**WEB GÜVENLİĞİ**

**MUTILLIDAE-OWASP Uygulamasının**

Web güvenliğinde karşımıza çıkan bazı açıkları ellimden geldiğince Mutillidae üzerinden örneklemeye çalışacağım. İşletim sistemi olarak Ubuntu, proxy aracı olarak da Burp Suite kullandım. Hazırlık aşamasını aşağıda kısaca anlatım.

Apache kurulumu:

sudo apt-get update   
sudo apt-get install apache2

PHP kurulumu:  
sudo apt-get install php5 libapache2-mod-php5

MYSQL server kurulumu:  
sudo apt-get install mysql-server

PHPMyadmin kurulumu  
sudo apt-get install libapache2-mod-auth-mysql php5-mysql phpmyadmin

Gerekli izinler alındıktan sonra www klasörüne dosyayı şu adresten indirebiliriz. https://sourceforge.net/projects/mutillidae/

İlk olarak Burp suite çalışabilmesi için OpenJDK nın kurulu olması gerekmekte.

sudo apt-get install openjdk-7-jdk openjdk-7-jre

Daha sonra dosyayı şu linkten indiriyoruz

<https://portswigger.net/burp/download.html>

Downloads klasöründe aşağıdaki komutu çalıştırarak proxy aracı açılır.

sudo java -jar -Xmx2g burpsuite\_free\_v1.6.25.jar

**Broken Authentication And Session Management**

**Nedir ?**

Güvenliği tam olarak alınmamış oturum nesnelerin(session ID) ve çerezlerin(cookie) korunmamasından kaynaklanan bir açıktır. Bu açık farklı yollar ile istismar edilebilir.

Detaylı bilgi için şu adresi ziyaret edebilirsiniz. https://www.owasp.org/index.php/Top\_10\_2013-A2Broken\_Authentication\_and\_Session\_Management

**Uygulama1**

Owasp 2013 -> A6 – Broken Authentication And Session Management -> Authentication Bypass ->Via cookies sayfasındaki açıktan yola çıkarak Mutillidae uygulamasında adım adım gerçekleştirilecektir

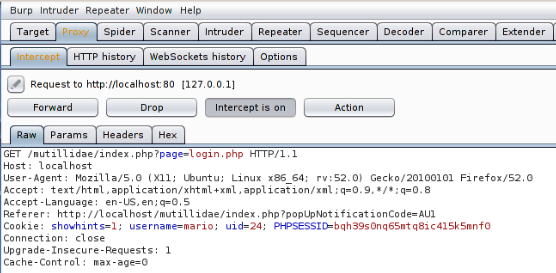
|  |  |
| --- | --- |
| URL | <http://localhost/mutillidae/index.php?popUpNotificationCode=AU1> |
| HTTP Talep Türü | Get |

* Öncelikle normal bir kullanıcı olarak uygulamada oturum açılarak cookie bigisi oluşturulur.



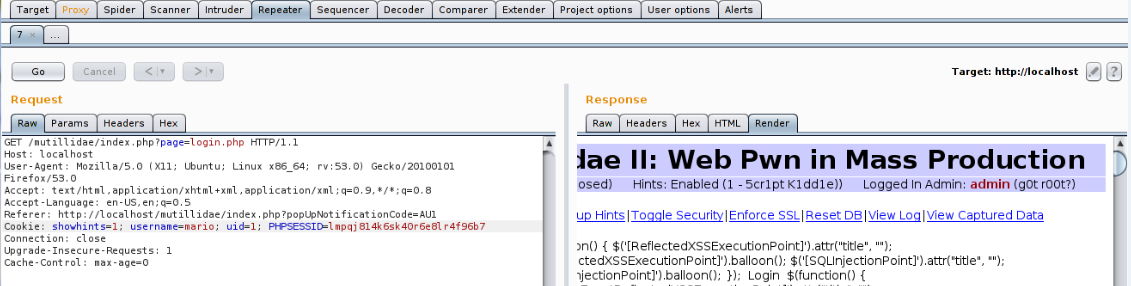
Sisteme mario kullanıcısı giriş yaptı.

* Burp Suit proxy aracında gerekli ayarlamalar yapıldıktan sonra Proxy->Intercept->Intercept is on tıklanarak istek yakalanır.



Proxy aracı ile istek yakalandı ve sunucuya kullanıcı uid’ si değiştirilerek gönderilecek.

