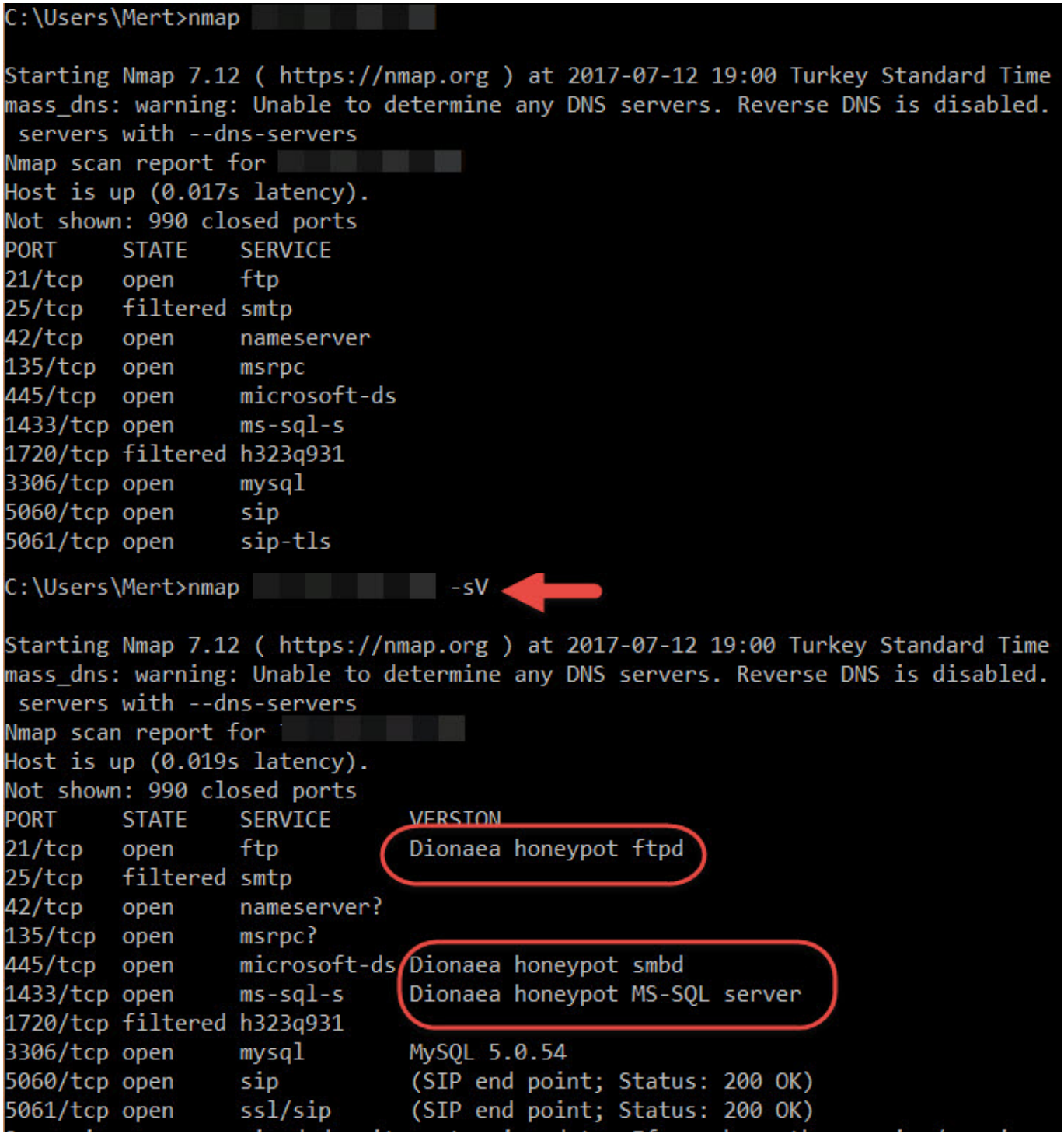
Balküpü Tespiti

written by Mert SARICA | 3 September 2018
If you are looking for an English version of this article, please visit here.

Yaklaşık bir yıl önce, Tuzak Sistem ile Hacker Avı çalışmamın planlarını yaparken, düşük etkileşimli mi yoksa yüksek etkileşimli mi (high interaction) bir balküpü sistemi kullanmalıyım ikileminde kalmıştım. Aralarındaki temel farka bakıldığında, düşük etkileşimli olan gerçek bir sistemi, servisi simüle ettiği için kurulumu, yönetimi ve güvenliğini sağlamak görece daha kolay diyebiliriz. Yüksek etkileşimli olana bakıldığında ise işin içinde gerçek, canlı bir sistem olduğu için kurulumu ve yönetimi zahmetliyken, güvenliğini sağlamak ise izolasyon sebebiyle bir o kadar zorlaşıyor.

Yönetim gözüyle değerlendirdiğinizde düşük etkileşimli balküpü sistemlerinin kullanımı kulağa çok daha pratik gelebiliyor olsa da balküpü sistemlerini kullanmanın ana amacı, siber saldırganları bu sistemlere çekerek kullandıkları taktikleri, teknikleri ve prosedürleri (TTP) öğrenmek olduğu için yüksek etkileşimli sistemlerin saldırganlar tarafından tespit edilmesi pratikte çok daha zor olabiliyor. 6 ay boyunca çalışan tuzak sistemimi hackleyen onlarca siber saldırganın davranışlarını izlediğimde çoğu siber saldırgan, sistemin tuzak bir sistem olma ihtimaline karşı özel kontroller gerçekleştirmemişti dolayısıyla yüksek etkileşimli yerel balküpü sistemlerini sıkılaştırmak için çok da çaba sarfetmenize gerek kalmayabilir.

