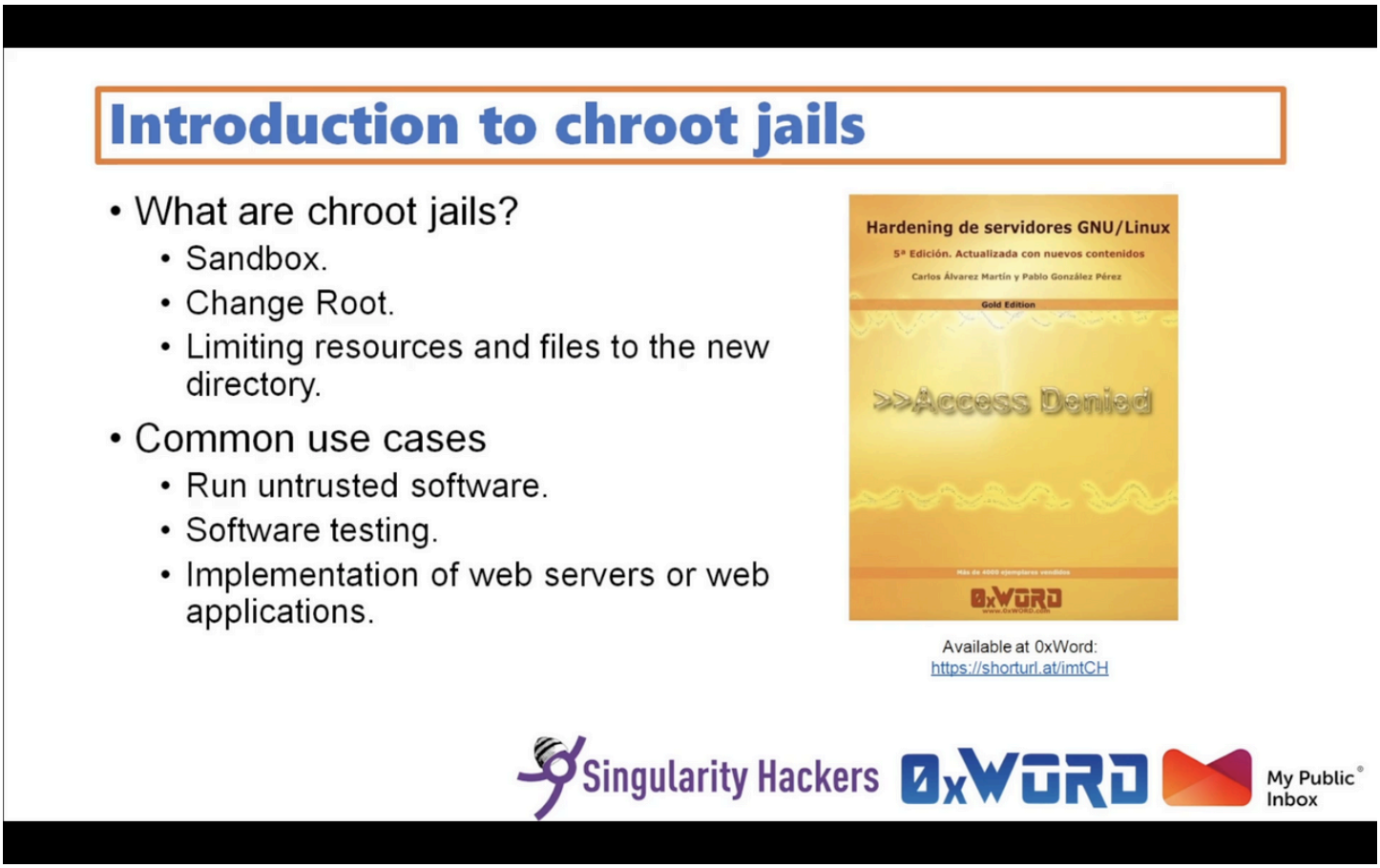
Chroot

En esta sesión vamos a tratar el tema de jaulas Chroot. Los puntos que vamos a ver a lo largo de esta presentación serán los siguientes.

Veremos una breve introducción a qué son las jaulas Chroot, cómo es su funcionamiento y cómo operan.

Realizaremos una prueba de concepto práctica para configurar una de estas jaulas y poder comprobar su correcto funcionamiento y evidenciar que el aislamiento del sistema funciona.

Y finalmente terminaremos con una serie de conclusiones a modo de resumen.



Comenzamos viendo qué son las jaulas Chroot.