* Yakalanan istek üzerine fare sağ tık yapılıp "Repeater" seçeneği seçilerek değişim yapıldığında sunucunun nasıl bir cevap döndürdüğünü görmek için denemeler yapılır. Sunucu tarafından gelen cevap "Response" sekmesine düşer. "response "sekmesinde "render" sekmesi acılır ve alınacak olası web sayfası görüntülenir



Proxy aracı ile istek yakalandı ve sunucuya kullanıcı uid’ si değiştirilerek cevap kontrol edildi.(uid=1)

* Bu örnekte kullanıcıyı eşsiz yapan iud değeri açık bir şekilde göründüğünden rahat bir şekilde değiştirildi. Saldırgan burada mevcut kullanıcı değerlerinin tabloda tutulduğunu ve tam yetkili kişinin uid'sinin "bir" olduğunu tahmin ederek deneme yanılma yolu ile sisteme tam yetki ile giriş yapmış oldu. Bazen bu değer belli bir özet algoritmasından geçirilebilir. Böyle bir durumda değiştirilecek değer tahmin edilen özet algoritmasından geçirildikten sonra yazılır. Özet algoritmaları için detaylı bilgi için bakınız.

https://tr.wikipedia.org/wiki/Hash\_fonksiyonu



Tam yetki almak istediğimiz için muhtemel olan admin uid‘si yazıldı. (uid=1)

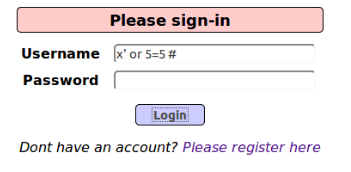
**Uygulama2**

Owasp 2013 -> A6 – Broken Authentication And Session Management -> Authentication Bypass ->Via sql injection sayfasındaki açıktan yola çıkarak Mutillidae uygulamasında adım adım gerçekleştirilecektir

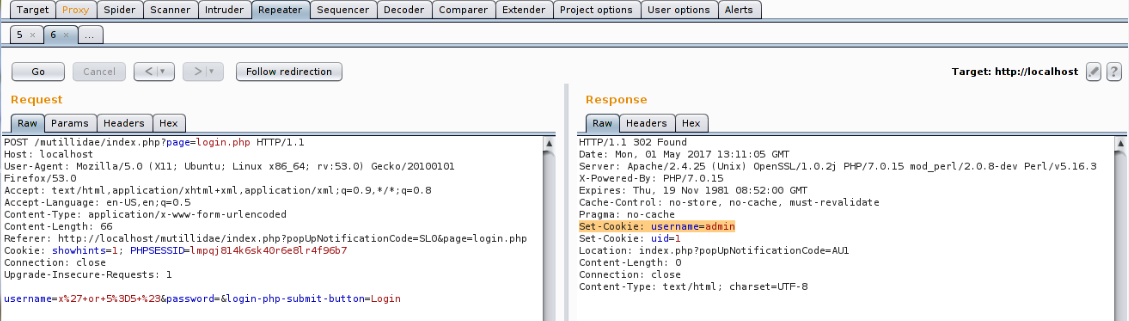
Bu uygulamadaki amaç ise sisteme giriş yapmak için kullanıcıadı ve parola girişi gerçekleştirmeden sistemde oturum açamaktır.

|  |  |
| --- | --- |
| URL | <http://localhost/mutillidae/index.phppage=login.php&popUpNotificationCode=LOU1> |
| HTTP Talep Türü | Post |
| Payload | X’ or "X" = "X" # |

Yazılan payload sayesinde istenilen kullanıcı adı ve parola yazılmasına gerek yoktur. Burda öncelikle sql açığı olup oladığını anlamak için "tek tırnak" işareti kullanılır. Eğer açık var ise saldırgana veritabanı hatası verecektir bu sayede sistemde kullanılan MYSQL veritabanını ögrenmiş olucaktır. Saldırgan arka planda "select " sorgusun çalıştığını düşünerek payload kullanacktır ve "select" sorgusu şu şekilde çalışmaktadır **WHERE ID ='$textboxa\_gelen\_deger** burda X yazılıp daha sonra tek tırnak kapatılarak sorgunun devam etmesi sağlanacaktır ve or sayesinde birinci ifade doğru değilse ikinci ifade yani "X" = "X" çalışacaktır. bu ifade sayesinde sonuç doğru dönecektir. "#" işareti ise yazılmış olan olası devam sorgusu bilinmediğinden ondan sonra gelenleri yorum kabul etmesini istemektedir. Saldırgan bu şekilde sisteme giriş yapacaktır.



