TCP/IP Ağlarda İleri Seviye Paket Analizi

Tshark Kullanarak Paket Analizi

Huzeyfe ÖNAL<honal@bga.com.tr> – Çağdaş DOĞRU <cdogru@bga.com.tr>

11/17/2010

[Bu yazı tshark aracı kullanarak günümüz iletişim dünyasının temeli olan TCP/IP protocol ailesinin ileri seviye ağ analizini içermektedir.]

İçerik

Giriş

Günümüz iletişim dünyasının temeli olan TCP/IP'nin önemi gün geçtikce artmaktadır. TCP/IP demek özünde paket, protokol demektir. TCP/IP ağlarda dolaşan tüm veriler birer paketdir ve bu paketlerin yapısı ne kadar iyi bilinirse TCP/IP ve güvenliği konusunda çıkabilecek sorunlar o kadar kolay anlaşılır ve çözüme kavuşturulur.

Sınır güvenliği(Router, Firewall, IDS, IPS, NDLP vs) bileşenlerinin temeline inildiğinde karşımıza çıkacak en önemli iki bileşen paket ve protokol olacaktır. Bir saldırı tespit ve engelleme sisteminin iyi paketle kötü paketi ayırt edebilmesi tamamen onu tasarlayan mühendisin ona saldırı imzası yazan sistem yöneticisinin paket, protokol bilgisiyle doğru orantılıdır.

Paket, protokol kavramlarının detaylı olaran anlaşılmasının en kolay yolu "Sniffer" olarak da adlandırılan ağ paket/protokol analiz programlarıyla pratik çalışmalar yapmaktır. Bu yazıda siber dünyada en sık kullanılan paket/protokol analiz programlarından Wireshark'ın komut satırı versiyonu kullanılarak ileri seviye paket, protokol analizi çalışmaları gerçekleştirilmiştir.

Paket Analizi

Paket, Protokol Kavramları

Paket ve protokol birbirleri yerine sık kullanılan ama gerçekte birbirinden farklı iki kavramdır. Paket kavramı protokol kavramına göre daha kuşatıcıdır(paket>protokol). Paket'den kastımız TCP/IP ağlarda tüm iletişimin temelidir, protokol ise paketlerin detayıdır.

Gönderip alınan mailler, web sayfası ziyaretleri , -VOIP kullanılıyorsa- telefon konuşmaları vs arka planda hep paketler vasıtasıyla hedef sisteme ulaştırılır. Bu paketleri izleme ve inceleme "sniffer" adı verilen programlar vasıtasıyla mümkündür.

Bir de bu paketler içerisinde gidip gelen protokoller vardır. Mesela web sayfalarına giriş için HTTP, 3G ya da GPRS bağlantıları için GTP, mail için SMTP gibi… Bu protokollere taşıyıcılık yapan daha alt seviye protokoller de vardır TCP, IP, UDP gibi. Tüm bu protokoller iletişime geçmek isteyen uçlar arasında azami standartları belirlemek için düşünülmüştür.

Paket ve protokol analizi için sniffer araçları kullanılır. Bazı snifferlar kısıtlı protokol analizi yapabilirken bazı snifferlar detaylı paket ve protokol analizi yapmaya olanak sağlar. Kısıtlı paket ve protokol analizine imkan sağlayan sniffer olarak tcpdump'ı, gelişmiş paket ve protokol analizine örnek olarak da Wireshark/Tshark örnek verebilir.

Paket İnceleme/Analiz Araçları

Internet üzerinden edinilebilecek oldukça fazla paket, protokol inceleme yazılımı vardır. Bunların arasında bazı yazılımlar çeşitli özellikleriyle ön plana çıkar. Mesela tcpdump, Wireshark, Snort gibi trafik analiz araçlar hem açık kaynak kod olarak dağıtılmaktadır hem de alternatiflerine oranla daha fazla protokol desteği sunmaktadırlar. Platform bağımsız bir şekilde tüm işletim sistemlerinde çalışmaları da cabası.

Tshark Kullanarak Paket Analizi

Tshark Nedir?

Tshark, açık kaynak kodlu güçlü bir ağ protokolleri analiz programıdır. Tshark komut satırından çalışır ve yine bir ağ trafik analiz programı olan Wireshark'da bulunan çoğu özelliği destekler.

Akıllara Wireshark varken neden Tshark kullanılır diye bir soru takılabilir?
Bu sorunun çeşitli cevapları olmakla birlikte en önemli iki cevabı Tshark'ın komut satırı esnekliği sağlaması ve Wireshark'a göre daha performanslı çalışmasıdır.