Bakıldığında düşük etkileşimli balküpü sistemlerini tespit etmek için saldırganların Nmap aracı ile bir basit bir tarama gerçekleştirilmeleri yeterli olabiliyor bu nedenle balküpü sistemini canlı sistemlerin yanına yerleştirmeden önce tanınmaz hale getirmek kullanan bireyler ve kurumlar için büyük önem taşıyor. Kimi zaman siber saldırganlardan önce USOM, zafiyet barındırdığı gerekçesi ile internet servis sağlayıcısı ile bu sistem özelinde iletişime de geçebiliyor. :)


C:\Users\Mert>nmap
Starting Nmap 7.12 (https://nmap.org) at 2017-07-12 19:00 Turkey Standard Time
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
servers with --dns-servers
Nmap scan report for
Host is up (0.017s latency).
Not shown: 990 closed ports
PORT STATE SERVICE
21/tcp open ftp
25/tcp filtered smtp
42/tcp open nameserver
135/tcp open msrpc
445/tcp open microsoft-ds
1433/tcp open ms-sql-s
1720/tcp filtered h323q931
3306/tcp open mysql
5060/tcp open sip
5061/tcp open sip-tls
C:\Users\Mert>nmap -sV
Starting Nmap 7.12 (https://nmap.org) at 2017-07-12 19:00 Turkey Standard Time
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
servers with --dns-servers
Nmap scan report for
Host is up (0.019s latency).
Not shown: 990 closed ports
PORT STATE SERVICE VERSION
21/tcp open ftp Dionaea honeypot ftpd
25/tcp filtered smtp
42/tcp open nameserver?
135/tcp open msrpc?
445/tcp open microsoft-ds Dionaea honeypot smbd
1433/tcp open ms-sql-s Dionaea honeypot MS-SQL server
1720/tcp filtered h323q931
3306/tcp open mysql MySQL 5.0.54
5060/tcp open sip (SIP end point; Status: 200 OK)
5061/tcp open ssl/sip (SIP end point; Status: 200 OK)

Ulusal Siber Olaylara Müdahale Merkezi (USOM) üzerinden [redacted] ip adresli bir sunucunuz üzerinde SMB servisi ile ilgili güvenlik zafiyeti tespit edildiğine dair bildirimde bulunmuştur.

16.03.2017 - MS17-010 - CVE-2017-0143-0148

25.09.2010 - MS10-061 - CVE-2010-2729

08.09.2009 - MS09-001 - CVE-2009-3103

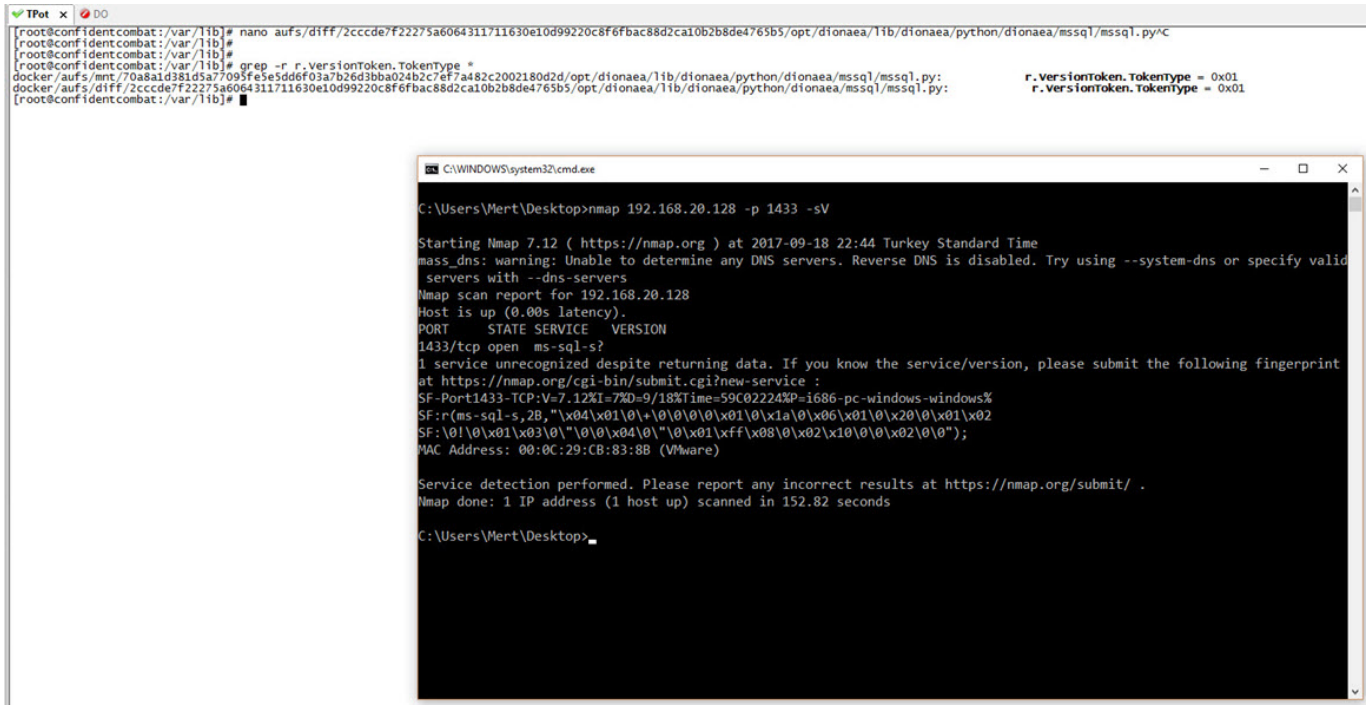
25.09.2008 - MS08-067 - CVE-2008-4250

Konu ile ilgili microsoft bülteninin adresi aşağıdaki gibidir.

https://technet.microsoft.com/tr-tr/library/security/ms08-067.aspx

Konu ile ilgili gerekli müdahaleleri gerçekleştirdikten sonra bilgi vermenizi rica ederiz.

Balküpü sistemi denilince çoğu kişinin aklına Dionaea gelecektir. Dionaea yukarıdaki ekran görüntüsünden de görüleceği üzere varsayılan (default) olarak kurulduğunda Nmap tarafından kolaylıkla tespit edilebilmektedir. Dionaea'yı tanınmaz hale getirmek için ise internette ufak bir araştırma yaptığınızda eskiden yeniye çok sayıda kaynağa (#1, #2, #3) ulaşabiliyorsunuz. Örneğin MSSQL servisini simüle eden Dionaea'yı tanınmaz hale getirmek için /dionaea/mssql/mssql.py dosyasındaki r.VersionToken.TokenType parametresinin 0x00 olan değerini 0x01 yaptığınızda Nmap artık 1433. bağlantı noktasında çalışan Dionaea'yı tespit edememektedir. Dionaea, zafiyet barındıran servisleri simüle ettiği (low interaction) için bu kaynaklarda yer alan bilgiler ışığında tanınmaz hale gelen Dionaea'yı siber saldırgan gözüyle tespit etmek pratikte aslında ne kadar kolay bunu araştırmaya karar verdim.