Como concepto previo. Para entender aún mejor y de forma más sencilla el concepto de jaula Chroot, vamos a comentar brevemente el concepto de sandbox o cajas de arena que son muy utilizadas a día de hoy ya que nos van a permitir realizar el aislamiento de procesos o aplicaciones en diferentes sistemas. Entonces podemos ver las jaulas Chroot como una técnica similar.

Al final estas jaulas son una técnica de aislamiento que son muy utilizadas en sistemas GNU/Linux y nos van a permitir poder ejecutar un proceso dentro de un entorno virtual que podemos decir que está limitado. Y aquí es donde entra en juego esa analogía con el concepto

de enjaulamiento o de jaula, donde se va a limitar la capacidad de actuación a ese entorno nuevo en el que nos vamos a encontrar.

Comentar que Chroot es la abreviación del término chainroot.

Esto es así porque al final lo que vamos a hacer realmente es modificar la raíz del sistema de archivos de un proceso de forma que apunte a un nuevo directorio específico indicado por nosotros a la hora de configurarlo.

Con esto lo que vamos a hacer es limitar a ese proceso para que únicamente pueda interactuar con los ficheros, con las herramientas, con las aplicaciones del sistema que vamos a definir en ese directorio y va a quedar totalmente aislado del resto del sistema, no va a poder interactuar con nada más salvo con lo que nosotros configuremos dentro de ese nuevo directorio.

Entre los casos de uso más comunes para este tipo de técnicas como son las jaulas chroot, pues nos encontramos por ejemplo el poder ejecutar software no confiable, por ejemplo son fuentes cargados de Internet de fuentes desconocidas, algún tipo de software que hemos capturado mediante evidencias en otro tipo de proceso de seguridad y podemos considerar que tiene malware y quisiéramos poder verificar el comportamiento. Son muy utilizadas también en procesos de testeo de software, ya sea de código nuevo, de actualizaciones, pero también en el uso de nuevas librerías e incluso para probar dependencias de software o diferentes configuraciones en base al entorno en el que se va a ejecutar, son una herramienta muy útil ya que al estar totalmente aislado podemos probar todas estas configuraciones de forma segura.

Uno de los casos de uso quizá más importantes sería la implementación de servidores web o de aplicaciones web.

Evidentemente al poder ejecutarse de forma aislada y controlada vamos a obtener una mayor seguridad y garantía de este tipo de entornos, que suelen ser entornos críticos expuestos en producción.

Algunos de los beneficios de utilizar esta técnica ya los hemos comentado, por resumirlo brevemente, vamos a tener una mayor capacidad de seguridad, vamos a proteger el sistema, vamos a reducir los riesgos, vamos a evitar posibles daños mayores, conseguimos también una protección contra intrusiones y malware, esto va a posibilitar dificultar una posible escalada de privilegios y la haces otro sistema en el caso de que un atacante llegase a realizar una explotación.