Basit Tshark Kullanımı

Tshark, çeşitli işlevleri olan bir sürü parametreye sahiptir. Eğer herhangi bir parametre kullanılmadan çalıştırılırsa ilk aktif ağ arabirimi üzerinden geçen trafiği yakalayıp ekrana basmaya başlar.

cyblabs ~ # tshark
Capturing on eth0
0.000000 192.168.2.23 -> 80.93.212.86 ICMP Echo (ping) request
0.012641 80.93.212.86 -> 192.168.2.23 ICMP Echo (ping) reply
0.165214 192.168.2.23 -> 192.168.2.22 SSH Encrypted request packet len=52
0.165444 192.168.2.22 -> 192.168.2.23 SSH Encrypted response packet len=52
0.360152 192.168.2.23 -> 192.168.2.22 TCP pcia-rxp-b > ssh [ACK] Seq=53 Ack=53 Win=59896 Len=0
0.612504 192.168.2.22 -> 192.168.2.23 SSH Encrypted response packet len=116
1.000702 192.168.2.23 -> 80.93.212.86 ICMP Echo (ping) request
1.013761 80.93.212.86 -> 192.168.2.23 ICMP Echo (ping) reply
1.057335 192.168.2.23 -> 192.168.2.22 SSH Encrypted request packet len=52
16 packets captured

Eğer çıktıların ekrana değil de sonradan analiz için bir dosyaya yazdırılması isteniyorsa -w dosya_ismi parametresi kullanılır.

cyblabs ~ # tshark -w home_labs.pcap
Running as user "root" and group "root". This could be dangerous.
Capturing on eth0
24

Gerektiğinde home_labs.pcap dosyası libpcap destekli herhangi bir analiz programı tarafından okunabilir. Tshark ya da tcpdump ile kaydedilen dosyadan paket okumak için -r parametresi kullanılır.

 Tshark çıktılarının anlaşılır text formatta kaydedilmesi isteniyorsa tshark komutu sonuna **> dosya_ismi** yazılarak ekranda görünen anlaşılır çıktılar doğrudan dosyaya yazdırılmış olur.

Dinleme Yapılabilecek Arabirimleri Listeleme

root@cyblabs:~# tshark -D
1. eth0
2. usbmon1 (USB bus number 1)
3. usbmon2 (USB bus number 2)
4. any (Pseudo-device that captures on all interfaces)
5. lo

Arabirim Belirtme

İstenilen arabirimi dinlemek için -i arabirim_ismi parametresi ya da "-i arabirim_sıra_numarası" parametresi kullanılır.

#tshark -i eth12
veya
#tshark –i 1
gibi.

-n parametresi ile de host isimlerinin ve servis isimlerinin çözülmemesi sağlanır.

BPF Kullanımı

Tshark'da tcpdump benzeri bpf filtreleri –f " " parametresiyle ya da doğrudan parametre olarak yazılarak kullanılabilir.

root@cyblabs:~# tshark -i eth0 -f "tcp port 22"
Capturing on eth0
 0.000000 10.10.10.65 -> 10.10.10.100 SSH Encrypted request packet len=52
^C 0.005105 10.10.10.100 -> 10.10.10.65 SSH Encrypted response packet len=52
 0.023778 10.10.10.100 -> 10.10.10.65 SSH Encrypted response packet len=116
 0.024119 10.10.10.65 -> 10.10.10.100 TCP nicelink > ssh [ACK] Seq=53 Ack=169 Win=65367 Len=0
 0.704312 10.10.10.65 -> 10.10.10.100 SSH Encrypted request packet len=52
5 packets captured

Protokol Tanımlama

Tcpdump veya benzeri klasik ağ analiz araçları port bazlı izleme yapar. Mesela http için bu protokole ait öntanımlı port numarası kullanılır(80). Tshark kullanırken herhangi bir portta çalışan uygulamanın özünde hangi protokol olduğu belirtilebilir.

-d tcp.port==8080

Parametresi kullanılarak 8080 portundan akan trafiğin tipinin http olduğu belirtilir. Bu belirtimden sonra Tshark bu portta gördüğü verileri http olarak çözümlemeye çalışacaktır.