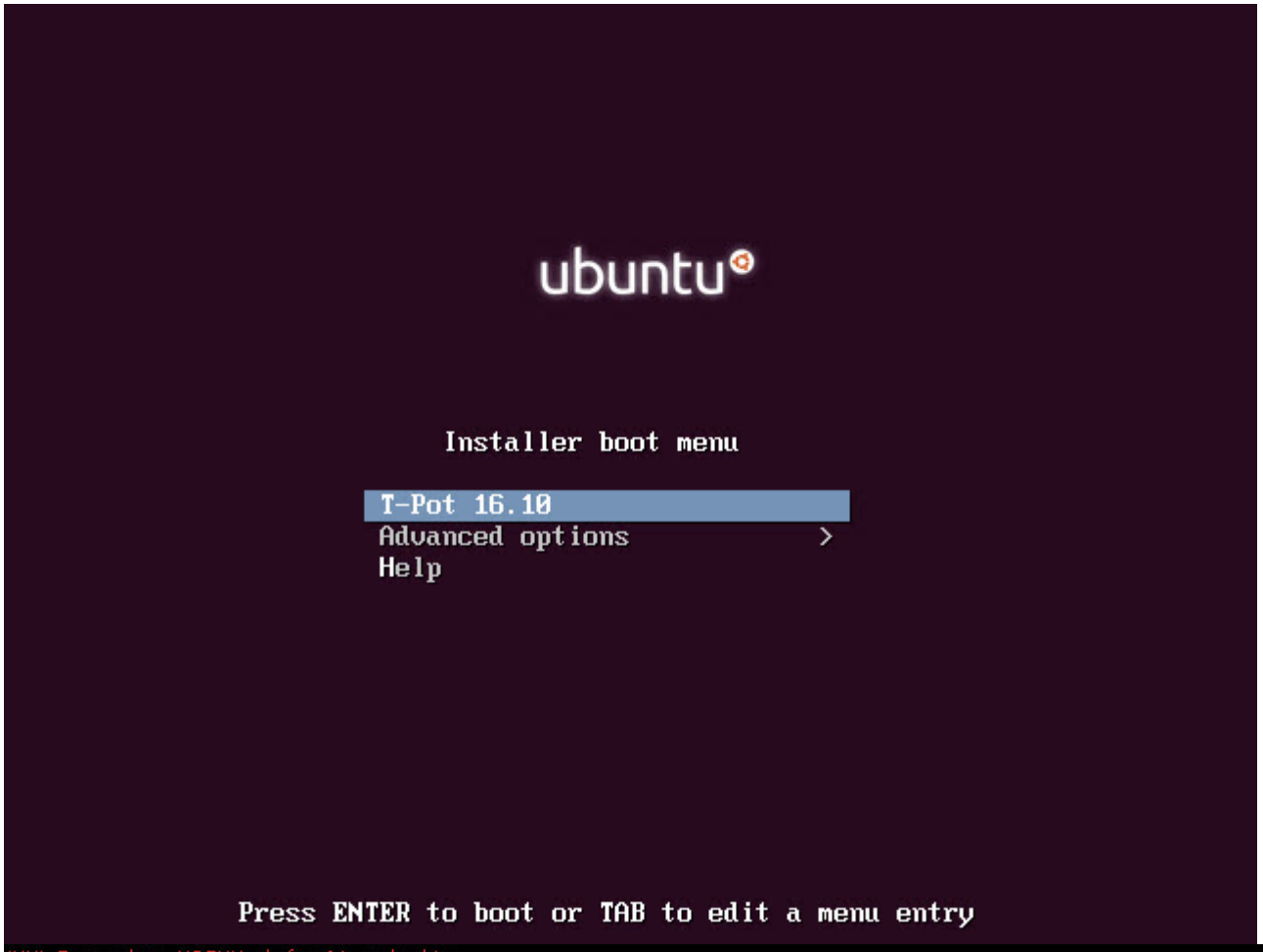
Dionaea'yı kurmakla zaman kaybetmemek için Deutsche Telekom tarafından geliştirilen ve üzerinde Dionaea da dahil olmak üzere çok sayıda balküpü sistemi bulunduran T-Pot balküpü sanal sistemini kurdum. Dionaea gibi ufacık tefecik bir balküpü sisteminin MSSQL servisini (TDS protokolü) tam anlamıyla simüle edemeyeceğini düşünerek işe 1433. bağlantı noktasından başlamaya karar verdim.