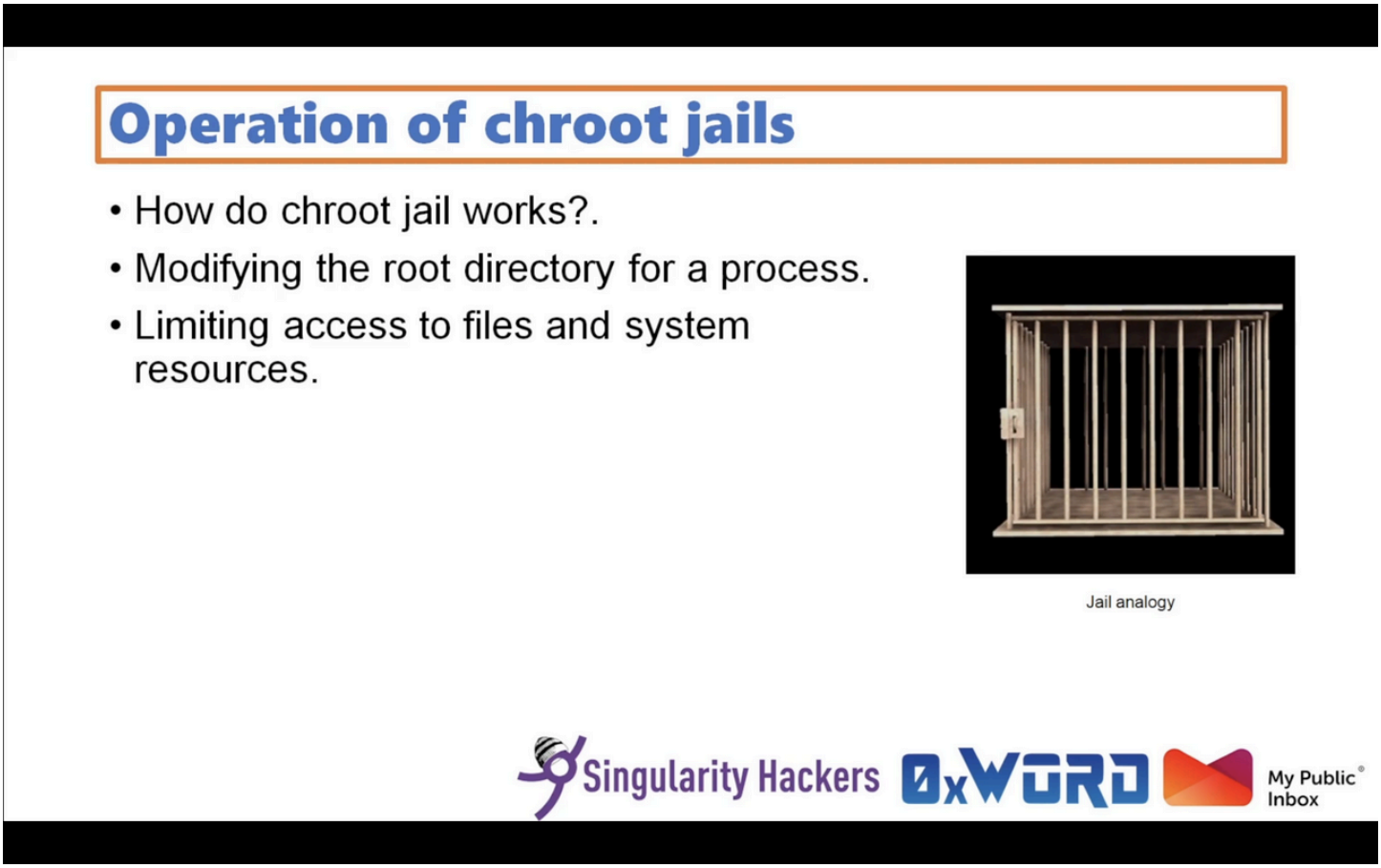
Como hemos dicho, va a facilitar la creación de entornos de prueba, vamos a poder probar software tanto nuevo como posibles actualizaciones sin afectar al resto del sistema, vamos a poder ejecutar software con diferentes dependencias, etc.

Es importante también determinar que este tipo de técnicas pues también tiene limitaciones. Las jaulas Chroot no es una solución mágica, no es una solución de seguridad infalible.

Si un usuario consigue acceder a la jaula consiguiendo modificar esos permisos o incluso instalar software vulnerable, este software vulnerable ejecutado dentro de la jaula podría llegar a producir una explotación del sistema y conceder acceso al posible atacante.

También hay que tener en cuenta que esta técnica no nos va a proteger contra otro tipo de ataques, como por ejemplo podrían ser ataques de tipo kernel. Si el kernel de nuestro sistema es vulnerable y un atacante lo sabe, podría llegar a realizar una explotación.

Entonces algunas consideraciones básicas que debemos de tener en cuenta a la hora de utilizar este tipo de técnica es que no debe de ser nunca, bajo ningún concepto la única solución de seguridad empleada, siempre debe de ir de la mano con otro tipo de medidas para así conseguir en su conjunto una seguridad efectiva del sistema. Hablamos por ejemplo de una correcta configuración de usuarios y de permisos, actualizaciones de seguridad, etc.



Pasamos a ver el funcionamiento de esta técnica, cómo trabajan las jaulas chroot,

Lo Primero de todo, como ya hemos comentado, el concepto de enjaulamiento va a consistir en modificar el directorio raíz de un proceso.

Cuando hablamos del directorio raíz o root, realmente lo que hacemos es referirnos al punto de partida en el que un proceso, usuario o aplicación va a acceder a todos los demás archivos y recursos del sistema.

Cuando realizamos ese cambio de directorio raíz lo que vamos a hacer es limitar a un proceso concreto a que sólo puede interactuar con el nuevo directorio que nosotros le vamos a indicar

mediante la configuración de la jaula.

Por lo tanto va a quedar aislado del sistema y únicamente tendrá visible y disponible los archivos y recursos dentro de ese directorio, que como veremos posteriormente es una de las tareas que vamos a realizar a la hora de configurar la jaula.

¿Y todo esto cómo se realiza?

Se realiza mediante la llamada sistema chroot.

Normalmente este comando va a venir preinstalado en la mayoría de distribuciones de Linux que vamos a utilizar, como veremos posteriormente, en nuestro caso vamos a utilizar una máquina virtual Ubuntu donde ya venía preinstalado de serie.

Como parámetro principal de este comando va a ser el nuevo directorio tiene diferentes usos.