Broadcast Paketleri Yakalama

root@cyblabs:~# tshark broadcast
Running as user "root" and group "root". This could be dangerous.
Capturing on eth0
 0.000000 Fujitsu_f4:38:a8 -> Broadcast   ARP Who has 10.0.9.2? Tell 10.0.0.67
 0.000026 Fujitsu_f4:38:a8 -> Broadcast   ARP Who has 10.0.0.1? Tell 10.0.0.67
 1.989765 HewlettP_1a:3a:55 -> Broadcast   ARP Who has 10.0.2.2? Tell 10.0.0.60
 5.302770 HewlettP_1a:3a:55 -> Broadcast   ARP Who has 10.0.2.2? Tell 10.0.0.60
 2.161011 HewlettP_1a:3a:55 -> Broadcast   ARP Who has 10.0.1.2? Tell 10.0.0.60
^C7 packets captured

Ipv6 Paketleri Yakalama

Tshark kullanarak Ipv6 paketlerini yakalamak için ipv6 parametresi kullanılır.

root@cyblabs:~# tshark ip6
Capturing on eth0
0.000000 :: -> ff02::1:ff18:349c ICMPv6 Multicast listener report
0.000041 :: -> ff02::2 ICMPv6 Router solicitation
0.000053 :: -> ff02::1:ff18:349c ICMPv6 Neighbor solicitation
1.000586 :: -> ff02::1:ff18:349c ICMPv6 Multicast listener report
4.000561 fe80::224:81ff:fe18:349c -> ff02::2 ICMPv6 Router solicitation
8.000544 fe80::224:81ff:fe18:349c -> ff02::2 ICMPv6 Router solicitation

Paket Çıktılarına Ait Belirli Alanların Gösterimi

Tshark ile –T ve –e parametreleri kullanılarak yakalanan paketlere ait sadece belirli alanların ekrana basılması sağlanabilir(kaynak_ip hedef_ip port_numarası vs vs).

Bu çıktı biçimi tcpdump'da cut, awk, uniq gibi ek UNIX programları kullanarak gerçekleştirilebilir.

Örnek:

root@cyblabs:~# tshark -i eth0 -T fields -e frame -e ip.src -e ip.dst –e
Running as user "root" and group "root". This could be dangerous.
Capturing on eth0
Frame 1 (60 bytes on wire, 60 bytes captured) 10.10.10.65 10.10.10.100
Frame 2 (170 bytes on wire, 170 bytes captured) 10.10.10.100 10.10.10.65
Frame 3 (92 bytes on wire, 92 bytes captured) 10.0.72.67 10.255.255.255
Frame 4 (60 bytes on wire, 60 bytes captured) 10.10.10.65 10.10.10.100
Frame 5 (60 bytes on wire, 60 bytes captured)
Frame 6 (92 bytes on wire, 92 bytes captured) 10.0.72.67 10.255.255.255
Frame 7 (442 bytes on wire, 442 bytes captured) 10.10.10.100 10.10.10.65
Frame 8 (60 bytes on wire, 60 bytes captured) 10.10.10.65 10.10.10.100

Detaylı Paket(->Protokol) Çıktısı

Paketleri ekrandan izlerken ilgili protokole ait tüm detayları görmek için -V parametresi kullanılabilir. –V parametresinin farkını görmek için aşağıdaki örneğin incelenmesi yeterli olacaktır.

-V kullanılamayan durum:

cyblabs~# tshark -i eth0 -n udp port 53
Capturing on eth0
0.000000 10.10.10.100 -> 10.10.10.65 DNS Standard query A www.bga.com.tr

–V parametresi eklenerek aynı komut tekrar edilirse DNS protokolüne ait tüm detaylar ekranda gözükecektir.

root@cyblabs:~# tshark -i eth0 -n udp port 53 -V
...Ethernet başlık bilgileri...
... IP başlık bilgileri...

User Datagram Protocol, Src Port: 43416 (43416), Dst Port: 53 (53)
Source port: 43416 (43416)
Destination port: 53 (53)
Length: 40
Checksum: 0x28f2 [validation disabled]
[Good Checksum: False]
[Bad Checksum: False]
Domain Name System (query)
Transaction ID: 0x0498
Flags: 0x0100 (Standard query)
0... .... = Response: Message is a query
.000 0... .... = Opcode: Standard query (0)
.... ..0. .... = Truncated: Message is not truncated
....1 .... = Recursion desired: Do query recursively
.... .0.. .... = Z: reserved (0)
....0 .... = Non-authenticated data OK: Non-authenticated data is unacceptable
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
Queries
www.bga.com.tr: type A, class IN
Name: www.bga.com.tr
Type: A (Host address)
Class: IN (0x0001)