System Placement

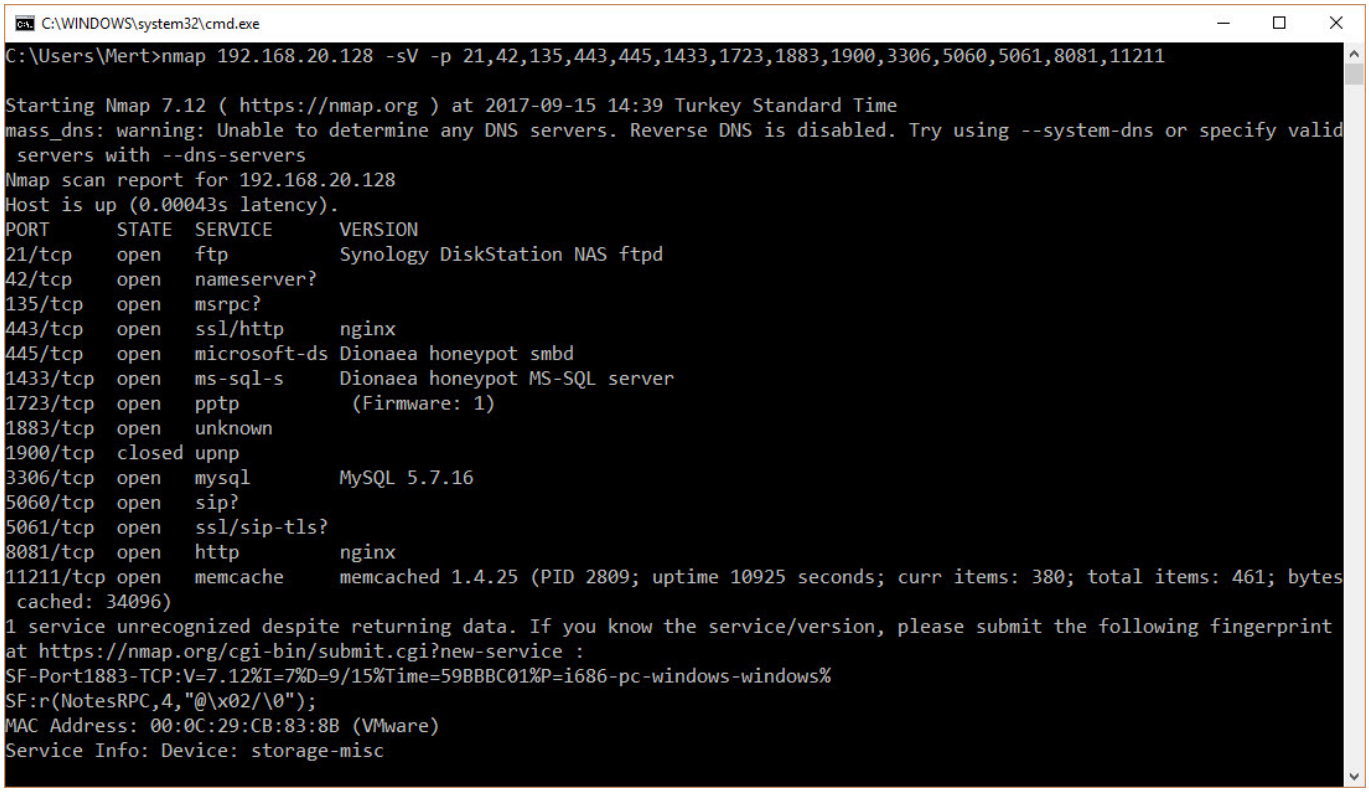
Make sure your system is reachable through the internet. Otherwise it will not capture any attacks, other than the ones from your hostile internal network! We recommend you put it in an unfiltered zone, where all TCP and UDP traffic is forwarded to T-Pot's network interface.

If you are behind a NAT gateway (e.g. home router), here is a list of ports that should be forwarded to T-Pot.

Honeypot	Transport	Forwarded ports
conpot	TCP	1025, 50100
cowrie	TCP	22, 23
dionaea	TCP	21, 42, 135, 443, 445, 1433, 1723, 1883, 1900, 3306, 5060, 5061, 8081, 11211
dionaea	UDP	69, 5060
elasticpot	TCP	9200
emobility	TCP	8080
glastopf	TCP	80
honeytrap	TCP	25, 110, 139, 3389, 4444, 4899, 5900, 21000

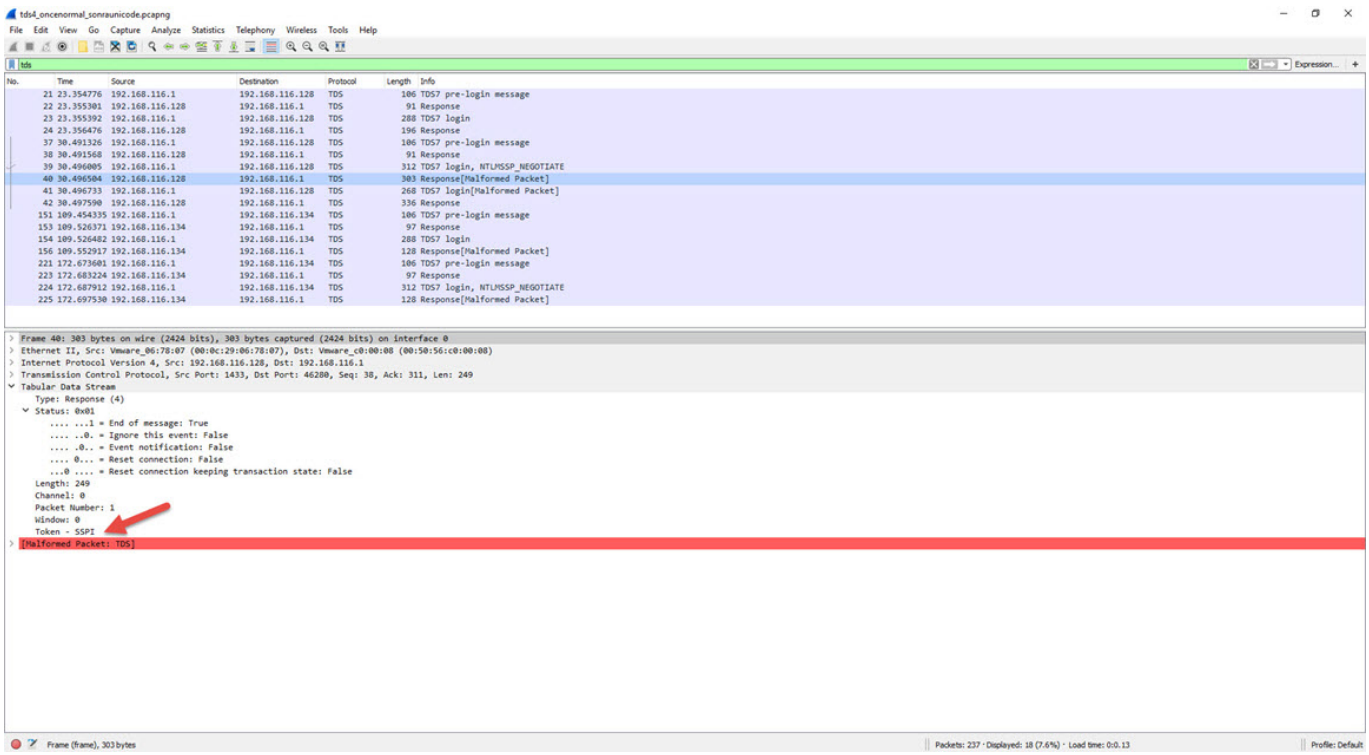



```
### Removing NGINX default website.
### Waiting a few seconds to avoid interference with service messages.
### Please choose your install type and notice HW recommendation.

    [T] - T-Pot Standard Installation
          - Cowrie, Dionaea, Elasticpot, Glastopf, Honeytrap, Suricata & ELK
          - 4 GB RAM (6-8 GB recommended)
          - 64GB disk (128 GB SSD recommended)

    [H] - Honeypots Only Installation
          - Cowrie, Dionaea, ElasticPot, Glastopf & Honeytrap
          - 3 GB RAM (4-6 GB recommended)
          - 64 GB disk (64 GB SSD recommended)

    [I] - Industrial
          - ConPot, eMobility, ELK & Suricata
          - 4 GB RAM (8 GB recommended)
          - 64 GB disk (128 GB SSD recommended)

    [E] - Everything
          - All of the above
          - 8 GB RAM
          - 128 GB disk or larger (128 GB SSD or larger recommended)

Install Type:
```

