Trabajo Práctico N°1

Seguridad Informática

Alumno: Navall, Nicolás Uriel. N-1159/2.

7) Un repositorio privado en github solo puede ser visto por personas autorizadas por el creador de dicho repositorio. Luego una vez obtenido el acceso para ver el repositorio uno puede crear un pull request para realizar cambios en el código. Si alguien no posee acceso para ver el repositorio (confidencialidad) entonces no podrá crear una pull request en este para editar el código (integridad).

11) a) *Disclosure*, *disruption*, *deception* y *usurpation* son formas en las que se puede categorizar una amenaza, a continuación se hablará en detalle de cada una de ellas.

Se llama revelación (disclosure) al acceso no autorizado de información. Esto incluye robar información, poner información a disposición de terceros inadecuadamente o interceptar el flujo de datos.

Se llama alteración (disruption) a la interrupción o prevención de una operación correcta. Esto puede incluir cambiar maliciosamente la lógica de un programa, un error humano que desactive un sistema, un corte de electricidad o una falla en el sistema por un bug. También puede referirse a cualquier obstrucción que dificulte el funcionamiento de un programa. Los famosos ataques DDoS los cuales saturan servidores con llamadas son un ejemplo de esta clase de amenazas.

Se llama engaño (deception) a la aceptación de datos falsos. Esto incluye enmascaramiento (hacerse pasar por una entidad autorizada), inyección de información falsa o modificación de información existente y repudiación (cuando alguien niega falsamente haber recibido u originado información). Por ejemplo, hacerse pasar por otra persona en transacciones bancarias utilizando sus credenciales.

Se llama usurpación (usurpation) al control no autorizado de alguna parte del sistema. Esto incluye robo de servicio o robo de información al igual que cualquier uso indebido del sistema como manipulación o acciones que resulten en una violacion de los privilegios del sistema.

Cabe recalcar que estas clases no son mutuamente exclusivas, y pueden existir amenazas que pertenezcan a más de una clase al mismo tiempo. Por ej, si en el último ejemplo dado en lugar de robar credenciales bancarias se obtuvieron privilegios de administrador en una página estaríamos tomando control sobre una parte del sistema, lo cual implica que es también una amenaza de tipo *usurpation*.

b) Dado que la confidencialidad se encarga de que solo los usuarios autorizados puedan ver los datos, la integridad de que solo los usuarios y mecanismos autorizados puedan modificar los datos y programas, y la disponibilidad que los usuarios autorizados puedan usar los datos y programas cuando lo necesiten entonces podemos afirmar que cada uno de estos se encargan de proteger el sistema contra amenazas de clase disclosure, deception y disruption respectivamente.

En cuanto a ataques de usurpation, como un ataque de esta clase se basa en controlar alguna parte del sistema entonces los conceptos de disponibilidad e integridad deben encargarse de proteger al sistema de amenazas de esta clase, ya que si el atacante hace posesion de datos o entidades indebidamente entonces su disponibilidad se ve afectada, y si también hace uso de esos sistemas (como en el ejemplo dado) entonces la integridad también está en riesgo.

Por lo tanto, asegurar los tres servicios de seguridad vistos en un sistema es suficiente para proteger a dicho sistema de las amenazas vistas.

21) Programa 3)

```
-- Prog 3
x = readH();
writeL("Loading");
while (x-- > 100)
    writeL(".");
writeH(x);
```

Hay flujo indebido ya que estoy leyendo un objeto que tiene clase de acceso alta, pero como se va a guardar en un objeto de clase de acceso baja un carácter “.” por cada valor arriba de 100 que tenga el valor leído anteriormente, puedo reconstruir la información almacenada en el objeto de clase alta leyendo la información en el objeto de clase baja, contando la cantidad de puntos grabados en este y sumándole 100.

22) Programa 3)

El programa en multi-ejecución segura comienza con seguridad de tipo L y como en la primera línea realiza un read en H entonces se realiza un fork del proceso, obteniendo así PH y PL, en donde PH habrá leído el valor dado por la primera instrucción. Luego el proceso PL al leer la segunda instrucción pasará a escribir en L “Loading”, lo cual modificará el entorno, pero cuando PH ejecute esta instrucción no modificara el entorno.

Luego PH pasará a ejecutar el bucle while pero al ejecutar cada instrucción de “writeL(".")” no modifica el entorno ni su memoria, por lo cual puede interpretarse como si la ignorara. En cambio, PL escribirá en L (cambiando el entorno) cada bucle de while que ejecute, pero dado que el valor de x no existe en la memoria de PL entonces el valor que éste use será aleatorio. Es decir, la cantidad de puntos grabados en L no podrán usarse para recuperar el valor de x, evitando así el flujo indebido.

Por último PH pasa a ejecutar la última línea modificando el entorno. PL en cambio no modificara el entorno.

23) Programa 5 (https://ifc-challenge.appspot.com/steps/allergy)

```
try{
  if(h){
      throw;
  }
  l=false;
}
catch skip;

try{
  if(!h){
      throw;
  }
  l=true;
}
catch skip;
```

El ataque es de clase “disclosure”, ya que estoy obteniendo información a la cual no debería tener acceso. En particular, es una de tipo indirecta ya que no se asigna el valor de **h** cualquiera sea a **l** sino que se busca

descubrir el valor de **h** con otras estructuras y asignarlo a **l** una vez descubierto.

La limitación del sistema es que si se aplica un “throw” en un if/loop con guardias altas este es considerado de clase de seguridad alta también, pero no así el resto del bloque try. La otra limitación que tiene es que un bloque catch no puede asignar variables low si su bloque try tiene una instrucción throw de seguridad alta.

Para impedir el ataque se debería incluir la siguiente regla: Cuando en un try exista un throw dentro de un bloque if cuya ejecución depende de una variable high, restringir la asignación de valores a variables low en lo que queda del bloque try.