XSS (Content-Sniffing)

Web tarayıcılar web sunuculardan gelen yanıt paketlerini okurken yanıt paketlerindeki Content-Type başlığına bakarlar ve ona göre paket içeriğini okuma / yorumlama ve ekrana sunma uygularlar. Fakat yanıt paketlerinde Content-Type başlığı bozuk değerde olursa, Content-Type başlığı paket içeriğiyle uyuşmazsa, veya Content-Type başlığı hiç yer almazsa yanıt paketinin doğru okunabilmesi / yorumlanabilmesi ve ekrana sunulabilmesi amacıyla eski web tarayıcı yazılımları (özellikle Internet Explorer'lar) yanıt paketi içerisindeki içerik üzerinde Content Sniffing prosedürünü uygularlar. Bu şekilde gelen paket içerisindeki içeriğin MIME türünü saptarlar ve bu bilgi doğrultusunda paketi okuma / yorumlama yapıp bize arayüzde sunarlar. Böylece yanıt paketi Content-Type referansı olmadan / kullanılmadan uygulama sayfasını doğru bir şekilde görüntüleyebiliyor oluruz.

Content-Type başlığının bozuk değerde olması, paket içeriğiyle uyuşmaz değerde olması veya hiç olmaması / eksik olması eski web tarayıcıların yanıt paketini ekrana sunması noktasında Content-Sniffing özelliğinin çalışmasını tetikler. Böylece paket Content-Type bilgisi paketin gövdesindeki veride Content Sniff'leme yapılmak suretiyle elde edilir ve paket okuması / yorumlaması buna göre yapılarak ekrana sunma gerçekleşir.

> Bilgi:
>
> Eski IE web tarayıcılar Content-Type yanıt başlığı doğru formatta olsa da veya doğru veri türünü gösterse de her halükarda Content Sniffing yapmaktadırlar ve Content Sniffing ile belirlenen veri türüne göre paket okuması / yorumlaması yapmaktadırlar. Content Sniffing özelliğini bu şekilde kullandıkları için XSS (Content - Sniffing) zafiyetinin sömürülmesi noktasında IE web tarayıcılar öndedirler.

Web tarayıcılardaki Content Sniffing özelliği web uygulamalara bir esneklik sunmak amacıyla tasarlanmıştır. Bu sayede web uygulamalarda paket türü bilgisi bozuk değerde olduğunda, paket gövdesindeki dökümanla aynı veri türünde olmadığında veya eksik olduğu durumlarda tarayıcılar yine de tür bilgisini kendileri saptayıp görüntüleme sunabilmekteler. Fakat bu aynı zamanda bir güvenlik riski teşkil etmektedir.

Örneğin bir web uygulamada girdi noktasına girilen geçersiz girdi durumunda json dosya yanıtıyla hata mesajı dönüldüğünde hataya sebep olan girdinin json yanıtında yer alması durumunda bu hata yanıtı, yani json dosyası kullanıcılarca ve/veya admince eski web tarayıcılarda görüntülendiğinde içindeki xss payload'ları çalışabilir ve xss saldırısı yaşanabilir.

Örneğin bir web uygulamaya dosya yükleme mekanizması ile xss payload'ları içeren txt dosyası yüklenebilir. Sadece txt dosyası kabul eden dosya yükleme mekanizması ile bu şekilde sadece zararsız dosya alınıyor şeklinde düşünülebilir. Fakat txt dosyası kullanıcılarca ve/veya admince eski web tarayıcılarda görüntülendiğinde content sniffing nedeniyle içindeki xss payload'ları çalışabilir ve xss saldırısı yaşanabilir.

XSS (Content-Sniffing) saldırısı kötü niyetli kullanıcının girdi olarak verdiği xss payload'unun geri sunulacağı sayfada normalde string olarak yansıması beklenirken web tarayıcının yanıt paketinde Content Sniff'leme yapması ve paketin türü bilgisini kullanıcı girdisi türü bilgisine göre değerlendirip ona göre paketi okuması / yorumlaması sonrası sunması sonucu xss payload'unun çalışır halde tarayıcıya yansıtılmasına denir. XSS (Content-Sniffing) zafiyeti Content Sniffing yapma prosedürüne sahip web tarayıcılarda meydana gelir.

Not: Tarayıcılarda Content Sniff’leme özelliği spesifik anlarda tetiklenir. Bunun için bir http spesifikasyonuna sahiptirler.