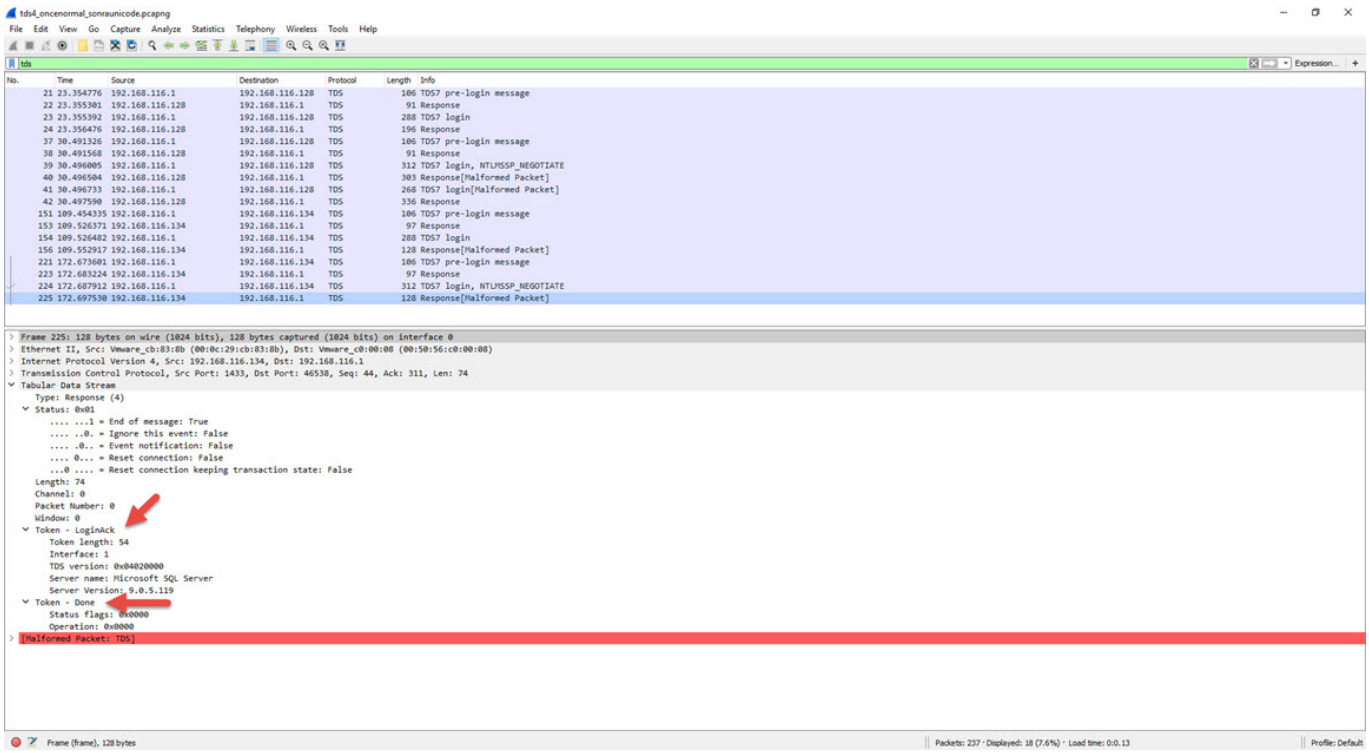


Bir MSSQL sunucusu ile bir istemcinin uygulama seviyesinde haberleşebilmesi için TDS (Tabular Data Stream) protokolünü kullanması gerekmektedir. TDS protokolü ile MSSQL sunucusuna giriş (login) yapabilmek için ezelden beri mümkün olan iki tür giriş yöntemi bulunmaktadır. Birincisi kullanıcı adı ve parola ile giriş, ikincisi ise windows doğrulama (NTLM) yöntemi ile giriştir. Normal şartlarda TDS protokolüne göre kullanıcı adı ve parola ile giriş yapmaya çalıştığınızda MSSQL sunucusundan gelen yanıtta, jeton olarak (token) LOGINACK_TOKEN (0xAD), windows doğrulama yöntemi ile giriş yapmaya çalıştığınızda ise jeton olarak SSPI TOKEN (0xED) olması gelmesi gerekmektedir ancak Dionaea her iki istek için de aynı sonucu dönmektedir. :)

