenario #1: Escaneo de puertos de servidores internos – Si la arquitectura de red no se encuentra segmentada, los atacantes pueden trazar un mapa de las redes internas y determinar si los puertos están abiertos o cerrados en los servidores internos a partir de los resultados de la conexión o del tiempo transcurrido para conectar o rechazar las conexiones de payload SSRF.

Escenario #2: Exposición de datos sensibles: los atacantes pueden acceder a archivos locales como servicios internos para obtener información confidencial como `file:///etc/passwd</code> y http://localhost:28017/.`

Escenario #3: Acceso al almacenamiento de metadatos de los servicios en la nube: la mayoría de los proveedores de la nube tienen almacenamiento de metadatos como `http://169.254.169.254/`. Un atacante puede leer los metadatos para obtener información confidencial.

Escenario #4: Exposición de los servicios internos: el atacante puede abusar de los servicios internos para realizar más ataques, como la ejecución remota de código (RCE) o la denegación de servicio (DoS).





A10:2021 – SSRF

Cómo se previene

Desde la capa de red

- Segmente la funcionalidad de acceso a recursos remotos en redes separadas para reducir el impacto de SSRF
- Haga cumplir las políticas de firewall "denegar por defecto" o las reglas de control de acceso a la red para bloquear todo el tráfico de la intranet excepto el esencial.

Consejos:

- ~ Establezca la propiedad y un ciclo de vida para las reglas de firewall basadas en aplicaciones.
- ~ Registre en logs todos los flujos de red aceptados y bloqueados en firewalls (consulte [A09: 2021-Fallas en el Registro y Monitoreo](#)).



A10:2021 – SSRF

Cómo se previene

Desde la capa de aplicación:

- Sanitice y valide todos los datos de entrada proporcionados por el cliente
- Haga cumplir el esquema de URL, el puerto y destino a través de una lista positiva de items permitidos
- No envíe respuestas en formato "crudo" a los clientes
- Deshabilite las redirecciones HTTP
- Tenga en cuenta la coherencia de la URL para evitar ataques como el enlace de DNS y las condiciones de carrera de "tiempo de verificación, tiempo de uso" (TOCTOU por sus siglas en inglés)

No mitigue SSRF mediante el uso de una lista de denegación o una expresión regular. Los atacantes poseen listas de payloads, herramientas y habilidades para eludir las listas de denegación.



Experimento

Evaluemos la criticidad de las siguientes

Ranear de más grave a menos grave las siguientes vulnerabilidades:

Para cada vuln, ayuda pensar en la situación más riesgosa que podría darse con la misma

- XXS
- SQLi
- SSRF
- Open Redirect
- Entidades Externas XML (XXE)



A10:2021 – Falsificación de Solicitudes del Lado del Servidor (SSRF)

Descripción

Las fallas de SSRF ocurren cuando una aplicación web está obteniendo un recurso remoto sin validar la URL proporcionada por el usuario. Permite que un atacante coaccione a la aplicación para que envíe una solicitud falsificada a un destino inesperado, incluso cuando está protegido por un firewall, VPN u otro tipo de lista de control de acceso a la red (ACL).

Dado que las aplicaciones web modernas brindan a los usuarios finales funciones convenientes, la búsqueda de una URL se convierte en un escenario común. Como resultado, la incidencia de SSRF está aumentando. Además, la gravedad de SSRF es cada vez mayor debido a los servicios en la nube y la complejidad de las arquitecturas.

Solo una partecita de las preocupaciones de seguridad en entornos cloud

Manejo de identidades y permisos en AWS metadata y riesgos asociados

– Si bien la charla hace foco en AWS, es aplicable a diferentes entornos nubes públicas y privadas. –

Matías Ferrigno & Nicolás Macia



Identity & Access Management



IAM Identities

Users: Un usuario con capacidad de autenticarse y hacer uso de recursos mediante credenciales.

Las credenciales pueden ser:

- Usuario y Contraseña
- Access Keys (Llaves programáticas)

Grupos: Coleccion de usuarios con requerimientos en común.

Roles: Un rol tiene vinculado un conjunto de políticas.