Alana payload yazlır.



İsteğe karşılık cevap kontrol edilir ve admin girişi gerçekleşir.

**Cross Site Scipting (XSS)**

**Nedir?**

https://www.owasp.org/index.php/Top\_10\_2013-A3-Cross-Site\_Scripting\_(XSS)

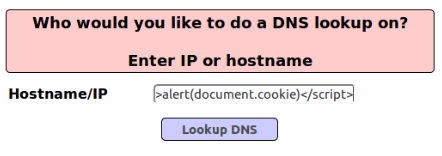
Cross Side Scripting kısaltılmış hali ile xss; tehlikeli bir güvenlik açığıdır. Farklı programlama dilleri ile yazılmış web uygulmarında bulunmaktadır. Cross Side Scripting kullanıcı verilerine dayanan bir saldırı türüdür. Temel olarak kullanıcıların bilgilerini çalmayı amaçlar.

**Uygulama1**

Owasp 2013 -> A3 – Cross Site Scipting -> Reflected (First Order) ->Dns lookup sayfasındaki açıktan yola çıkarak Mutillidae uygulamasında adım adım gerçekleştirilecektir

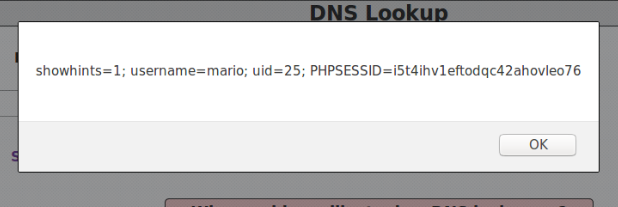
|  |  |
| --- | --- |
| URL | [http://localhost/mutillidae/index.php?page=dns-lookup.php](http://localhost/mutillidae/index.phppage=login.php&popUpNotificationCode=LOU1) |
| HTTP Talep Türü | Get |
| Payload | <script>alert(document.cookie)<script> |

* Herhangi bir şekilde bir girdi isteyen bir alan ile karşılaşıldığında eğer güvenliği alınmamış ise bu tür alanlar xss zafiyeti doğurur.



TextBox girişi

* Saaldırgan script çalıştırdığını anladıktan sonra cookie bilgisini toplamak için sekil de gösterilen payloadı kullanarak istediğini elde etmiş olur.



İstismar edildi

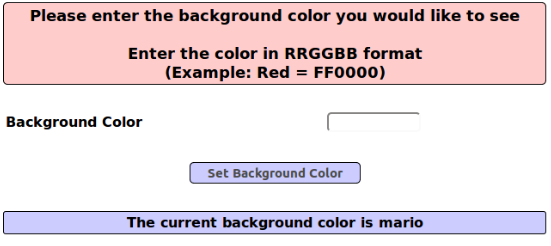
**Uygulama3**

Owasp 2013 -> A3 – Cross Site Scipting -> Reflected (First Order) -> Set Backgraund Color

sayfasındaki açıktan yola çıkarak Mutillidae uygulamasında adım adım gerçekleştirilecektir

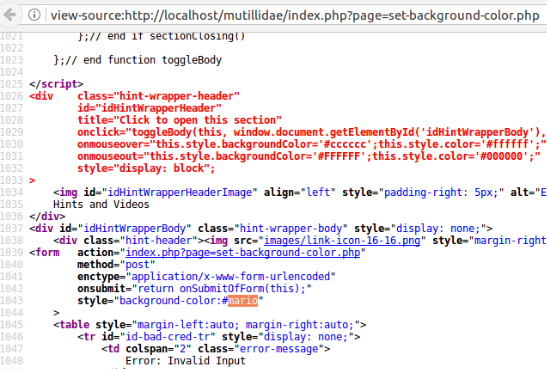
|  |  |
| --- | --- |
| Amaç | Renk kodunu yazdığımız textbox 'ı istenilen şekilde istismar etmek |
| URL | [http://localhost/mutillidae/index.php?page=set-background-color.php](http://localhost/mutillidae/index.phppage=login.php&popUpNotificationCode=LOU1) |
| Payload | "><script>alert("burda XSS var")</script><span><span" |

* Öncelikle sitenin nasıl çalıştığını anlamak adına renk yazılması gereken yere herhangi bir yazı yazılarak renk olmayan bir girdiye nasıl davrandığını görüyoruz. Yazılan "mario" yazısı renk değişimine neden olmadı fakat sayfada gösterildi. Bu sayede sayfa kaynağını görüntülerken kodun nerde çalıştığını rahat bir şekilde bulmamızı sağlıycaktır ve kodu nasıl istismar edebiliriz sorusunun cevabı alınmış olucaktır.