Os animo desde aquí a comprobar la documentación oficial de esta llamada al sistema para ver todos los posibles parámetros y opciones que permite y posteriormente veremos un caso de uso concreto.

Para poder garantizar y entender que la jaula está funcionando correctamente debemos de entender realmente qué es lo que va a realizar o cómo va a poner ese límite la jaula.

El proceso dentro de la jaula no va a poder acceder a ningún archivo o aplicación fuera de la jaula, incluyendo aplicaciones del sistema, archivos del sistema, bibliotecas, cualquier otro tipo de aplicación y cualquier otro tipo de recurso.

Esto nos va a ser muy útil a la hora de certificar que la jaula está bien configurada, porque comenzaremos a ver una serie de errores en el momento en el que intentemos acceder a recursos fuera de la jaula.

PoC chroot jails

- Creation of the jail directory.
- Copy of the necessary files.
- Evidence.

Ubuntu. Source: Wikipedia

```
root@aromero:/var/test# sudo chroot --userspec=1001:1001 /var/test/
bash-5.1$ whoami
whoami: cannot find name for user ID 1001: No such file or directory
```

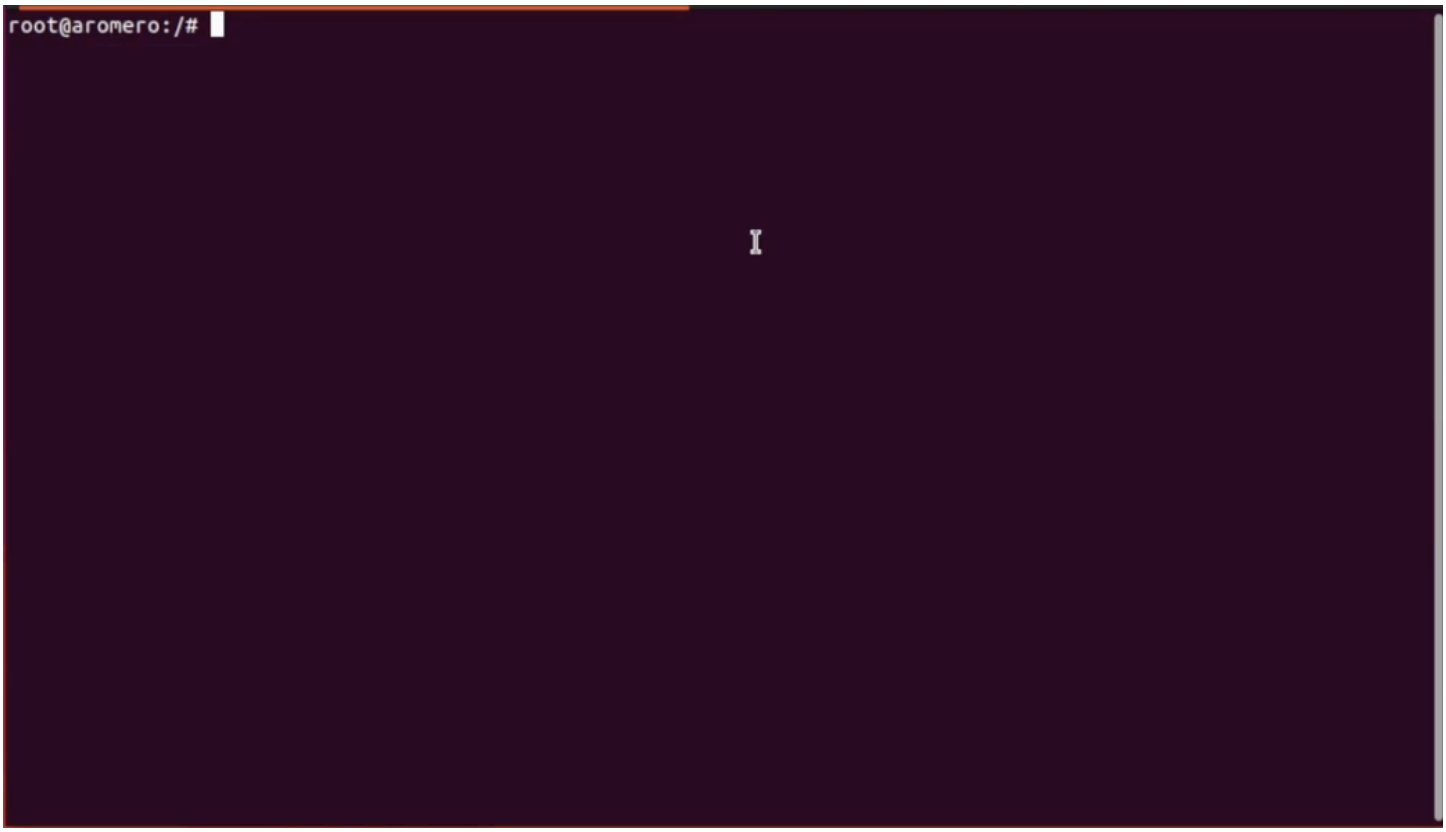
Chroot jail whoami evidence.