- Los roles pueden ser asumidos por entidades autorizadas (Servicios / Usuarios).

Policy

Define permisos que pueden ser asignados por un administrador a una entidad.



IAM Users

- Pueden tener asociada, o no, una contraseña.
- Pueden tener asignada una o dos AK (access keys).

¿Access Keys?

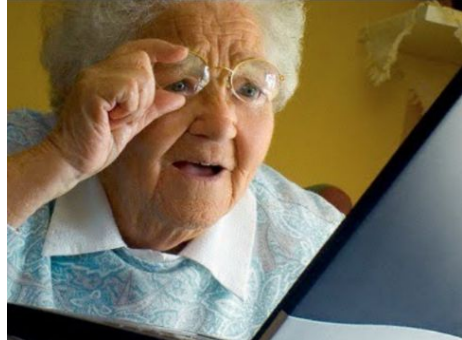
Es una API KEY asociada al usuario que podría usar en un script para ejecutar acciones con los permisos que dicho usuario tenga.

Las access keys se pueden borrar y volver a crear de forma manual.

- ¿Y si me equivoco y las pusheo a GitHub?
- F



IAM Users, todo lo que está mal.



- **¿Mal usar usuarios? Use usuarios toda mi vida!**

En realidad, el título es un bait para que sigas leyendo, lo único que está mal son: las contraseñas y las access keys.

- **Eso es el 98% de un usuario!**

Lo que está mal en realidad, es que tanto las contraseñas como las access keys son credenciales “long-term”, no el usuario en sí.

<Acá es donde no tengo que olvidarme de comentar los riesgos>

- ***Usuarios que descuidan sus credenciales.***



IAM Users, todo lo que está mal.



- **Pregunta rápida, ¿los usuarios pueden tener permisos?**

Sí, los usuarios pueden tener asignados permisos directamente.

Pero... por lo que hablamos antes, los métodos de autenticación y autorización tienen algunas desventajas, quizás hay algo mejor.

IAM Identities

Users: Un usuario con capacidad de autenticarse y hacer uso de recursos mediante credenciales.

Las credenciales pueden ser:

- Usuario y Contraseña
- Access Keys (Llaves programáticas)

Grupos: Colección de usuarios con requerimientos en común.

Roles: Un rol tiene vinculado un conjunto de políticas.

- Los roles pueden ser asumidos por entidades autorizadas (Servicios / Usuarios).

Policy

Define permisos que pueden ser asignados por un administrador a una entidad.



Los roles tienen su gracia

Los roles no se “asignan a usuarios o a grupos”.

Los roles **necesitan ser asumidos** por otra entidad para brindar los permisos que estén asociados al rol:

- Usuarios / Grupos
- Servicios o recursos

“Asumir” un rol, suena extraño...



AWS Security Token Service (STS)

AWS Security Token Service

Es un servicio de AWS que tiene una API que permite solicitar una credencial temporal.

¿Para quién?

- Para un usuario IAM o un usuario de identidad federada.
- Para un recurso



AWS Security Token Service (STS)

Ventajas

- Credenciales temporales (minutos / horas).
- Pueden renovarse todas las veces que se solicite, siempre y cuando el solicitante mantenga los permisos para hacerlo.