Tshark, Tcpdump Farklılıkları

tcpdump ile HTTP trafigi analizi

cyblabs~# tcpdump -i eth0 -ttttnn tcp port 80 -vv

2009-09-08 05:47:23.057285 IP (tos 0×0, ttl 64, id 28117, offset 0, flags [DF], proto TCP (6), length 52) 10.200.169.163.45196 > 64.233.169.147.80: S, ck sum 0xfdbf (correct), 906286265:906286265(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 6>

2009-09-08 05:47:23.328826 IP (tos 0×0, ttl 52, id 58449, offset 0, flags [none], proto TCP (6), length 40) 64.233.169.147.80 > 10.200.169.163.45196: F, cksum 0xa6ff (correct), 569:569(0) ack 200 win 107

Yukardaki çıktıya bakılacak olursa her bir satır bir paketi işsaret eder. Satırlar incelenirse HTTP protokolüne ait bilgi edinilemez, sadece TCP protokolüne ait bazı bilgiler elde edilebilir.

Bunun sebebi analiz için kullandığımız yazılımın(tcpdump) kısıtlı protokol analizine sahip olmasıdır(tcpdump tcp, ip, udp vs gibi alt seviye protokollere ait analiz imkanı sunar). Bizim görmek istediğimiz HTTP'e ait başlık bilgileri ise tcpdump ile görüntülenemez. Bunun için Wireshark/Tshark kullanılabilir.

Aşağıdaki çıktılar Tshark sniffer programından alınmıştır. Görüleceği üzere bir HTTP paketine ait tüm detaylar ekrana basılmaktadır.

Capturing on eth0

…
Transmission Control Protocol, Src Port: 56537 (56537), Dst Port: http (80), Seq: 1, Ack: 1, Len: 199
  Source port: 56537 (56537)
  Destination port: http (80)
  Sequence number: 1 (relative sequence number)
  [Next sequence number: 200 (relative sequence number)]
  Acknowledgement number: 1 (relative ack number)
  Header length: 20 bytes
  Flags: 0×18 (PSH, ACK)
    0… …. = Congestion Window Reduced (CWR): Not set
    .0.. …. = ECN-Echo: Not set
    ..0. …. = Urgent: Not set
    …1 …. = Acknowledgment: Set
    …. 1… = Push: Set
    …. .0.. = Reset: Not set
    …. ..0. = Syn: Not set
    …. …0 = Fin: Not set

```
    Window size: 5888 (scaled)
    Checksum: 0x9f9e [incorrect, should be 0xf736 (maybe caused by "TCP checksum
offload"?)]
        [Good Checksum: False]
        [Bad Checksum: True]
Hypertext Transfer Protocol
    GET / HTTP/1.0\r\n
        Request Method: GET
        Request URI: /
        Request Version: HTTP/1.0
    Host: www.google.com\r\n
    Accept: text/html, text/plain, text/css, text/sgml, */*;q=0.01\r\n
    Accept-Encoding: gzip, bzip2\r\n
    Accept-Language: en\r\n
    User-Agent: Lynx/2.8.6rel.4 libwww-FM/2.14\r\n
    \r\n
```

Tshark Filtreleri

Tshark aynı Wireshark'da olduğu gibi iki çeşit filtreleme özelliğine sahiptir. Bunlardan biri trafik yakalama esnasında kullanılan ve tcpdump ile hemen hemen aynı özelliklere(Berkley Paket Filter) sahip olan capture filter, diğeri de yakalanan trafik üzerinde detaylı analiz yapmaya yarayan read(Display) filter dır.

Tshark'da (read)display filterlar aynı zamanda paket yakalama esnasında da kullanılabilir. Paket yakalama esnasında read filter kullanılırsa sistem çok daha fazla CPU, RAM ihtiyaç duyar ve paket kaçırmaya başlar.

Display filter Kavramı

Display filter özelliği ile Tshark çözümleyebildiği protokollere ait tüm detayları gösterebilir ve sadece bu detaylara ait paketleri yakalamaya yardımcı olur. Mesela amaç tüm dns trafiği değil de dns trafiği içerisinde sadece www.lifeoverip.net domainine ait sorgulamaları yakalamaksa aşağıdaki gibi bir filtreleme iş görecektir.

'dns.qry.name==www.lifeoverip.net'

Display Filter için -R 'filtreleme detayı' seçeneği kullanılır.