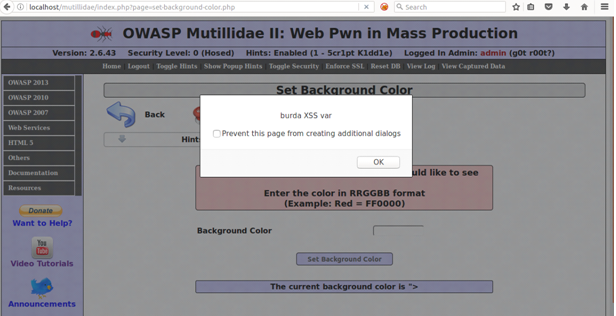
Renk yazılması gereken yerde herhangibir yazıya izin verildi.

* Rengini değiştirebileceğimiz bu sitede arka planda şöyle **style="<bockground-color:#"** bir kodun çalıştığını görüyoruz.



yazılan kelime aranır ve kod yapısı incelenir

* Kodun devamı niteliğnde olan **"><** ifade sayesinde kodun sorunsuz çalışması sağlanarak payload yazılır



Payload yazıldıktan sonra çıkan sonuç

**Uygulama 4**

Owasp 2013 -> A3 – Cross Site Scipting -> Reflected (First Order) -> HTLM5 Web Storage

sayfasındaki açıktan yola çıkarak Mutillidae uygulamasında adım adım gerçekleştirilecektir

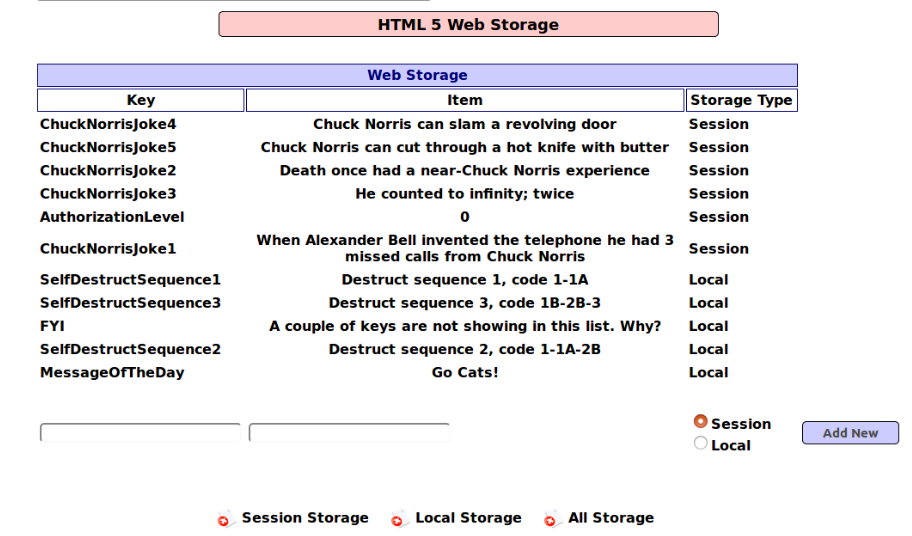
HTML5'te web sayfaları, veriyi yerel (local) olarak, tarayıcı (browser) içinde saklayabilir. Veri, sadece sorgulama olduğu zaman kullanılır; eskisi gibi sunucunun her talebinde verinin tekrar yüklenmesi söz konusu değildir. Veriyi, yerelde saklamak için HTML5'in sağladığı iki yeni nesne mevcuttur:

**localStorage**: Son kullanma tarihi söz konusu olmadan veriyi depolama olanağı sağlar. Örneğin, istemci tarafında kimlik doğrulama bilgisinin saklanması gibi.

**sessionStorage**: Veriyi sadece bir oturum için saklar.

HTML5 web depolama, web sayfaların kayıtlarını saklamaya izin veren tarayıcılar tarafından uygulanan basit bir veritabanıdır. Kayıtlar string tipinde olan değer ve anahtar çiftinden oluşuyor. Tüm HTML5 depolama biriminin ve tüm JavaScript'in istemci tarafı olduğunu ve makinenizde çalıştığı anlamına geldiğini unutmamak gerekir.