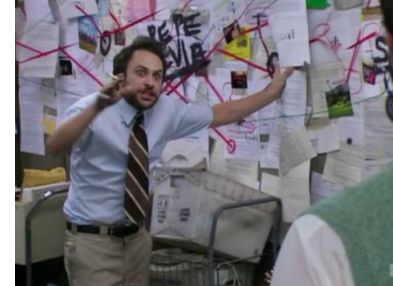
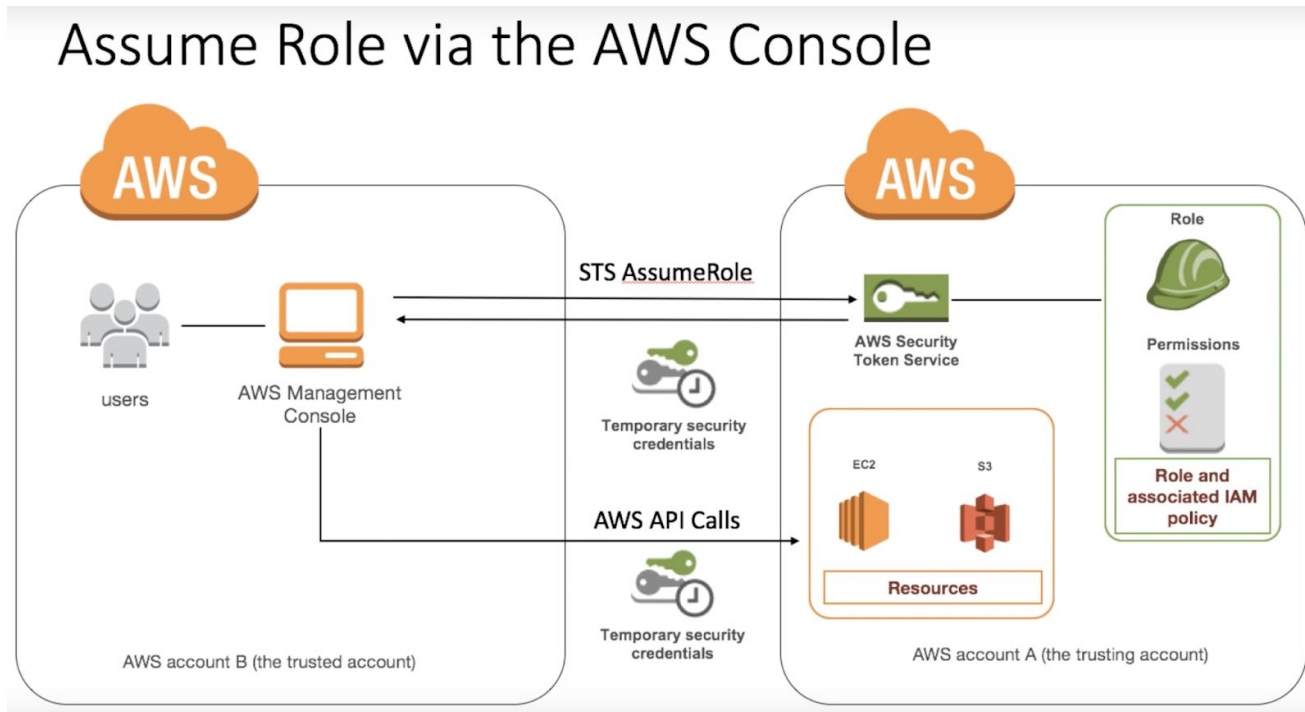
Repaso rápido



- **Users**
 - Son locales a cada cuenta.
 - Permiten acceder a recursos a los que tiene permisos, directa o indirectamente.
 - Credenciales de largo plazo.
- **Users de identidad federada (se coló... ahora quiero saber!)**
 - Involucra muchos componentes, pero lo que importa ahora es: Permiten tomar una fuente de identidad externa y el mecanismo por el que accede a recursos es vía un servicio como STS.
- **STS**
 - API de credenciales de corto plazo

Pongamos todo junto

Assume Role via the AWS Console



AWS Security Token Service (STS)



<Acá es donde no tengo que olvidarme de comentar como mejora los riesgos>

- *Veamos un ejemplo: Millon Dolar Bug*



(2015) - Instagram's Million Dollar Bug

Facebook Widens 'Bug Bounty' Program to Combat Internal Breaches

By Jordan Robertson

26 de julio de 2012, 01:01 ART

Facebook Inc.'s computer-security team faced a quandary after getting an unexpected tip in May.

An outside researcher unearthed a weakness in the company's network that left internal communications vulnerable to eavesdropping. Facebook engineers quickly fixed the bug.

Then came a bigger dilemma: whether to reward the tipster, who by using information for good is known in hacking circles as a "white hat." The team made an unheard-of choice. Facebook would offer a bounty to anyone who finds a hole in its corporate network and then opts to report -- and not exploit -- it.

"If there's a million-dollar bug, we will pay it out," said Ryan McGeehan, who manages Facebook's security-incident response unit.