Sin más, vamos a pasar a realizar esta prueba de concepto de forma práctica en una máquina virtual Ubuntu.

Vamos a ver cómo se crea el nuevo directorio de la jaula, vamos a ver cómo determinar según el proceso que queramos permitir ejecutar dentro de la jaula, cómo conocer las posibles dependencias que tiene para que esto funcione y vamos a evidenciar la correcta configuración de la jaula.

Pues pasamos a la máquina virtual donde vamos a realizar la prueba de concepto.

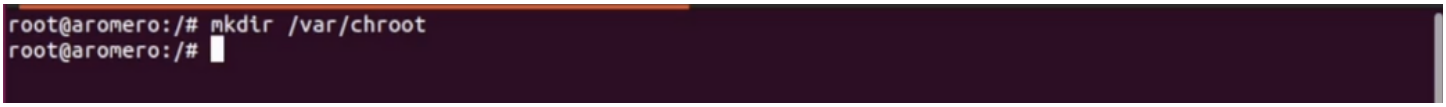
En este caso tenemos una máquina virtual con un sistema GNU Linux Ubuntu instalado y tenemos ya preparada una terminal, una consola donde tenemos la cuenta habilitada de root.



Ya sabemos que por seguridad y por una recomendación de uso correcta, esta cuenta no se debe utilizar bajo ningún concepto, exceptuando casos de administración como realmente puede ser este momento.

Entonces vamos a comenzar sin más la configuración de una jaula de chroot.

Lo primero vamos a crear un nuevo directorio, en realidad se podría montar en cualquier punto del sistema, pero bueno, nosotros hemos decidido para este ejercicio realizarlo en /var y vamos a crear un directorio llamado chroot, con el comando mkdir /var/chroot.



Para este ejercicio también vamos a contar con un usuario que previamente hemos creado y que se llama jailuser.

Para este usuario (jailuser) a lo largo de la prueba de concepto vamos a necesitar conocer su identificador de usuario y su identificador de grupo.

Abrimos una nueva pestaña de la consola y para conocer estos datos vamos a ejecutar este comando (sabiendo que estamos en el equipo de /home/aromero):

/cat /etc/passwd | grep jailuser

Vamos a acceder al fichero passwd donde están los usuarios con sus contraseñas y vamos a realizar un filtrado por por temas de seguridad para acceder solamente al registro de este usuario que ya hemos creado.

En este caso pues aquí tenemos los datos del usuario.

```
root@aromero:/home/aromero# cat /etc/passwd | grep jailuser
jailuser:x:1002:1002::/home/jailuser:/bin/sh
root@aromero:/home/aromero#
```

Y realmente 1002:1002 son los datos que nos interesan, el UID y el GUID.

Bueno, esto lo utilizaremos en pasos posteriores.

Para continuar con nuestro ejercicio lo que vamos a hacer es nos vamos a posicionar en el nuevo directorio que hemos creado, ahora explicaremos por qué hacemos este paso.

```
root@aromero:/# cd /var/chroot/
root@aromero:/var/chroot# ls
root@aromero:/var/chroot#
```

Ya estamos aquí, vemos que el directorio está vacío. Nos hemos posicionado en el directorio que acabamos de crear porque cuando vamos a ejecutar ahora el comando chroot le vamos a indicar ./

Cuando ejecute este comando lo que estamos haciendo es hacer la llamada al sistema y con ./ lo que estamos diciendo es que cambie la actual raíz del sistema del directorio del proceso activo a el punto en el que nos encontramos, que en este caso va a ser /var/chroot.

Lo ejecutamos.

```
root@aromero:/var/chroot# chroot ./
chroot: failed to run command '/bin/bash': No such file or directory
root@aromero:/var/chroot#
```

Vemos que tenemos un error en pantalla, esto no quiere decir que el ejercicio lo estemos realizando mal, ni mucho menos, esto lo que quiere decir es que al lanzar el comando chroot se debería de levantar una terminal.

Como dentro del directorio que hemos visto está vacío, no tenemos nada, pues nos devuelve este error porque realmente está intentando ya lanzar la consola desde el nuevo directorio en el que nos encontramos.

Entonces, para poder configurar correctamente la jaula y poder lanzar la consola, vamos a copiar el fichero pass dentro de nuestro nuevo directorio.