HTML5 depolama alanını kullanan bir web sitesini ziyaret ediyorsak, depolama alanı tarayıcınızda olduğundan enjeksiyon değerleri göreceli olarak kolaydır; ancak HTML5 depolama alanının alanları, protokolü ve portu kullanarak depolama değerlerini ayarlamak için kullanılan herhangi bir kodun hedef sayfa bağlamında yapılması gerekir. Bu nedenle aşağıdaki kodu enjeckte ederiz.

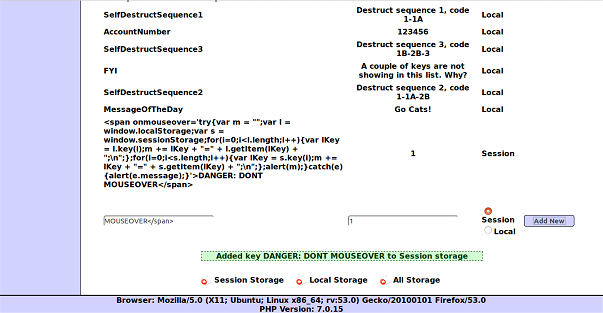


<script>

localStorage.setItem("AccountNumber","123456"); sessionStorage.setItem("EnterpriseSelfDestructSequence","A1B2C3"); sessionStorage.setItem("SessionID","japurhgnalbjdgfaljkfr"); sessionStorage.setItem("CurrentlyLoggedInUser","1233456789");

</script>

en son kullanılan script aşağıdaki şekilde alana yazıp yazarsak;



<script>

localStorage.setItem("AccountNumber","123456"); sessionStorage.setItem("EnterpriseSelfDestructSequence","A1B2C3"); sessionStorage.setItem("SessionID","japurhgnalbjdgfaljkfr"); sessionStorage.setItem("CurrentlyLoggedInUser","1233456789");

try

{

var m ="";

var l = window.localStorage;

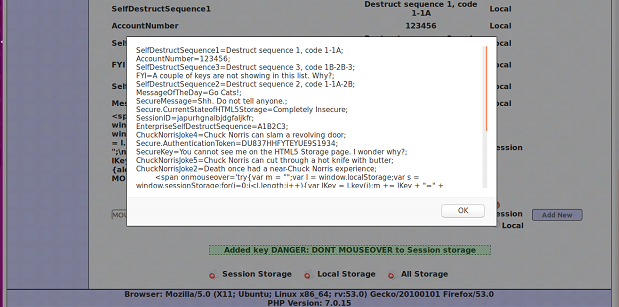
var s = window.sessionStorage;

for(i=0;i<l.length;i++){ var lKey = l.key(i); m += lKey + "=" + l.getItem(lKey) + ";\n";}; for(i=0;i<s.length;i++){ var lKey = s.key(i); m += lKey + "=" + s.getItem(lKey) + ";\n";}; alert(m);

}

catch(e){ alert(e.message); }

</script>



Farenin haraket etmesi ile tüm verileri ekrana yazdırır.

**3.5 XML Validator (XML dogrulama)**

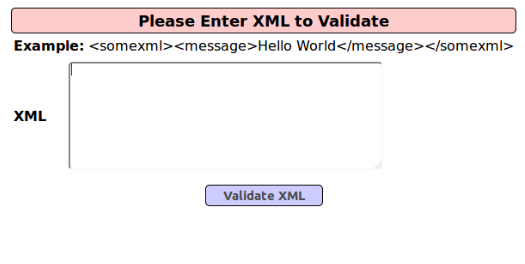
|  |  |
| --- | --- |
| Amaç | İstenilen verileri görüntelemek |
| URL | <http://localhost/mutillidae/index.php?page=xml-validator.php> |
| Parametre | Get |
| Payload | <?xml version="1.0"?>  <!DOCTYPE change-log [ <!ENTITY systemEntity SYSTEM "robots.txt">]>  <change-log>  <text>&systemEntity;</text>  </change-log> |

şekildeki gibi xml girdisi kabul eden bir alanda saldırgan öncelikle xml ifadelerini kabul edip etmediğini anlmak için bir kaç tane örnek değer girerek sistemi öğrenecektir.