Bu saldırıdan kullanıcıları korumak için bir http güvenlik başlığı olan X-Content-Type-Options yanıt paketlerine eklenmelidir. Bu sayede yanıt paketi Content-Type bilgisi bozuk değerde olsa da, paket içeriğiyle uyuşmayan değere sahip olsa da veya Content-Type bilgisi eksik olsa da X-Content-Type-Options yanıt başlığı tarayıcıya Content Sniff’leme özelliğini çalıştırmayacaksın direktifi verecektir ve paket türü bilgisi için saptama çalıştırılmayacaktır. Paket türü bilinmeyen olarak bırakılıp içeriğin farklı şekilde yorumlanıp ekrana sunulması önlenecektir. Bu sayede kullanıcı girdisinin geri yansıtıldığı yanıt paketlerinde girdinin normalde string kalması gerekirken çalıştırılabilir hale gelmesi önlenebilir. Yani XSS için xss payload’u string kalmalıyken çalıştırılabilir olması önlenebilir.

Sahadan Bir Örnek

Örneğin uygulama bir sayfası parametreye girdi olarak verilen değer sonucu json formatta hata yanıtı dönmektedir. Uygulamanın bu sayfasındaki parametreye XSS payload’u verildiğinde yanıt olarak hata mesajı içerisinde XSS payload’u dönmektedir. Fakat yanıt paketi json formatta hata mesajını döndüğünden içerisindeki XSS payload’u (javascript kodu) çalışmamaktadır.

// Uygulama sayfasındaki op parametresine XSS payload’u girilir

Http Talep:

```
GET /webhdfs/v1/?op=LISTSTATUS'"()%26%25<acx><ScRiPt%20>J4Um(9462)</ScRiPt> HTTP/1.1
Referer: http://X.X.X.X:5070/
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: X.X.X.X:5070
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, l
ike Gecko) Chrome/73.0.3683.103 Safari/537.36
Connection: Keep-alive
```

Yanıt olarak dönen paket json formattadır ve içerik olarak json içerik barındırırken gönderdiğimiz XSS payload'unu da hataya sebebiyet verdi diye bilgilendirmek maksatlı geri yansıtmaktadır.

Http Yanıt:

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Expires: Wed, 08 Jan 2020 07:30:25 GMT
Date: Wed, 08 Jan 2020 07:30:25 GMT
Pragma: no-cache
Expires: Wed, 08 Jan 2020 07:30:25 GMT
Date: Wed, 08 Jan 2020 07:30:25 GMT
Pragma: no-cache
Content-Type: application/json                    // Paket Json Formatta
X-FRAME-OPTIONS: SAMEORIGIN
Server: Apache
```

Content-Length: 288

{"RemoteException":{"exception":"IllegalArgumentException",
"javaClassName":"java.lang.IllegalArgumentException",
"message":"Invalid value for webhdfs parameter \"op\": No enum
constant org.apache.hadoop.hdfs.web.resources.
GetOpParam.Op.LISTSTATUS\"()&%<ACX><SCRIPT >
J4UM(9462)</SCRIPT>"}} // XSS Payload'u Barındırıyor

Uygulama bu sayfasındaki hata mesajlarını json formatta dönmektedir. Json formatta gelen hata mesajı içerisindeki XSS kodu ise paket formatı gereği json okumasıyla okunduğundan kullanıcı ekranında çalışmayacaktır. Bu XSS kodu sayfada hata mesajıyla beraber json içeriğinde string olarak yer edecektir. Eğer uygulama sunucusunda yanıt paketlerine X-Content-Type-Options başlığı ilavesi yapılmamaktaysa Content-Type ile eş durumda olmayan paket alt içeriği XSS payload girdisi eski web tarayıcılarda Content-Type düzeltmesi yapmak için Content Sniffing özelliğini çalıştırmayı tetikleyebilir ve bu yolla paket türü bilgisi alt içeriğe göre şekillenerek paket okuması / yorumlaması