Facebook is becoming the first big technology provider to reward

(2015) - Instagram's Million Dollar Bug

Según se explica en <http://www.exfiltrated.com/research-Instagram-RCE.php> alguien encontró un bug de estos en Instagram

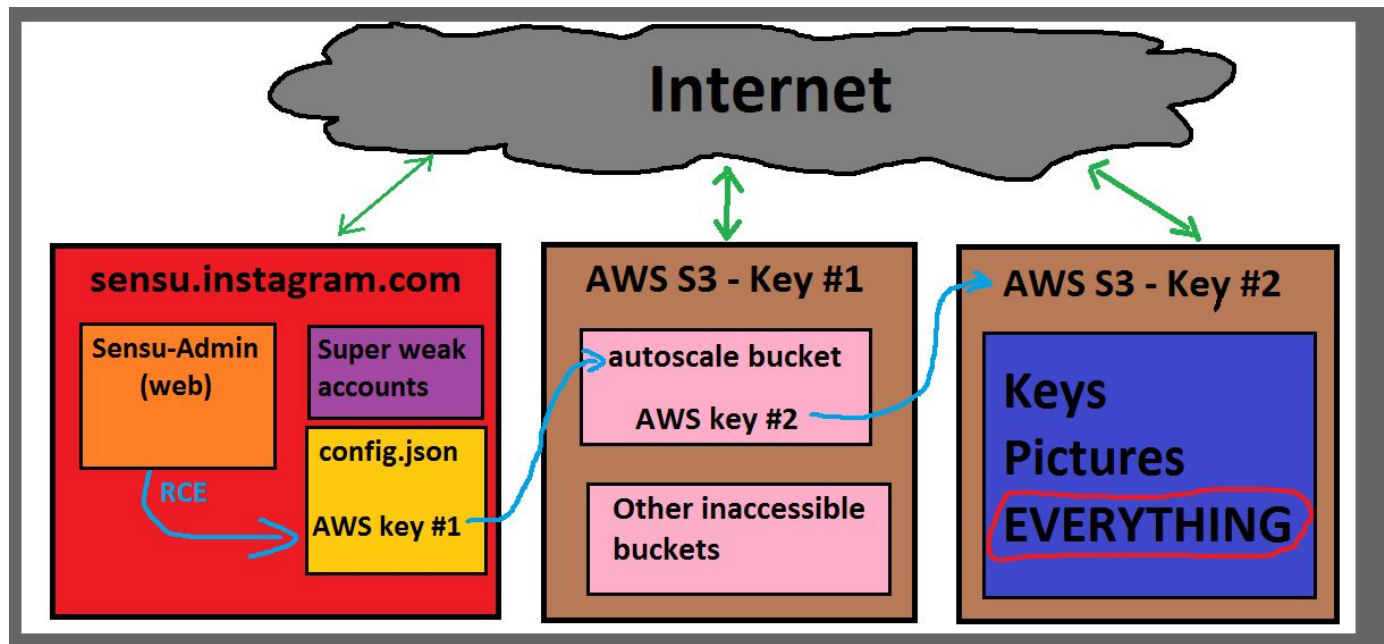
Todo empezó como siempre:

- Un poco de reconocimiento
- Una app usando un token hardcodeado derivó en un RCE
- Crack de contraseñas...

extremely slow when it comes to password cracking. Having come this far, however, I loaded all the passwords into JtR, and kicked it off. To my surprise, passwords immediately came back. In only a few minutes of password cracking, I had recovered 12 passwords! These passwords were all extremely weak, which is why I was able to crack them despite them being bcrypt encrypted:

- Six of the passwords cracked were "changeme"
- Three were the same as the user's name
- Two were "password"
- One was "instagram"

(2015) - Instagram's Million Dollar Bug



(2015) - Instagram's Million Dollar Bug

No le pagaron el millón, es que encontró pocas cosas...