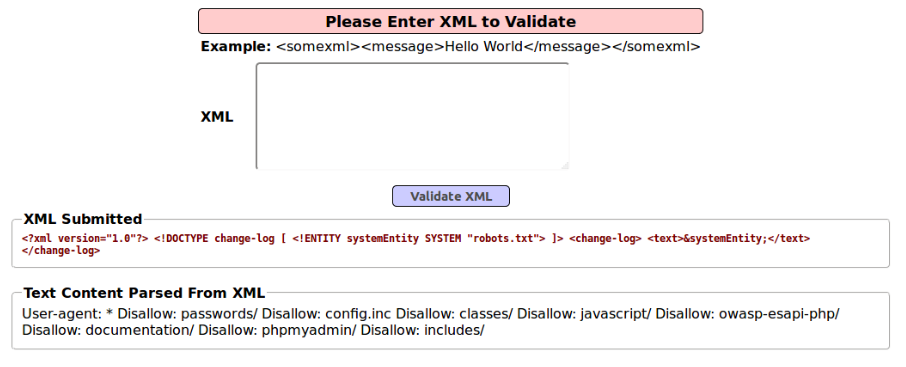
<?xml version="1.0"?><change-log><text>Mario</text></change-log>

<?xml version="1.0"?><change-log><text>&quot;Mario&quot;</text></change-log>

Bir XML belgesinin <! ENTITY> bölümü isteğe bağlı olarak XML belgesinin parçası olarak dahil edilecek harici dosyaları tanımlar.



XML girişi



Payload değiştirilerek istenilen kritik dosyalar sistemden çekilebilir.

**3.6 poll user**

|  |  |
| --- | --- |
| Amaç | Güvenliği alınmamış url den veri değişimi yapabilmek |
| URL | <http://localhost/mutillidae/index.php?page=user-poll.php&csrf-token=&choice=kismet&initials=&user-poll-php-submit-button=Submit+Vote> |
| Parametre | Get |

Url ‘ den hemen sonra &choice=istediğimiz\_kelime yazlılarak sayda oluşturabilirz.

Url bundan kurtarmak için şöyle bir php kodu yazılabilir.

switch ($\_SESSION["security-level"]){

case "0": // This code is insecure

$lEnableHTMLControls = FALSE;

$lEncodeOutput = FALSE;

$lProtectAgainstMethodTampering = FALSE;

$lHTTPParameterPollutionDetected = FALSE;

$lLoggedInUser = $logged\_in\_user;

break;

case "1": // This code is insecure

// DO NOTHING: This is insecure

$lEnableHTMLControls = TRUE;

$lEncodeOutput = FALSE;

$lProtectAgainstMethodTampering = FALSE;

$lHTTPParameterPollutionDetected = FALSE;

$lLoggedInUser = $logged\_in\_user;

break;

case "2":

case "3":

case "4":

case "5": // This code is fairly secure

$lEnableHTMLControls = TRUE;

$lEncodeOutput = TRUE;

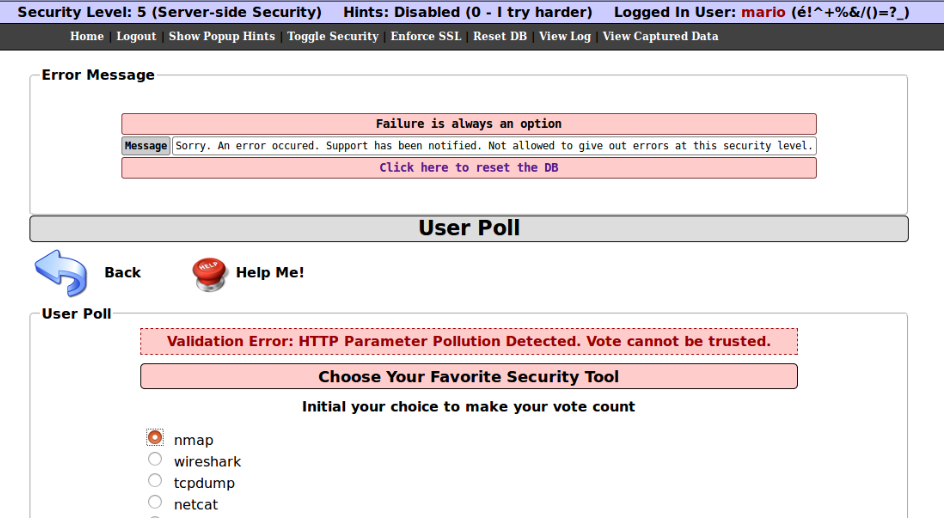
$lProtectAgainstMethodTampering = TRUE;