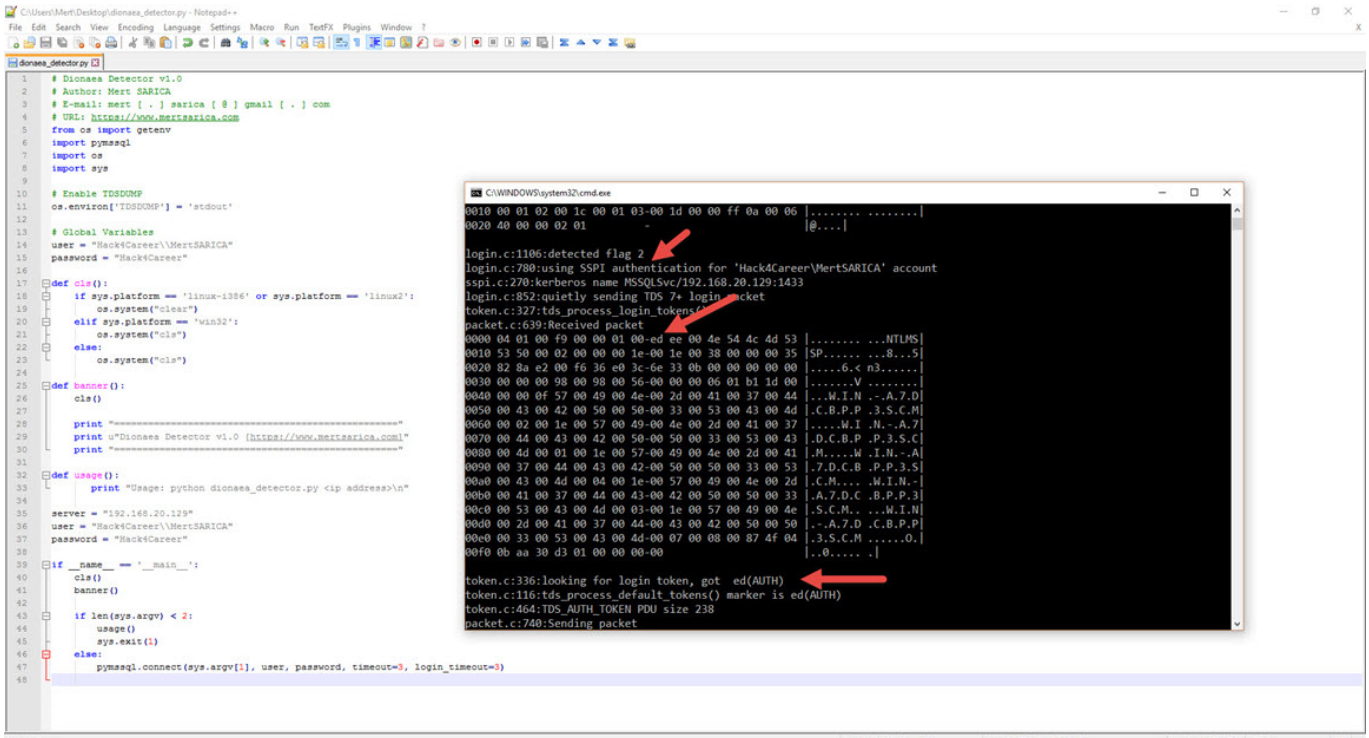
Windows doğrulama isteğine Microsoft SQL 2008 Server Express sürümünden dönen yanıt