- Static content for Instagram.com websites. Write access was not tested, but seemed likely.
- Source code for fairly recent versions of the Instagram server backend, covering all API endpoints, some image processing libraries, etc.
- SSL certificates and private keys, including both instagram.com and *.instagram.com
- Secret keys used to sign authentication cookies for Instagram
- OAuth and other Instagram API keys
- Email server credentials
- iOS and Android app signing keys
- iOS Push Notifications keys
- Twitter API keys
- Facebook API keys
- Flickr API keys
- Tumblr API keys
- Foursquare API keys
- Recaptcha key-pair

<http://www.exfiltrated.com/research-Instagram-RCE.php>
https://summitroute.com/blog/2015/12/24/instagram_bounty_case_study_for_defense/





Ahora sí,
sigamos con SSRF...

Pero antes, nos llama
otro concepto...



Instance Metadata

Es un feature muy común que pertenece a instancias de la mayoría de los proveedores cloud (privados o públicos).

El uso más trivial:

- Brinda información acerca de la instancia que el administrador puede usar para configurar o administrar la instancia en ejecución.

Ejemplo de información provista: hostname, eventos, security groups, instance profile...

¿Quedó claro que los roles pueden ser asumidos por recursos?

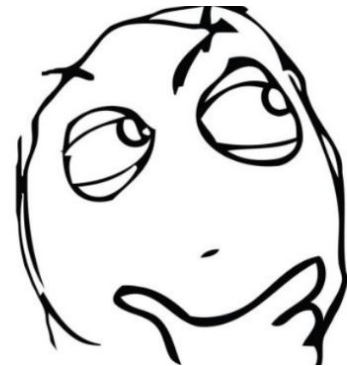
La metadata también permite a la instancia conocer las credenciales temporales STS cada vez la solicite.



¿Y por qué una instancia tendría un rol?

Hay casos de uso simples y otros mucho más complejos.

- Supongamos que la instancia necesita acceder a archivos en algún almacenamiento.
 - **Opción 1:** Creamos un usuario con permisos de acceso al almacenamiento y dejamos las access keys en el código.
 - **Opción 2:** Le asignamos a la instancia un rol con permisos para acceder al recurso.
 - **Hay más opciones, pero hasta acá ¿qué pasaría ante un compromiso de la instancia en ambos casos?**




Accediendo a la metadata

¿Cómo se accede a la metadata?

Está expuesta mediante IPv4 en 169.254.169.254 y en IPv6 mediante fd00:ec2::254.

¿Quién puede acceder a la metadata?

Solo se puede acceder desde la instancia y no cuenta con mecanismo de autenticación y tampoco encriptación. Cualquier software corriendo en la instancia puede observar la metadata. Por eso no se debe almacenar información sensible ahí.



Hay 2 versiones

IMDSv1

- Request / Response method

IMDSv2 (fuertemente recomendada)

- Session-oriented method



¿Cómo se ve la metadata v1?

```
$ curl -s "http://169.254.169.254/latest/meta-data/"  
ami-id  
ami-launch-index  
ami-manifest-path  
auth-identity-credentials/  
block-device-mapping/  
events/  
hostname  
identity-credentials/  
instance-action  
instance-id  
instance-life-cycle  
instance-type  
local-hostname  
local-ipv4  
mac  
managed-ssh-keys/  
metrics/  
network/  
placement/  
profile  
public-hostname  
public-ipv4  
public-keys/  
reservation-id  
security-groups  
services/  
system
```

¿Cómo se ve la metadata v2?



```
$ curl "http://169.254.169.254/latest/meta-data" -I
HTTP/1.0 401 Unauthorized
Content-Length: 343
Content-Type: text/html
Date: Wed, 19 Oct 2022 21:19:44 GMT
Connection: close
Server: EC2ws
```



```
$ TOKEN=
`curl -s -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 3600"`

$ echo $TOKEN
AQAAADBhvl9w4Vu7sIwb3Sa15ftVgVAKpg-THJPtyoplUjkb6RQ7tQ==
```



```
$ curl -H "X-aws-ec2-metadata-token: $TOKEN" "http://169.254.169.254/latest/meta-data/"
ami-id
ami-launch-index
ami-manifest-path
auth-identity-credentials/
block-device-mapping/
events/
hostname
identity-credentials/
instance-action
instance-id
instance-life-cycle
instance-type
local-hostname
local-ipv4
mac
managed-ssh-keys/
metrics/
network/
placement/
profile
public-hostname
public-ipv4
public-keys/
reservation-id
security-groups
services/
system
```

Está la meta-data y la user-data...