Benzer şekilde SYN Flood DDoS saldırılarında kullanılan SYN paketlerini yakalanmak istenirse *tshark -R "tcp.flags.syn == 1 and tcp.flags.ack ==0" gibi* bir filtreleme kullanılabilir.

root@cyblabs:~# tshark -R "tcp.flags.syn == 1 and tcp.flags.ack ==0" -c 30

Capturing on eth0
 0.270441 124.244.224.98 -> 10.10.10.65 TCP res > http [**SYN**] Seq=0 Win=512 Len=0
 0.270747 246.139.122.150 -> 10.10.10.65 TCP beeyond-media > http **[SYN**] Seq=0 Win=512 Len=0
 0.271112 215.168.246.224 -> 10.10.10.65 TCP close-combat > http [**SYN**] Seq=0 Win=512 Len=0
 0.271734 151.82.40.118 -> 10.10.10.65 TCP dialogic-elmd > http [**SYN**] Seq=0 Win=512 Len=0
 0.272243 61.254.73.234 -> 10.10.10.65 TCP tekpls > http [**SYN]** Seq=0 Win=512 Len=0
 0.272414 125.115.51.184 -> 10.10.10.65 TCP hlserver > http [**SYN**] Seq=0 Win=512 Len=0
 0.272595 103.0.213.166 -> 10.10.10.65 TCP eye2eye > http [**SYN**] Seq=0 Win=512 Len=0
 0.272765 89.83.110.62 -> 10.10.10.65 TCP ismaeasdaqlive > http [**SYN**] Seq=0 Win=512 Len=0
 0.273140 163.213.112.35 -> 10.10.10.65 TCP ismaeasdaqtest > http [**SYN**] Seq=0 Win=512 Len=0
 0.273310 3.175.144.130 -> 10.10.10.65 TCP bcs-lmserver > http [**SYN**] Seq=0 Win=512 Len=0
 0.273481 199.6.209.0 -> 10.10.10.65 TCP mpnjsc > http [**SYN**] Seq=0 Win=512 Len=0
 0.273650 31.57.73.233 -> 10.10.10.65 TCP rapidbase > http [**SYN**] Seq=0 Win=512 Len=0
 0.273938 9.139.242.39 -> 10.10.10.65 TCP abr-api > http [**SYN**] Seq=0 Win=512 Len=0
 0.274115 236.225.75.83 -> 10.10.10.65 TCP abr-secure > http [**SYN**] Seq=0 Win=512 Len=0
 0.274288 207.57.128.29 -> 10.10.10.65 TCP vrtl-vmf-ds > http [SYN] Seq=0 Win=512 Len=0

22 packets captured

Display Filterlari akılda tutmak ya da ilgili protokole ait tüm detayları bilmek zordur. Bu sebeple Wireshark projesi tarafından hazırlanmış wireshark Display Filter Reference kullanılabilir.
Bu adresten ilgili protokole ait desteklenen tüm filtrelemeler incelenebilir.

Dns.filter Örneği

Display Filter Reference: Domain Name Service

Protocol field name: dns
Versions: 0.99.0 to 1.0.5

Back to Display Filter Reference

Field name	Type	Description	Versions
dns.count.add_rr	Unsigned 16-bit integer	Additional RRs	0.99.0 to 1.0.5
dns.count.answers	Unsigned 16-bit integer	Answer RRs	0.99.0 to 1.0.5
dns.count.auth_rr	Unsigned 16-bit integer	Authority RRs	0.99.0 to 1.0.5
dns.count.prerequisites	Unsigned 16-bit integer	Prerequisites	0.99.0 to 1.0.5
dns.count.queries	Unsigned 16-bit integer	Questions	0.99.0 to 1.0.5
dns.count.updates	Unsigned 16-bit integer	Updates	0.99.0 to 1.0.5
dns.count.zones	Unsigned 16-bit integer	Zones	0.99.0 to 1.0.5
dns.flags	Unsigned 16-bit integer	Flags	0.99.0 to 1.0.5
dns.flags.authenticated	Boolean	Answer authenticated	0.99.0 to 1.0.5
dns.flags.authoritative	Boolean	Authoritative	0.99.0 to 1.0.5
dns.flags.checkdisable	Boolean	Non-authenticated data OK	0.99.0 to 1.0.5
dns.flags.opcode	Unsigned 16-bit integer	Opcode	0.99.0 to 1.0.5
dns.flags.rcode	Unsigned 16-bit integer	Reply code	0.99.0 to 1.0.5
dns.flags.recavail	Boolean	Recursion available	0.99.0 to 1.0.5