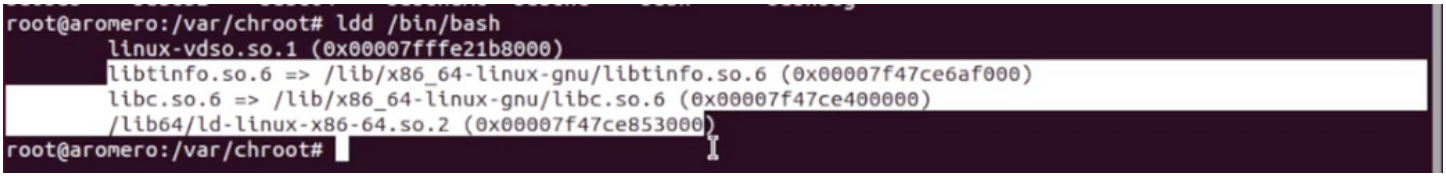
Para ello no basta con solo copiarnos el fichero /bin/bash dónde está el proceso, sino que debemos conocer las dependencias adicionales que tiene.

Para ello vamos a utilizar el comando ldd y le vamos a indicar el directorio sobre el que queremos comprobar las dependencias, en este caso /bin/bash.

```
root@aromero:/var/chroot# ldd /bin/bash
        linux-vdso.so.1 (0x00007fffe21b8000)
        libtinfo.so.6 => /lib/x86_64-linux-gnu/libtinfo.so.6 (0x00007f47ce6af000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f47ce400000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f47ce853000)
root@aromero:/var/chroot#
```

El resultado del comando son las dependencias del proceso en cuestión.

A nosotros lo que nos va a interesar para nuestro ejercicio son estas tres líneas de aquí, que es donde realmente están las dependencias que vamos a necesitar llevarnos a nuestro nuevo directorio para que la consola funcione.



Entonces, el siguiente paso va a ser crear el árbol de directorio completo dentro de nuestro nuevo directorio chroot para poder alojar todas estas dependencias y poder alojar el proceso bash.

Entonces comenzaríamos creando con mkdir una carpeta llamada bin donde vamos a copiar posteriormente el proceso bash.

Vamos a crear también una ruta para poder alojar estas dos dependencias de aquí arriba.

Y vamos a crear la ruta lib64 para la tercera dependencia.

Perfecto, hacemos un ls.

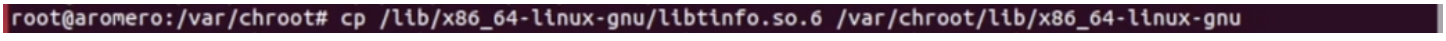
Ya tenemos los tres directorios creados bin, lib y lib64, uno para el proceso bash y las otras dos para el resto de dependencias, listos para poner las dependencias dentro.

Entonces ahora tendríamos que comenzar a copiar uno a uno tanto el proceso bash como las tres dependencias en sus respectivas rutas.

Entonces comenzamos copiando bash dentro de /var/chroot/bin.

Perfecto, ya tendríamos la primera.

Vale, ahora vamos a hacer la copia de la dependencia libtinfo.so.6 (qué está en la ruta x86_64-linux-gnu) en /var/chroot/lib/x86_64-linux-gnu.



Bueno, ahora vamos a hacer el mismo proceso pero para la dependencia libc.so.6 que también se encontraba dentro de la misma ruta /var/chroot/lib/x86_64-linux-gnu.

```
root@aromero:/var/chroot# cp /lib/x86_64-linux-gnu/libc.so.6 /var/chroot/lib/x86_64-linux-gnu
```

Y finalmente vamos a copiar la dependencia ld-linux x86-64.so.2, que ésta se encontraba dentro de la ruta lib64, y la copiamos en /var/chroot/lib64.

```
root@aromero:/var/chroot# cp /lib64/ld-linux-x86-64.so.2 /var/chroot/lib64
```

Comentar también que en este caso, al estar trabajando bajo una máquina con un sistema Ubuntu, pues tenemos estas dependencias. Podrían llegar a variar en función del tipo de sistema operativo GNU Linux que en nuestro caso estemos utilizando para realizar el ejercicio. El resultado siempre tiene que ser el mismo.

Bueno, pues ya tenemos nuestro árbol de directorios con todos los ficheros necesarios para levantar una terminal dentro de nuestra jaula.