La metadata puede contener user-data, básicamente es una sección que **puede** incluirse durante el ciclo de vida de la instancia y podría por ejemplo tener un script de inicialización de la instancia.

```
$ curl -s "http://169.254.169.254/latest/meta-data/"  
ami-id  
ami-launch-index  
ami-manifest-path  
auth-identity-credentials/  
block-device-mapping/  
events/  
hostname  
identity-credentials/  
instance-action  
instance-id  
instance-life-cycle  
instance-type  
local-hostname  
local-ipv4  
mac  
managed-ssh-keys/  
metrics/  
network/  
placement/  
profile  
public-hostname  
public-ipv4
```

<http://169.254.169.254/latest/user-data/>

Algunas historias

(2014) Prezi

Nicolas Grégoire descubrió que Prezi permitía referenciar en un slide, información del servidor.

1. Reportó un IDOR (Insecure Direct Object Reference)

- **Modify <url> to point to a file you control**
- **The web editor will load the remote resource**
- **But everything is done client-side **FAIL!****



```
...  
<object>  
  <source>  
    666031337  
    <url>file:///etc/passwd</url>  
  </source>  
  <sourceUrl>blabla.swf</sourceUrl>  
</object>  
...
```

→ El fix fue usar una whitelist para permitir solamente referencias http://

Referencias:

https://www.agarri.fr/docs/Easy_hacks_for_complex_apps-INS14.pdf

<https://www.youtube.com/watch?v=mQjTgDuLsp4>

<https://engineering.prezi.com/prezi-got-pwned-a-tale-of-responsible-disclosure-ccdc71bb6dd1>

(2014) Prezi

Prezi's feedback

*We finished our investigation [...] and we think that with some hacking **this vulnerability can be exploited pretty badly**, e.g. an attacker would be able to gain access to some critical credentials, therefore [...] we would like to reward you with a **\$2000 bounty**.*

If the URL can't start with file:// ...

... abuse http://



Analizando el fix, explotó un SSRF para acceder a la metadata

Ahí podemos encontrar credenciales temporales que STS le da a la instancia en caso que la instancia tenga un rol asociado.

En particular, descubrió que en la user-data había información sensible

Referencias:

https://www.agarri.fr/docs/Easy_hacks_for_complex_apps-INS14.pdf

<https://www.youtube.com/watch?v=mQjTgDuLsp4>

<https://engineering.prezi.com/prezi-got-pwned-a-tale-of-responsible-disclosure-ccdc71bb6dd1>

(2014) Prezi

<http://169.254.169.254/latest/user-data/>

/etc/chef/client.rb

```
chef_server_url "https://api.opscode.com/organizations/prezi"  
validation_client_name "prezi-validator"
```

etc/chef/validation.pem

-----BEGIN RSA **PRIVATE** KEY-----

MIIEpQIBAAKCAQEA09U/TBxe[...]iRLSo6sJTJm6RCk6qZqRxM7UCbBw=

-----END RSA PRIVATE KEY-----

/etc/chef/encrypted_data_bag_secret

gqrnkG+M/t/1/3KhCzRNEiMBL[...]lohHq2lil/P8fS21aZJkXYmHyKdMJ2qo=

Bash script (150+ lines)

– Creates critical files

- **/etc/chef/client.rb**
- **/etc/chef/validation.pem**
- **/etc/chef/encrypted_data_bag_secret**

(2014) Prezi

Prezi's feedback

*[...] this exploitation has the **same root cause** as your previous local file access, **however the attack path is different** and [...] your submission gave some nice ideas where to improve ourselves, therefore we would like to offer you **\$2000** for this issue as well. Congratz! :)*

Prezi's actions

- Add a black-list
 - Private IP addresses are forbidden (using IPy)
 - Impedance mismatch? Yes, using octal format!
 - Bypass: 0251.0376.0251.0376 **WIN! \$500**
- Detect and manage HTTP redirects
 - Black-list applied to the final destination
- Chef secrets moved to the AMI itself
 - Referenced from the user-data script
 - Readable only by root
- Renewal of every Chef key
 - Wasn't an easy step



FIN