yapılabilir. Bu ise alt içeriğin tarayıcı ekranında string olarak değil, çalışır halde yansıması demek olur. Dolayısıyla kullanıcılar op parametresine XSS payload'u ekli halde url adresine gittiklerinde tarayıcıların Content Sniffing özelliğini kullanmaları ile Content-Type düzeltmesi yaşamaları ve ekrana yansıtılan yanıt paketini bu türden görüntülemeleri sonucu zarar görebilirler.

Gerçek bir senaryoda bu paketi oluşturup yollayacak bir url adresi hazırlanabilir ve çeşitli yöntemlerle kurbanlara aktarılarak tıklanması sonucu XSS saldırısı uygulanabilir ve kurbanların çerezleri çalınabilir, hangi tuş takımını tuşladıkları bilgisi toplanabilir, ddos yaptırılabilir, ... vs.

Sonuç olarak eski web tarayıcılardaki Content-Sniffing özelliği XSS saldırılarına imkan tanımaktadır. Eski web tarayıcılarda Content Sniffing yoluyla normalde çalıştırılabilir olmaması gereken unsurların çalıştırılabilir hale geçmesi XSS için elverişli bir yol açar. Örneğin txt dosyasındaki xss payload'larının çalışması gibi veya json dosyasındaki xss payload'larının çalışması gibi. Dolayısıyla eski web tarayıcı kullanan kullanıcılar için olası XSS saldırıları yaşamamaları adına X-Content-Type-Options başlığı kullanılmalıdır ve eski web tarayıcıların content sniffing'lemesinin önüne geçilmelidir. X-Content-Type-Options güvenlik başlığı eski web tarayıcıların content sniffing yapmamasını sağlar.