Entonces vamos a ejecutar el comando: chroot --userspec=1002:1002 /var/chroot. En esta ocasión le decimos que utilice como usuario el user jailuser que hemos creado previamente y que antes veíamos cómo obtener sus identificadores de usuario y de grupo y que vimos que era el 1002 y le vamos a decir que el nuevo directorio va a estar en /var/chroot.

```
root@aromero:/var/chroot# chroot --userspec=1002:1002 /var/chroot/
```

Ahora sí se ha levantado una consola de tipo bash al ejecutar este comando chroot, lo que significa que todo el proceso que hemos hecho anteriormente de copiar tanto el propio proceso bash como todos los ficheros que son dependencias para su correcto funcionamiento, pues lo hemos realizado correctamente. Me sale bash-5$ cómo prompt ahora, la terminal de bash parece funcionar, pero ¿Está aislada?.

Entonces, bueno, ¿Cómo podemos comprobar y asegurarnos que la jaula está correctamente configurada?

Por ejemplo, podríamos intentar lanzar el comando whoami para que nos diga qué usuario somos y como vemos por pantalla esto devuelve un error.

```
bash-5.1$ whoami
bash: whoami: command not found
bash-5.1$
```

Esto lo que significa es que el comando whoami dentro de Linux no se encuentra dentro de nuestro directorio y la jaula nos está aislando realmente y no podemos acceder a una herramienta que está instalada en cualquier distribución.

Esto significa que vamos por buen camino, que podemos confiar que la jaula está correctamente configurada.

Pero bueno, vamos a intentar certificarlo un poquito más y vamos a hacer una cosa, vamos a ver las dependencias de whoami, vamos a copiar el contenido necesario dentro de nuestro directorio chroot y vamos a volver a ejecutarlo para ver qué nos devuelve.

Entonces, lo primero de todo, salimos de la consola que estaba levantada con el comando exit, volvemos a nuestra cuenta de root y vamos a ejecutar de nuevo el comando ldd, pero en este caso lo vamos a hacer con /urs/bin/whoami.

Ahí tenemos las dependencias que tiene.

```
root@aromero:/var/chroot# ldd /usr/bin/whoami
        linux-vdso.so.1 (0x00007fff0154c000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fc6f4200000)
        /lib64/ld-linux-x86-64.so.2 (0x00007fc6f44ff000)
root@aromero:/var/chroot#
```

En este caso nos interesan estas dos.

```
root@aromero:/var/chroot# ldd /usr/bin/whoami
        linux-vdso.so.1 (0x00007fff0154c000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fc6f4200000)
        /lib64/ld-linux-x86-64.so.2 (0x00007fc6f44ff000)
root@aromero:/var/chroot#
```

Y si las comparamos con las que nos ha devuelto el comando ldd para /bin/bash, vamos a ver que ya las tenemos integradas.

Entonces, en este caso lo único que vamos a necesitar hacer es crearnos un directorio dentro de chroot que sea usr/bin, con el comando mkdir -p usr/bin, nos aseguramos que está correctamente creado con ls, ahí vemos el directorio usr.

```
root@aromero:/var/chroot# mkdir -p usr/bin
root@aromero:/var/chroot# ls
bin  lib  lib64  usr
```

Ahora simplemente tendríamos que hacer una copia de usr/bin/whoami y así nos lo vamos a llevar a /var/chroot/usr/bin que acabamos de crear.

```
root@aromero:/var/chroot# cp /usr/bin/whoami /var/chroot/usr/bin/
```

Si ahora volvemos a ejecutar de nuevo el comando chroot con el mismo usuario jailuser e indicándole que el nuevo directorio raíz va a ser var chroot, vemos que se levanta de nuevo la consola porque el prompt es bash-5.

```
root@aromero:/var/chroot# chroot --userspec=1002:1002 /var/chroot/
```

Y si ahora tecleamos el comando whoami vemos que el comando funciona pero quenos devuelve otro error.

```
bash-5.1$ whoami
whoami: cannot find name for user ID 1002: No such file or directory
```

Este error lo que nos viene a decir es que el usuario con el id 1002, que sabemos que es jailuser, no funciona, o no se encuentra mejor dicho.