Örnekler: Tshark'da Filtre Kullanımı

Alan Adına Göre DNS Paketlerini Yakalama

İçerisinde www.lifeoverip.net sorgusu geçen dns paketlerini yakalamak için:

 # tshark -i eth0 -n -R 'dns.qry.name==www.lifeoverip.net'
Running as user "root" and group "root". This could be dangerous.
Capturing on eth0
11.467730 192.168.2.23 -> 192.168.2.1 DNS Standard query A www.lifeoverip.net
13.467968 192.168.2.23 -> 192.168.2.1 DNS Standard query A www.lifeoverip.net
17.936486 192.168.2.23 -> 192.168.2.1 DNS Standard query A www.lifeoverip.net
17.938038 192.168.2.1 -> 192.168.2.23 DNS Standard query response A 80.93.212.86

CDP Paketlerinin Analizi

CDP Cisco cihazlarin kendilerini tanıtmalari/tanımalari için kullandıkları bir protokoldur. CDP paketleri multicast yayılma gösterirler ve ağda bulunan herhangi bir kişi bu paketleri dinleyerek çalışan sistemler hakkında detaylı bilgi edinebilir.

CDP ile bir Cisco sisteme ait cihazın host adresi, IP Adresi, Interface bilgileri, detaylı IOS bilgisi, platform bilgisi, VTP domain ismi vs gibi bilgiler alınabilir. CDP paketlerini Tshark kullarak analiz etmek için aşağıdaki parametreler iş görecektir.

cyblabs ~ # tshark -i eth1 -V -f "ether host 01:00:0c:cc:cc:cc"

Cisco Discovery Protocol
Version: 2
TTL: 180 seconds
Checksum: 0xd50d [incorrect, should be 0xd60b]
[Good: False]
[Bad : True]
Device ID: SMG1117N0XW(x9-User)
Type: Device ID (0×0001)
Length: 33
Device ID: SMG1117N0XW(Kx-User)
Addresses
Type: Addresses (0×0002)
Length: 17
Number of addresses: 1
IP address: x.x.x.x.
Protocol type: NLPID
Protocol length: 1
Protocol: IP
Address length: 4
IP address: x.x.x.x
Port ID: 9/11
Type: Port ID (0×0003)
Length: 8
Sent through Interface: x/11
Capabilities
Type: Capabilities (0×0004)
Length: 8
Capabilities: 0x0000002a
…. …. …. …. …. …. …. …0 = Not a Router
…. …. …. …. …. …. …. ..1. = Is a Transparent Bridge
…. …. …. …. …. …. …. .0.. = Not a Source Route Bridge
…. …. …. …. …. …. …. 1… = Is a Switch
…. …. …. …. …. …. …0 …. = Not a Host
…. …. …. …. …. …. ..1. …. = Is IGMP capable
…. …. …. …. …. …. .0.. …. = Not a Repeater
Software Version
Type: Software version (0×0005)

Length: 102
Software Version: WS-C6509-E Software, Version McpSW: 8.5(8) NmpSW: 8.5(8)
Copyright (c) 1995-2006 by Cisco Systems
Platform: WS-C6509-E
Type: Platform (0×0006)
Length: 14
Platform: WS-C6509-E
VTP Management Domain:
Type: VTP Management Domain (0×000

Mac Adresine Göre Filtreleme

cyblabs#tshark -i eth1 -R "eth.addr == 00:01:02:03:04:05"

Nmap ve Hping Taramalarını Yakalama

Herhangi bir sniffer ile bakıldığında port tarama programlarının kendilerine özgü bazı farklılılar bulunabilir. Mesela Hping paket gönderirken MSS size değerini koymaz, random ip adreslerinden SYN paketi gönder denildiğinde Window_size değerini 512, gerçek ip adresinden paket gönder denildiğinde Window_size değerini 1024 yapar. Nmap ise taramalarında değişken Window_size kullanır.