Uygulama

Bu uygulamada XSS (Content-Sniffing) saldırısı örneği uygulanacaktır ve sonra saldırı önlemi olan http güvenlik başlığı kullanılarak saldırının bertaraf edilişi gösterilecektir.

Gereksinimler

Ubuntu 18.04 LTS - Apache	// Web Sunucu (Ana Makina)
/var/www/XSS (Content-Sniffing) Uygulaması	// Demo Web Sayfası
Windows 7 Home Premium	// Son Kullanıcı Bilgisayarı (Sanal Makina)
Internet Explorer 8	// Son Kullanıcı Web Tarayıcısı

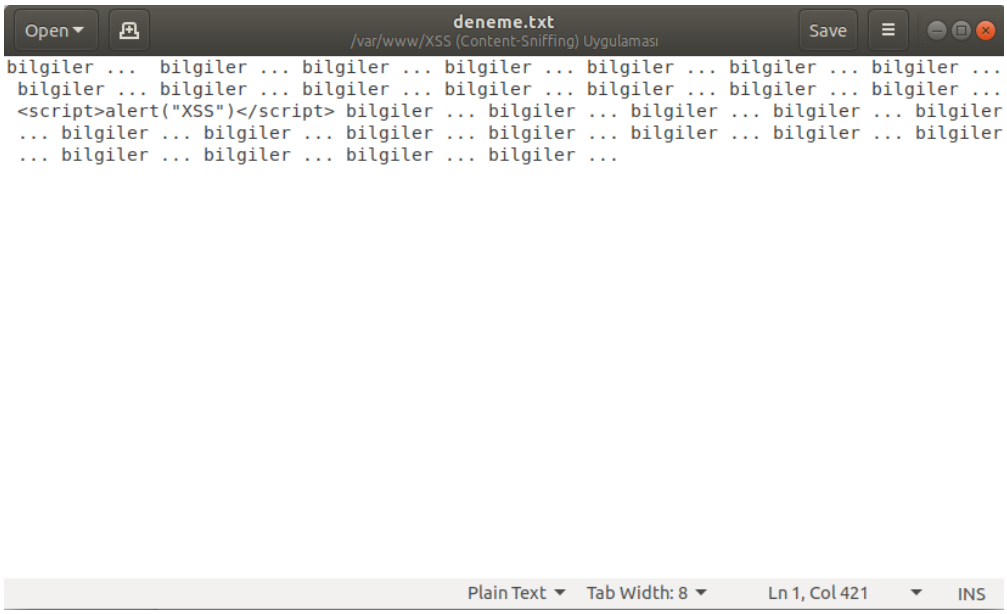
Demo web sayfasına bir göz atalım.

Ubuntu 18.04 LTS:

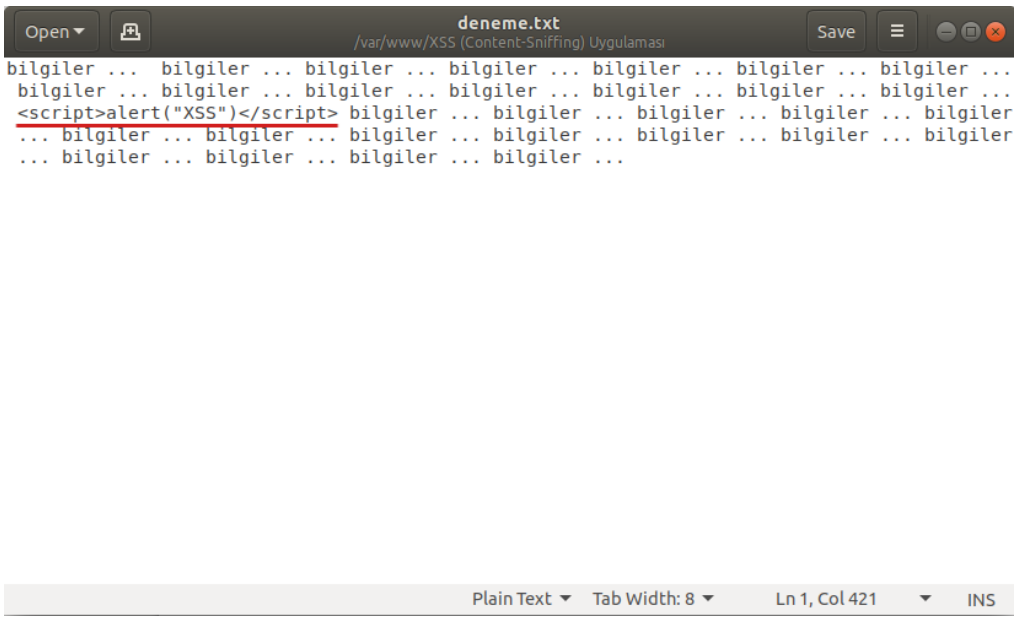
> cd "/var/www/XSS (Content-Sniffing) Uygulaması/"
> ls

Çıktı:

deneme.txt

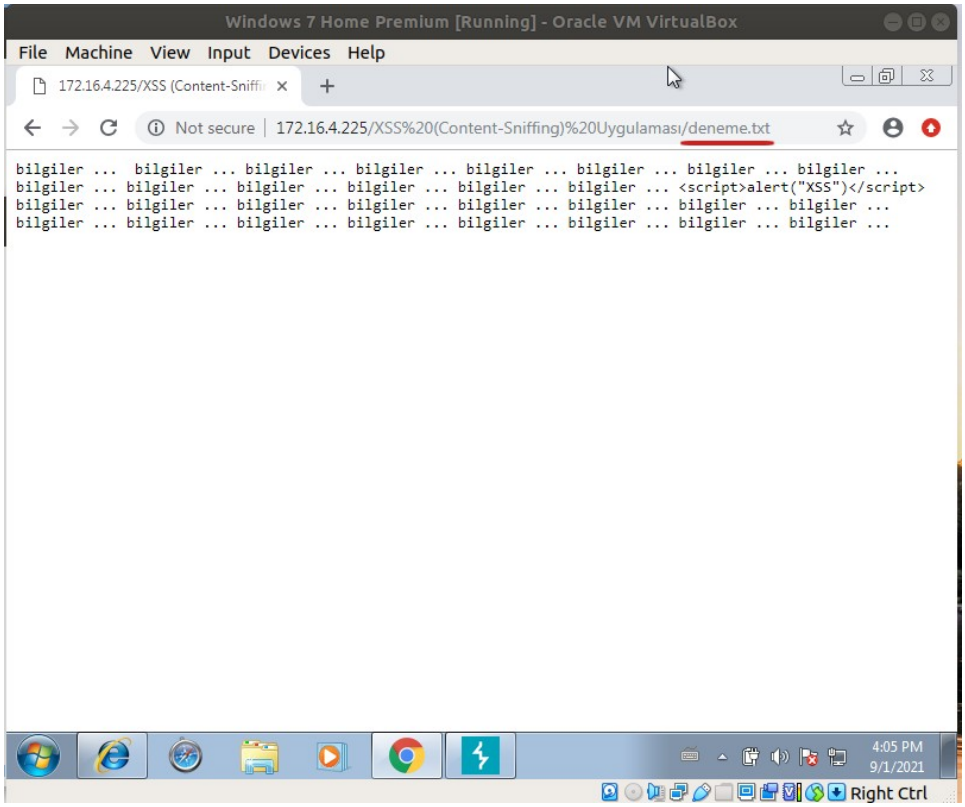


Web sunucusundaki deneme.txt dosyasına modern web tarayıcılardan erişildiğinde web tarayıcılar txt metin belgesi görüntüleyeceklerdir. Metnin arasında yer alan <script> … </script> bloğu metin olarak kalacaktır ve çalışmayacaktır.



Örneğin son kullanıcı bilgisayarı Windows 7 Home Premium VM'den Chrome web tarayıcısı ile hedef web sunucudaki bu txt dosyaya erişildiğinde ekrana metin belgesi gelecektir.

Windows 7 Home Premium VM [Chrome Web Tarayıcısı]:

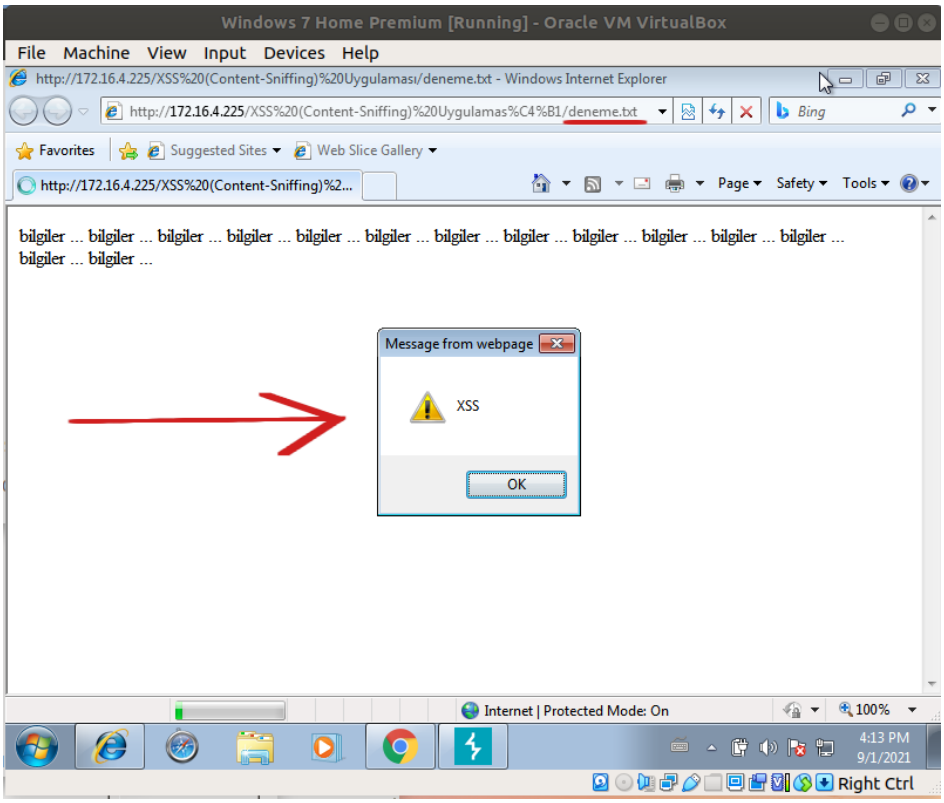