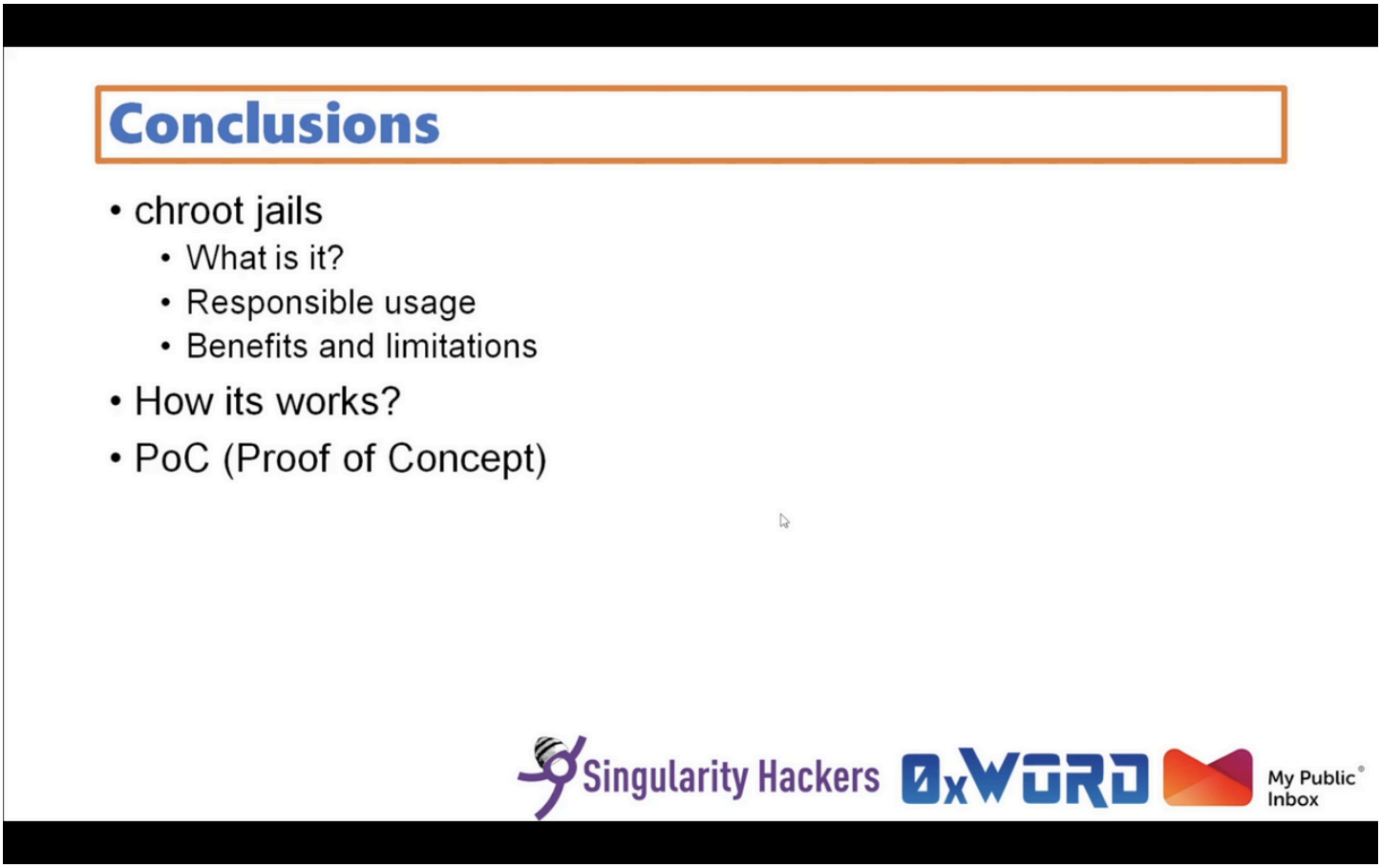
Dicho esto lo que quiere decir es que realmente estamos operando dentro de una jaula chroot, donde como hemos visto pues la consola se ha levantado correctamente porque hemos configurado el proceso, el comando whoami se puede lanzar porque lo hemos configurado,

pero vemos que el resto de llamadas no funciona, por ejemplo el comando ls tampoco se encuentra.

```
bash-5.1$ ls
bash: ls: command not found
```

Con esto podemos garantizar que la jaula está correctamente configurada y que nos está limitando el acceso al resto del sistema. El siguiente paso sería comenzar a configurarlo para el proceso que realmente queramos ejecutar, trayendo el resto de archivos de llamadas al sistema que fuesen necesarios.

Con esto finalizaríamos la prueba de concepto y pasaríamos al resumen y las conclusiones de esta presentación.



Hemos visto qué son las jaulas chroot.

Hemos visto que son una potente herramienta de seguridad que nos proporciona mayor flexibilidad y mayor eficiencia a la hora de administrar sistemas GNU Linux.

Hemos visto como requieren de un uso responsable, no se pueden adoptar como una única medida de seguridad, debemos de tener en cuenta que no es una solución infalible y que no nos protegerán frente a otro tipo de ataques, pero que utilizadas de forma adecuada y en combinación con otras medidas de seguridad podemos asegurar una buena protección de nuestro sistema.

Hemos visto cómo funciona el comando chroot, hemos realizado una prueba de concepto donde además hemos podido certificar que la configuración que hemos realizado de la jaula era

correcta y que realmente estábamos en un sistema totalmente aislado.