Aşağıdaki tshark çıktısı hedef sisteme synflood yapılırken alınmıştır:. Görüleceği üzere hping "—rand-source" parametresi kullanılarak syn flood denemesi yapıldığında öntanımlı olarak window_size değerini 512 o larak ayarlamaktadır. (Bu değer hping parametreleri kurcalanarak değiştirilebilir)

cyblabs# tshark -R " tcp.window_size == 512" tcp port 80

0.000000 14.166.37.62 -> 10.10.10.65 TCP syncserverssl > http [SYN] Seq=2049939788 Win=512 Len=0
1.011424 47.69.238.116 -> 10.10.10.65 TCP pxc-sapxom > http [SYN] Seq=648501585 Win=512 Len=0
2.012156 142.133.88.190 -> 10.10.10.65 TCP mpnjsomb > http [SYN] Seq=157406144 Win=512 Len=0
3.017778 91.136.212.6 -> 10.10.10.65 TCP 2682 > http [SYN] Seq=477110631 Win=512 Len=0
4.018923 62.5.13.138 -> 10.10.10.65 TCP ncdloadbalance > http [SYN] Seq=468035736 Win=512 Len=0
5.029191 46.247.152.67 -> 10.10.10.65 TCP mpnjsosv > http [SYN] Seq=769571481 Win=512 Len=0
6.030837 73.202.63.49 -> 10.10.10.65 TCP mpnjsocl > http [SYN] Seq=1127813978 Win=512 Len=0
7.038696 81.8.234.1 -> 10.10.10.65 TCP mpnjsomg > http [SYN] Seq=1775939116 Win=512 Len=0

Bu uygulamada örnek olarak seçilen 512 byte'lık Window_size değerini başka uygulamalar da kullanıyor olabilir. Bu sebeple hping paketlerini engellemek için window_size=512 olan paketleri engelle gibi bir kural yazmak risklidir.

HTTP GET Flood Paketlerini Analiz Etme

Tshark'ın filtreleme özellikleri kullanılarak web sayfanıza gelen paketler arasındaki anormallikleri farkedilebilir. Aşağıdaki örnek http Get flood yapılan bir sistem üzerinde alınan paketlerin tshark ile analiz edilmesini göstermektedir.

Komutun açıklaması: http isteklerini ip adresi, istek tipi ve istenen URL şeklinde göster:

root@cyblabs:~# tshark -R http.request -T fields -e ip.src -e http.request.method -e http.host -e http.request.uri

Capturing on eth0
10.10.10.87 GET 10.10.10.65 /ddos_test.php
10.10.10.67 GET 10.10.10.65 /ddos_test.php
10.10.10.63 GET 10.10.10.65 /ddos_test.php
10.10.10.22 GET 10.10.10.65 /ddos_test.php
10.10.10.23 GET 10.10.10.65 /ddos_test.php
10.10.10.11 GET 10.10.10.65 /ddos_test.php
10.10.10.100 GET 10.10.10.65 /ddos_test.php
10.10.10.100 GET 10.10.10.65 /ddos_test.php
10.10.10.100 GET 10.10.10.65 /ddos_test.php
10.10.10.100 GET 10.10.10.65 /ddos_test.php
10.10.10.90 GET 10.10.10.65 /ddos_test.php
10.10.10.90 GET 10.10.10.65 /ddos_test.php
10.10.10.90 GET 10.10.10.65 /ddos_test.php
10.10.0.10 GET 10.10.10.65 /ddos_test.php
10.10.10.1 GET 10.10.10.65 /ddos_test.php
10.10.10.1 GET 10.10.10.65 /ddos_test.php
10.10.10.1 GET 10.10.10.65 /ddos_test.php
10.10.10.1 GET 10.10.10.65 /ddos_test.php
10.10.10.1 GET 10.10.10.65 /ddos_test.php
10.10.10.1 GET 10.10.10.65 /ddos_test.php

Bir sonraki aşama bu istekler arasında belirli eşik değerini(1000 http isteğinden fazla gibi) aşan ip adreslerinin bulunması olmalıdır.

Mysql Sorgularını Yakalama

Günümüz güvenlik dünyasının en önemli maddelerindne biri de veritabanlarına yapılan sorgulamaların merkezi olarak loglanması ve saklanmasıdır. Bunun için temelde iki yöntem tercih edilmektedir:

İlk yöntem veritabanı üzerinde -özelliğine göre- çeşitli fonksiyonları aktif hale getirerek tüm işlemlerin veritabanı tarafından loglanması, diğer yöntem de ağ üzerinden veritabanına giden

trafiğin bir kopyasını sniffer aracılığıyla yakalayarak paketler arasından ilgili sorgu/cevapları yakalama.

Tshark, mysql veritabanına ait tüm özellikleri desteklediği için ağ üzerinden mysql sunuculara giden trafiği dinleyerek trafikten yapılan sorgulamaları, cevapları ekrana/dosyaya basabilir.