Ancak eski web tarayıcılarda (örn; IE8'de) aynı dosyayı görüntülemeyi tekrarladığımızda web tarayıcı gelen yanıt paketinin gövdesinde Content Sniffing yapacağından, yani yanıt paketinin gövdesindeki içeriğe tür tespit etme işlemi uygulayacağından içerik metin belgesi olarak değerlendirilmeyecektir. Çünkü content sniffing sırasında içerikte <script>...</script> bloğuna

rastlanacağından paketin gövdesindeki içerik javascript belgesi olarak belirlenecektir ve o şekilde görüntülecektir.

Şimdi aynı dosyaya son kullanıcı bilgisayarı Windows 7 Home Premium VM'den bu sefer Internet Explorer 8 web tarayıcısı ile erişelim.

Windows 7 Home Premium VM [Internet Explorer 8 Web Tarayıcısı]:



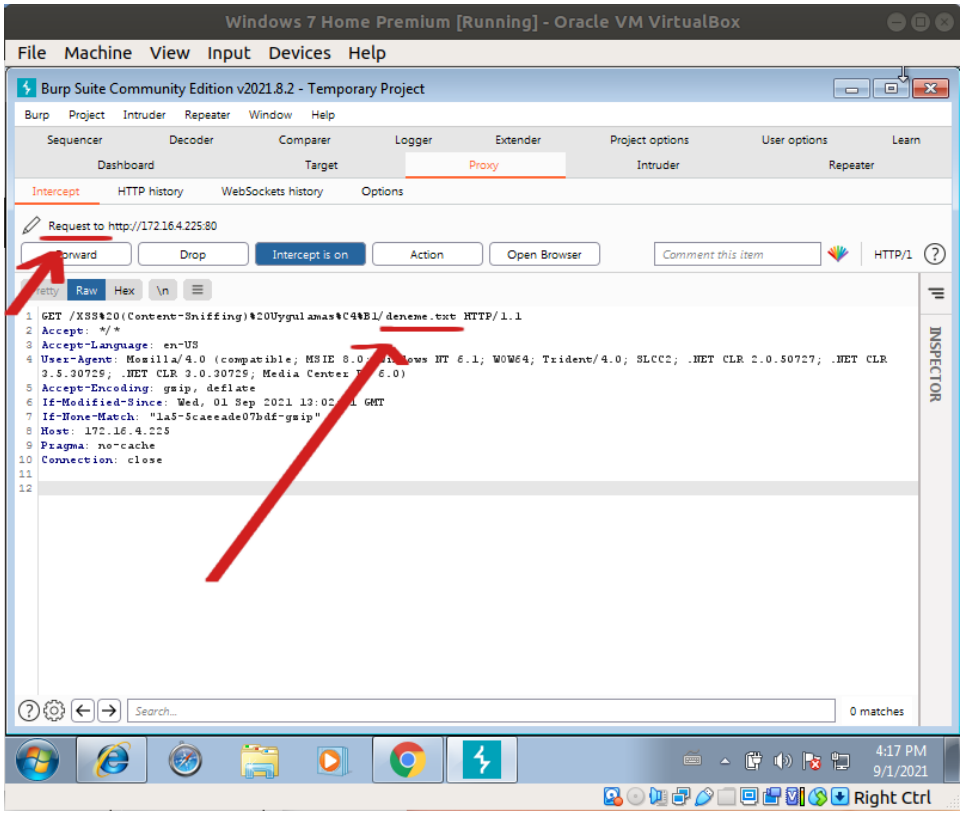
Görüldüğü gibi Windows 7 Home Premium VM'de yüklü gelen Internet Explorer 8 web tarayıcısı ile deneme.txt dosyasına eriştiğimizde metin belgesi içerisindeki javascript kodu çalışır olmuştur ve ekrana popup gelmiştir.

NOT:

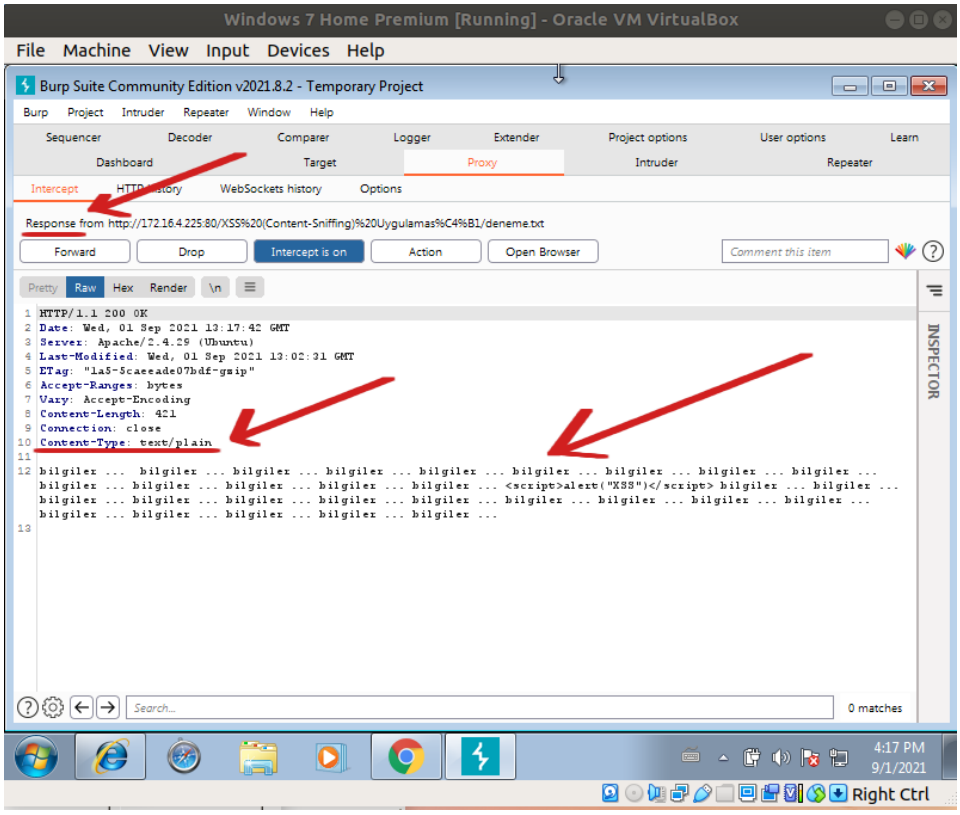
Web sunucudaki txt belgesinin içerik uzunluğu string'lerle arttırıldığında IE8 web tarayıcıda content sniff'leme bir noktadan sonra çalışmadı ve javascript bloğu olsa da yanıt paketinde gelen belge txt şeklinde yorumlandı. Dolayısıyla bu şekilde bir pratik durum söz konusu.

Deneme.txt belgesine erişirken gönderilen ve gelen paketlere bakacak olursak;

Http Request:



Http Response:



paketler olması gerektiği gibidir. Gönderilen pakette deneme.txt dosyası talebi yapılmaktadır ve gelen pakette Content-Type başlığı ile paketin gövdesindeki içeriğin txt belgesi olduğu belirtilmektedir Eski web tarayıcılar (bu örnekte Internet Explorer 8) yanıt paketinin gövdesinde content sniffing yaptıklarından gelen belgenin türünü kendileri belirlerler. Bu belgede javascript kodu olduğundan content sniffing ile belgeyi javascript belgesi kabul ederler ve çalışmaması gereken javascript bloklarının çalışmasına sebep olurlar.

Son olarak demo web sayfasının bulunduğu web sunucuda konfigurasyon ayarı yapalım ve X-Content-Type-Options yanıt başlığı önlemini ekleyelim. Böylece eski web tarayıcılar content sniff'leme özelliklerini kullanmasın direktifi verelim.

Ubuntu 18.04 LTS:

> sudo nano /etc/apache2/apache2.conf

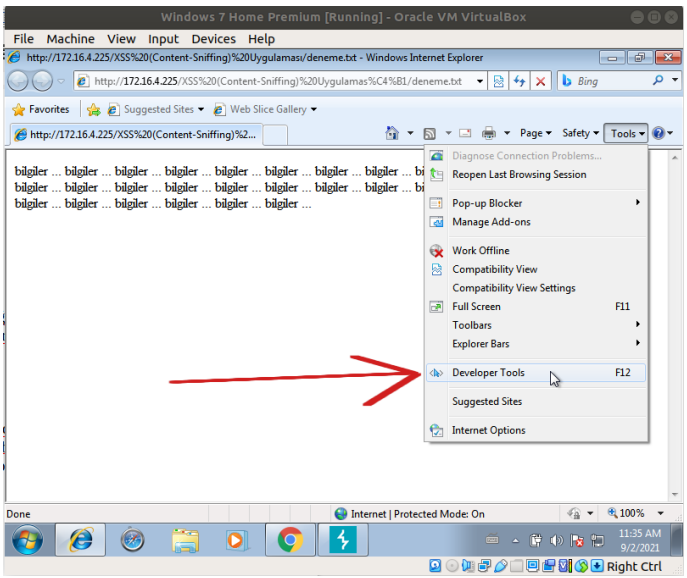
(En Alta Eklenir)
…
…
Header set X-Content-Type-Options "nosniff"

> service apache2 restart

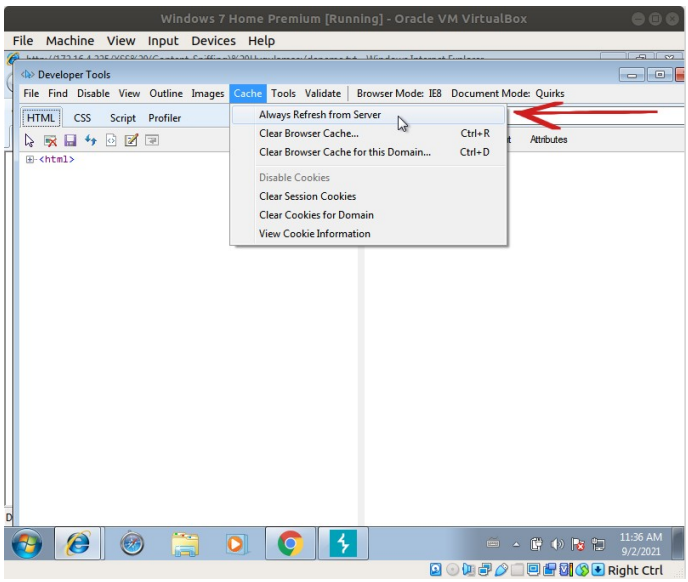
Başlık eklendikten sonra eski bir web tarayıcı olan Internet Explorer 8'den tekrar demo web sayfasına erişmeye çalışalım.

(!) Uyarı:

IE8'de cache'leme özelliği kapatılmalıdır. Aksi takdirde web sunucudan yanıt paketi almak yerine cache'den yanıt paketi geleceğinden uygulanan X-Content-Type-Options başlıklı yanıt paketi alınamayacaktır. Dolayısıyla önlemin çalışırlığı gözlemlenemeyecektir. Cache kapatıldığında X-Content-Type-Options başlıklı yanıt paketi gelecektir ve IE8'in content sniffing yapması önlenerek script bloğunun string kalması sağlanabilecektir.