$lHTTPParameterPollutionDetected = isParameterPollutionDetected($\_SERVER['QUERY\_STRING']);

$lLoggedInUser = $MySQLHandler->escapeDangerousCharacters($logged\_in\_user);

break;

}// end switch



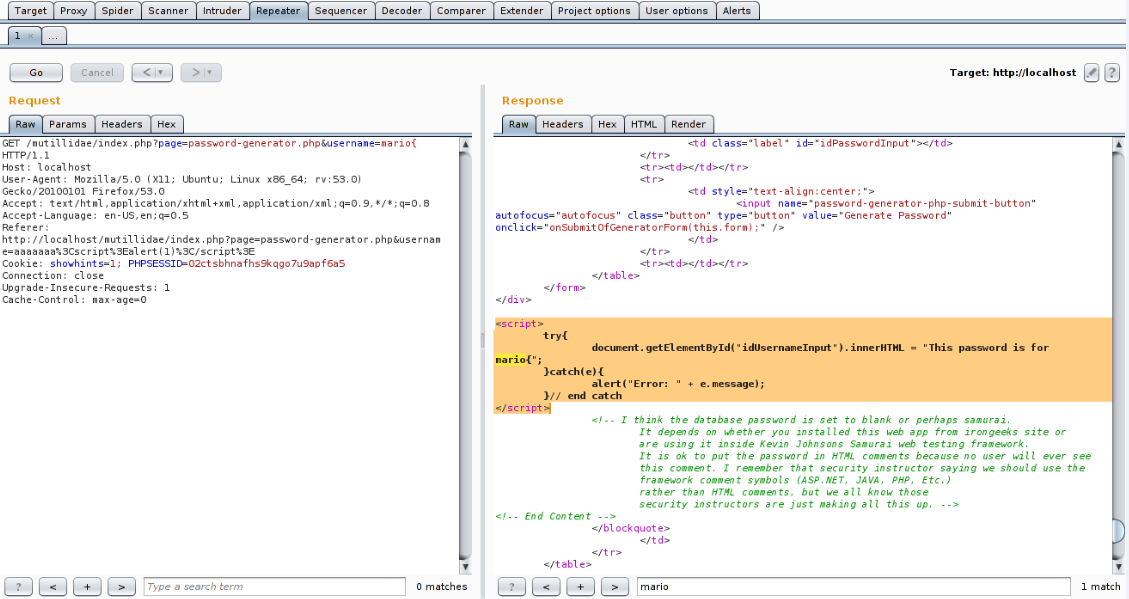
Bu kod sayesinde url de başka parametrelere izin vermekyecektir. Ve Şöyle bir hata verecektir.

**3.7 Password generate**

|  |  |
| --- | --- |
| Amaç | Javascript üzerinden xss yapmarak siteyi istismar etmek. |
| URL | <http://localhost/mutillidae/index.php?page=password-generator.php&username=mario> |
| Parametre | Get |

HTTPFox ve Burp Suite, bir sayfayı "GET" ederken tüm girdileri görmek için iyi bir araçtır (istek) ve sunucunun yanıt verdiği tüm çıktıyı (yanıt).

Sayfa girdisini bulduktan sonra, "CANARY-INPUT-1" gibi basit bir kanarya enjekte etmeyi deneyin sonra kanaryanın nereden geldiğini görmek için sonuç sayfasında arama yapın. Giriş, Javascript'in bir bölümünü oluşturmak için kullanılıyorsa, bağlamdan kurtulmak ve kullanıcı tarafından kontrol edilen Javascript enjekte etmek mümkün olabilir. Çıktının kodlanıp kodlanmadığını test etmek için "CANARY123! @ # $% ^ & \* () \_ + - = [] {}; ':", ./ <>? "Gibi bir test dizesi enjekte edin.



Gerçek tarayıcı çıktısında bir kanarya aramak iyi bir fikir değildir. "Gerçek" yanıtı görmek için "kaynak görüntüleme" tarayıcılarını kullanın. HTTPFox gibi araçlar bunun için de mükemmeldir. Burp gibi daha fazla özelliklere sahip araçlar daha da iyi fakat öğrenme eğrisi daha fazla. Burp, siteyi örümcekken ziyaret ettiğiniz her sayfanın kaynağını hatırlayacak.