Bunun için aşağıdaki gibi bir tshark komutu iş görecektir.

cyblabs# tshark -i eth0 -f 'tcp port 3306' -T fields -e mysql.query |uniq

select @@version_comment limit 1

show databases

SELECT DATABASE()

show databases

show tables

...
Mysql ile kullanılabilecek diğer parametler için
http://www.wireshark.org/docs/dfref/m/mysql.html adresine gözatılabilir.

HTTP Trafiği İçerisinde GET, PUT ve OPTIONS Kullanılan İstekleri Yakalama.

cyblabs#tshark -i eth0 -n -R 'http.request.method contains GET or http.request.method contains PUT or http.request.method contains OPTIONS'
Capturing on eth0
7.571543 192.168.2.22 -> 80.93.212.86 HTTP OPTIONS / HTTP/1.111
14.925700 192.168.2.22 -> 80.93.212.86 HTTP GET / HTRTP/1.1

Bir TCP Bağlantısına Ait Başlangıç Ve Bitiş Paketlerini Yakalama

İçerisinde SYN veya FIN bayrağı set edilmiş paketleri yakalamak için

 # tshark -n -R 'tcp.port==80 and tcp.flags.fin==1 or tcp.flags.syn==1'
Running as user "root" and group "root". This could be dangerous.
Capturing on eth0
1.245831 192.168.2.22 -> 80.93.212.86 TCP 36566 > 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=2759271 TSER=0 WS=5

1.259797 80.93.212.86 -> 192.168.2.22 TCP 80 > 36566 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1452 WS=1 TSV=2754203455 TSER=2759271
3.966800 80.93.212.86 -> 192.168.2.22 TCP 80 > 36566 **[FIN, ACK**] Seq=212 Ack=11 Win=66240 Len=0 TSV=2754206160 TSER=2759947
3.966919 192.168.2.22 -> 80.93.212.86 TCP 36566 > 80 [FIN, ACK] Seq=11 Ack=213 Win=6912 Len=0 TSV=2759952 TSER=2754206160,

Filtrelemelerde kullanılacak operatörler(==, !=, contains, vs) için www.wireshark.org/docs/dfref/ adresi incelenebilir.

Trafik İstatistikleri

Tshark kullanırken yakalanan/kaydedilen trafik hakkında istatistiki bilgi almak gerekirse başvurulacak komut satırı parametresi –z olacaktır.
-z parametresinin yanına uygun değerler vererek bir trafik hakkında oldukça geniş yelpazede istatistiki çıktı alınabilir.

 # tshark -r bga_Test.pcap -q -z io,phs

```
===================================================================
Protocol Hierarchy Statistics
Filter: frame

frame                                    frames:1743 bytes:702358
  eth                                    frames:1743 bytes:702358
        ip                               frames:1499 bytes:687442
        tcp                              frames:1495 bytes:687066
        ssh                              frames:392 bytes:330492
        short                            frames:332 bytes:325324
        http                             frames:153 bytes:161659
        short                            frames:150 bytes:161396
        ssl                              frames:208 bytes:148619
        short                            frames:151 bytes:107046
        unreassembled                          frames:39 bytes:39903
        igmp                             frames:2 bytes:120
        udp                              frames:2 bytes:256
        snmp                             frames:2 bytes:256
        short                            frames:2 bytes:256
        arp                              frames:204 bytes:12240
        llc                              frames:37 bytes:2220
```

```
stp                     frames:37 bytes:2220
lldp                    frames:3 bytes:456
short                   frames:3 bytes:456
================================================================
```

Tshark Kullanarak Wireshark İle Uzak Sistemlerde Paket Analizi

GUI'si olmayan sistemlerde tshark&Wireshark kullanarak uzaktaki sisteme gelen paketleri yerel bilgisayarımızdaki Wireshark'a aktarabiliriz.

cyblabs#wireshark -k -i < (ssh -l root IP_adresi /usr/sbin/tshark -i eth0 -w - tcp port 80)

Bu komutla uzak sistemde çalışan Tshark'ın yakaladığı paketleri anlık olarak yerel bilgisayardaki Wireshark'a göndermesi sağlanmaktadır.
Uzak sistemlerdeki paketleri yereldeki Wireshark'a aktarmak için denenebilecek alternatif yöntemler http://wiki.wireshark.org/CaptureSetup/Pipes adresinden okunabilir.