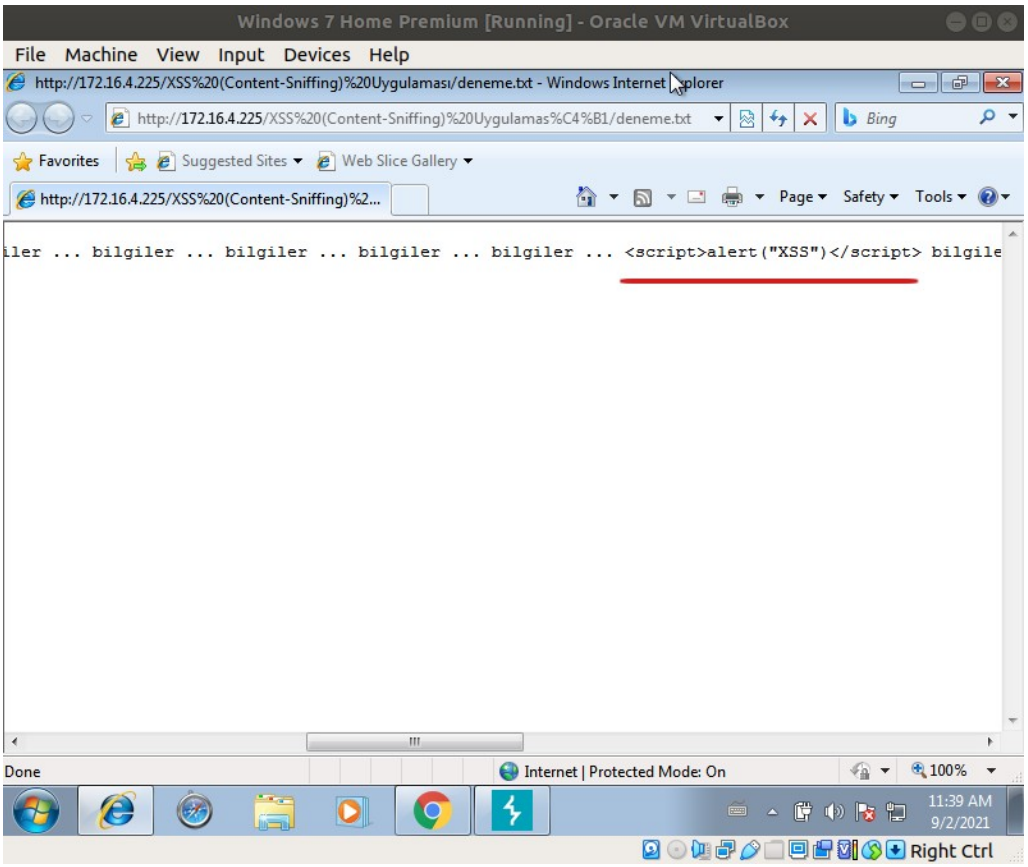
Internet Explorer 8'de Cache Kapama I



Internet Explorer 8'de Cache Kapama II

Internet Explorer 8'de (cache kapatıldıktan sonra) sayfayı yenilediğimizde txt dosyası bu sefer txt olarak görüntülecektir.

Windows 7 Home Premium VM [Internet Explorer 8 Web Tarayıcısı]:



Görüldüğü üzere X-Content-Type-Options etkinken web tarayıcı gelen yanıt paketindeki belgede content sniffing bu sefer yapmadığından txt belgedeki script bloğu çalışır hale geçmemiştir. Yanıt paketinin de belirttiği gibi belge txt olarak görüntülenmiştir.

Kaynaklar

https://en.wikipedia.org/wiki/Content_sniffing
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options
https://www.geeksforgeeks.org/http-headers-x-content-type-options/
https://geekflare.com/http-header-implementation/
https://www.acunetix.com/vulnerabilities/web/cross-site-scripting-content-sniffing/
https://www.denimgroup.com/resources/blog/2019/05/mime-sniffing-in-browsers-and-the-security-implications/
https://hackerone.com/reports/363845
https://security.stackexchange.com/questions/57615/content-sniffing-xss-vulnerable-browsers
https://www.denimgroup.com/resources/blog/2019/05/mime-sniffing-in-browsers-and-the-security-implications/
https://vulncat.fortify.com/en/detail?id=desc.dataflow.java.cross_site_scripting_content_sniffing