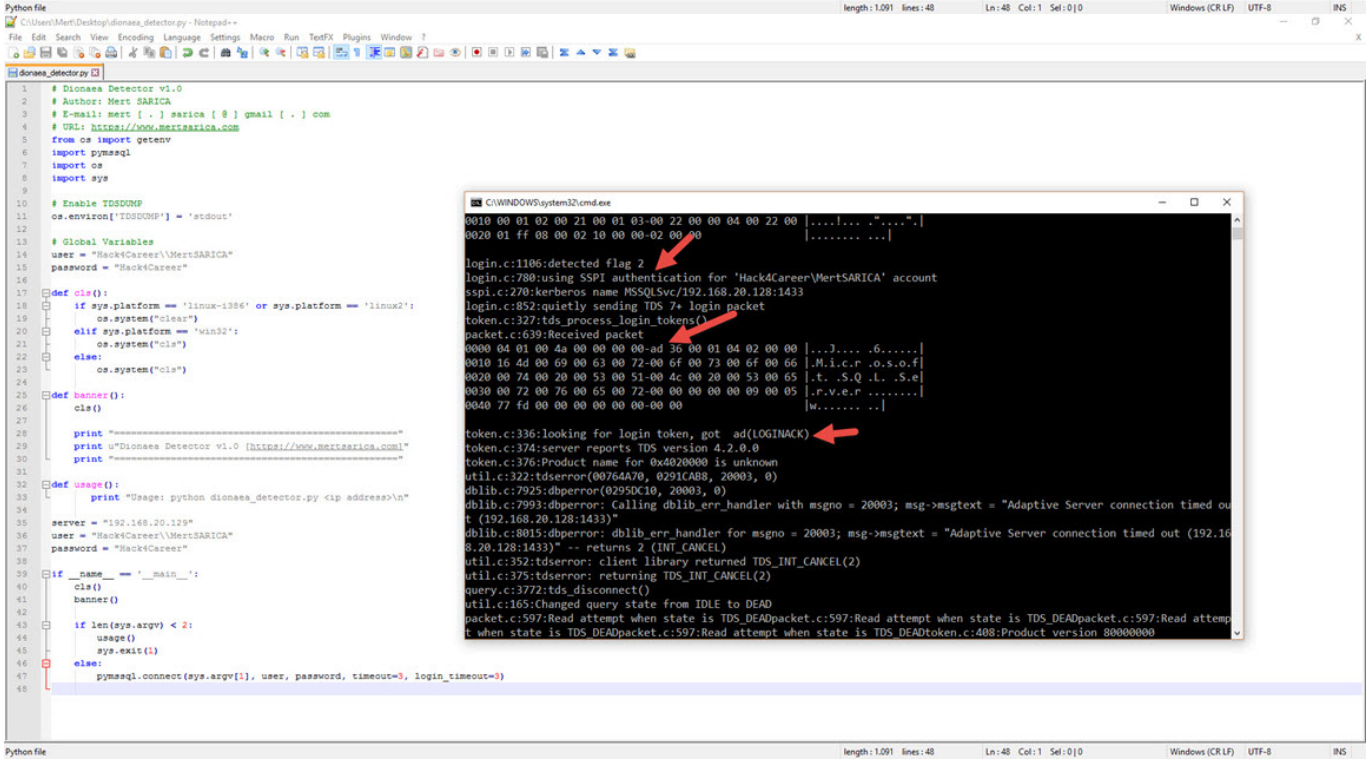


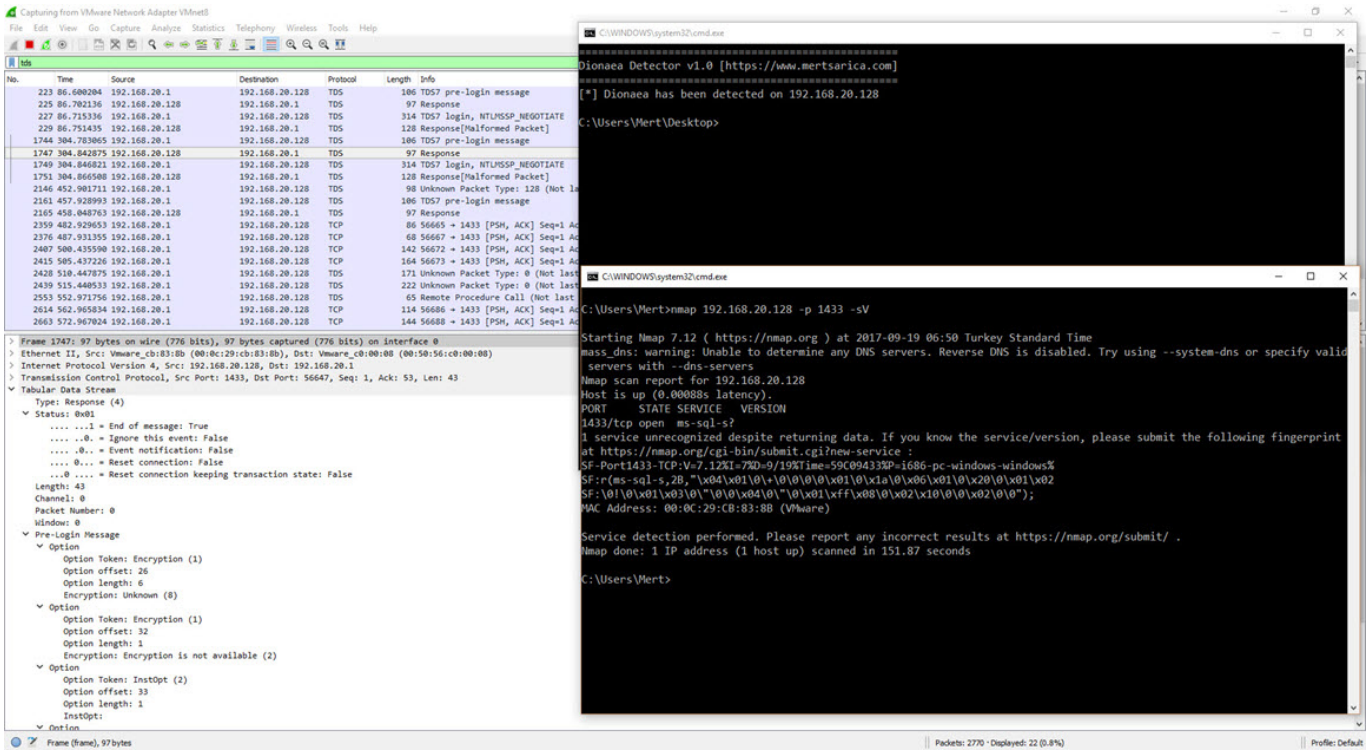
Windows doğrulama isteğine Dionaea'dan dönen yanıt



Durum böyle olunca Python ile pymssql kütüphanesinden de faydalanarak hızlıca bu farkı tespit edebilen dionaea_detector.py adında basit bir araç hazırladım. Bu araç sayesinde Nmap'in tespit edemediği Dionaea bal küpü sistemini basit bir kontrol ile tespit ederek, art niyetli kişilerin pratikte bunu ne kadar basit bir şekilde tespit edebileceklerini öğrenmiş oldum.


C:\Users\Mert\Desktop\dionaea_detector.py - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
dionaea_detector.py
# Dionaea Detector v1.0
# Author: Mert SARICA
# E-mail: mert [.] sarica [@] gmail [.] com
# URL: https://www.mertsarica.com
from os import getenv
import pymssql
import os
import sys
# Enable TDSDUMP
os.environ['TDSDUMP'] = 'stdout'
# Global Variables
user = "Hack4Career\\MertSARICA"
password = "Hack4Career"
def cls():
if sys.platform == 'linux-i386' or sys.platform == 'linux2':
os.system("clear")
elif sys.platform == 'win32':
os.system("cls")
else:
os.system("cls")
def banner():
cls()
print "=="
print u"Dionaea Detector v1.0 (https://www.mertsarica.com)"
print "=="
def usage():
print "Usage: python dionaea_detector.py <ip address>\n"
server = "192.168.20.129"
user = "Hack4Career\\MertSARICA"
password = "Hack4Career"
if __name__ == '__main__':
cls()
banner()
if len(sys.argv) < 2:
usage()
sys.exit(1)
else:
pymssql.connect(sys.argv[1], user, password, timeout=3, login_timeout=3)
C:\WINDOWS\system32\cmd.exe
0010 00 01 02 00 1c 00 01 03-00 1d 00 00 ff 0a 00 06 |........ ....|
0020 40 00 00 02 01 @....|
login.c:1106:detected flag 2
login.c:780:using SSPI authentication for 'Hack4Career\MertSARICA' account
sspi.c:270:kerberos name MSSQLSvc/192.168.20.129:1433
login.c:852:quietly sending TDS 7+ login packet
token.c:327:tds_process_login_tokens()
packet.c:639:Received packet
0000 04 01 00 f9 00 00 01 00-ed ee 00 4e 54 4c 4d 53 |........ ...NTLMS|
0010 53 50 00 02 00 00 00 1e-00 1e 00 38 00 00 00 35 |SP..... ...8...5|
0020 82 8a e2 00 f6 36 e0 3c-6e 33 0b 00 00 00 00 00 |.....6.< n3......|
0030 00 00 00 98 00 98 00 56-00 00 00 06 01 b1 1d 00 |.......V ....|
0040 00 00 0f 57 00 49 00 4e-00 2d 00 41 00 37 00 44 |...W.I.N .-.A.7.D|
0050 00 43 00 42 00 50 00 50-00 33 00 53 00 43 00 4d |.C.B.P.P .3.S.C.M|
0060 00 02 00 1e 00 57 00 49-00 4e 00 2d 00 41 00 37 |.....W.I .N.-.A.7|
0070 00 44 00 43 00 42 00 50-00 50 00 33 00 53 00 43 |.D.C.B.P .P.3.S.C|
0080 00 4d 00 01 00 1e 00 57-00 49 00 4e 00 2d 00 41 |.M.....W .I.N.-.A|
0090 00 37 00 44 00 43 00 42-00 50 00 50 00 33 00 53 |.7.D.C.B .P.P.3.S|
00a0 00 43 00 4d 00 04 00 1e-00 57 00 49 00 4e 00 2d |.C.M.... .W.I.N.-|
00b0 00 41 00 37 00 44 00 43-00 42 00 50 00 50 00 33 |.A.7.D.C .B.P.P.3|
00c0 00 53 00 43 00 4d 00 03-00 1e 00 57 00 49 00 4e |.S.C.M.. ...W.I.N|
00d0 00 2d 00 41 00 37 00 44-00 43 00 42 00 50 00 50 |.-.A.7.D .C.B.P.P|
00e0 00 33 00 53 00 43 00 4d-00 07 00 08 00 87 4f 04 |.3.S.C.M ......O.|
00f0 0b aa 30 d3 01 00 00 00-00 |..0..... .|
token.c:336:looking for login token, got ed(AUTH)
token.c:116:tds_process_default_tokens() marker is ed(AUTH)
token.c:464:TDS_AUTH_TOKEN PDU size 238
packet.c:740:Sending packet
Python file
length : 1.091 lines : 48
Ln : 48 Col : 1 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS


C:\Users\Mert\Desktop\dionaea_detector.py - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
dionaea_detector.py
# Dionaea Detector v1.0
# Author: Mert SARICA
# E-mail: mert [.] sarica [@] gmail [.] com
# URL: https://www.mertsarica.com
from os import getenv
import pymssql
import os
import sys
# Enable TDSDUMP
os.environ['TDSDUMP'] = 'stdout'
# Global Variables
user = "Hack4Career\\MertSARICA"
password = "Hack4Career"
def cls():
if sys.platform == 'linux-i386' or sys.platform == 'linux2':
os.system("clear")
elif sys.platform == 'win32':
os.system("cls")
else:
os.system("cls")
def banner():
cls()
print "=="
print u"Dionaea Detector v1.0 (https://www.mertsarica.com)"
print "=="
def usage():
print "Usage: python dionaea_detector.py <ip address>\n"
server = "192.168.20.129"
user = "Hack4Career\\MertSARICA"
password = "Hack4Career"
if __name__ == '__main__':
cls()
banner()
if len(sys.argv) < 2:
usage()
sys.exit(1)
else:
pymssql.connect(sys.argv[1], user, password, timeout=3, login_timeout=3)
C:\WINDOWS\system32\cmd.exe
0010 00 01 02 00 21 00 01 03-00 22 00 00 04 00 22 00 |....!... ."....".|
0020 01 ff 08 00 02 10 00 00-02 00 00 |........ ...|
login.c:1106:detected flag 2
login.c:780:using SSPI authentication for 'Hack4Career\MertSARICA' account
sspi.c:270:kerberos name MSSQLSvc/192.168.20.128:1433
login.c:852:quietly sending TDS 7+ login packet
token.c:327:tds_process_login_tokens()
packet.c:639:Received packet
0000 04 01 00 4a 00 00 00 00-ad 36 00 01 04 02 00 00 |...J.... .6......|
0010 16 4d 00 69 00 63 00 72-00 6f 00 73 00 6f 00 66 |.M.i.c.r .o.s.o.f|
0020 00 74 00 20 00 53 00 51-00 4c 00 20 00 53 00 65 |.t. .S.Q .L. .S.e|
0030 00 72 00 76 00 65 00 72-00 00 00 00 00 09 00 05 |.r.v.e.r ....|
0040 77 fd 00 00 00 00 00 00-00 00 |w..... ..|
token.c:336:looking for login token, got ad(LOGINACK)
token.c:374:server reports TDS version 4.2.0.0
token.c:376:Product name for 0x4020000 is unknown
util.c:322:tdserror(00764A70, 0291CAB8, 20003, 0)
dblib.c:7925:dbperror(0295DC10, 20003, 0)
dblib.c:7993:dbperror: Calling dblib_err_handler with msgno = 20003; msg->msgtext = "Adaptive Server connection timed out (192.168.20.128:1433)"
dblib.c:8015:dbperror: dblib_err_handler for msgno = 20003; msg->msgtext = "Adaptive Server connection timed out (192.168.20.128:1433)" -- returns 2 (INT_CANCEL)
util.c:352:tdserror: client library returned TDS_INT_CANCEL(2)
util.c:375:tdserror: returning TDS_INT_CANCEL(2)
query.c:3772:tds_disconnect()
util.c:165:Changed query state from IDLE to DEAD
packet.c:597:Read attempt when state is TDS_DEADpacket.c:597:Read attempt when state is TDS_DEADpacket.c:597:Read attempt when state is TDS_DEADpacket.c:597:Read attempt when state is TDS_DEADtoken.c:408:Product version 80000000
Python file
length : 1.091 lines : 48
Ln : 48 Col : 1 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS



Kısadan hisse, balküpü sistemi kullanmadan önce yüksek ve düşük etkileşimli balkü sistemlerinin artılarını ve eksilerini baştan sona değerlendirip, siber saldırganlar tarafından tespit edilmesi zor olanını tercih etmeniz veya zorlaştırma adına mevcut sistemler üzerinde değişiklikler yapmanız sizin veya kurumunuzun yararına olacaktır.

Bir sonraki yazıda görüşmek dileğiyle herkese güvenli günler dilerim.

Not: Ekran görüntüleri T-Pot 16.10 sürümüne ait olsa da, dionaea_detector.py aracının T-Pot'un son sürümü olan 17.10 sürümü ile gelen Dionaea'yı da başarıyla tespit edebildiği teyit edilmiştir.