Honeypot Detection

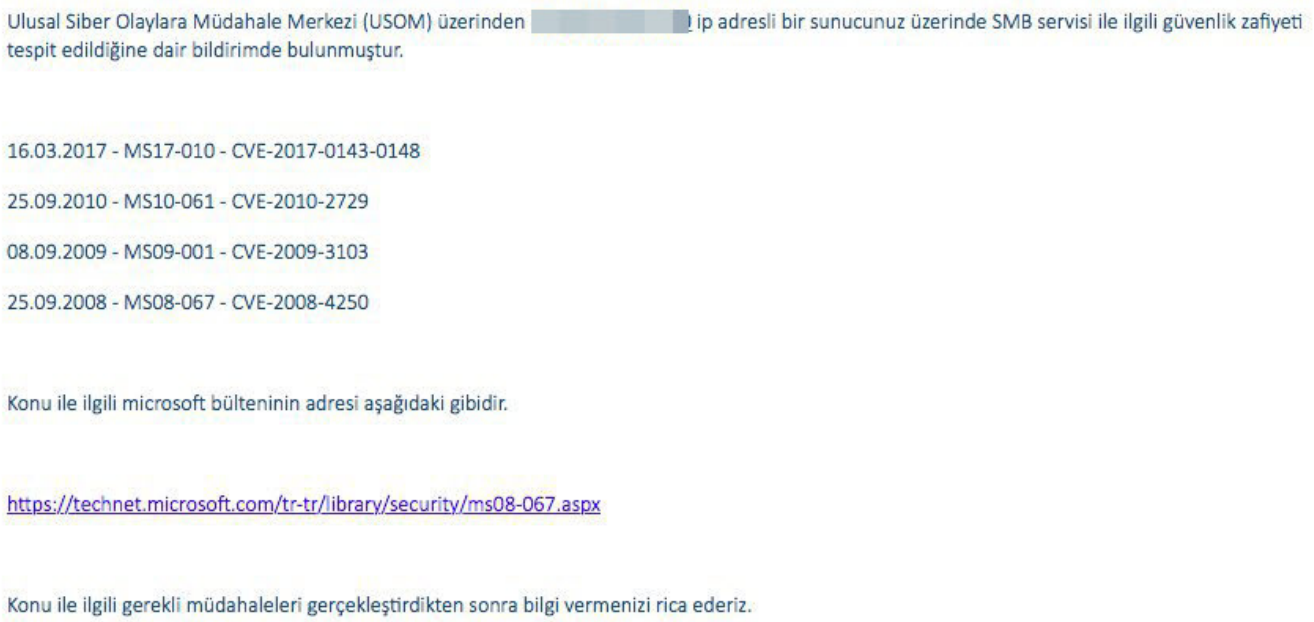
written by Mert SARICA | 3 September 2018

About a year ago, when I was planning my Hacker Hunt with a Deception System project, I was faced with the dilemma of whether to use a low-interaction or high-interaction honeypot system. When looking at the fundamental difference between them, we can say that a low-interaction honeypot, which simulates a real system or service, is relatively easier to set up, manage, and secure. On the other hand, a high-interaction honeypot involves a real, live system, making its installation, management, and security more challenging due to isolation.

From a management perspective, the use of low-interaction honeypot systems may sound more practical. However, the main purpose of using honeypots is to attract and learn about the tactics, techniques, and procedures (TTP) used by cyber attackers. In practice, it can be much more difficult for attackers to detect high-interaction honeypots. When I observed the behaviors of numerous cyber attackers who attempted to hack my honeypot system for six months, most of them did not perform specific checks to determine if the system was a trap. Therefore, you may not need to exert much effort to harden high-interaction local honeypot systems.

Indeed, when it comes to detecting low-interaction honeypot systems, attackers can often perform a simple scan using tools like Nmap. This is why it is crucial for individuals and organizations that use honeypots to make them appear undetectable before placing them alongside live systems. In some cases, even before cyber attackers, the National Cybersecurity Intervention Center (USOM) may contact the internet service provider regarding this system, citing its vulnerability. :)

```
C:\Users\Mert>nmap [redacted]

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 [redacted]
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 [redacted] -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 [redacted]
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

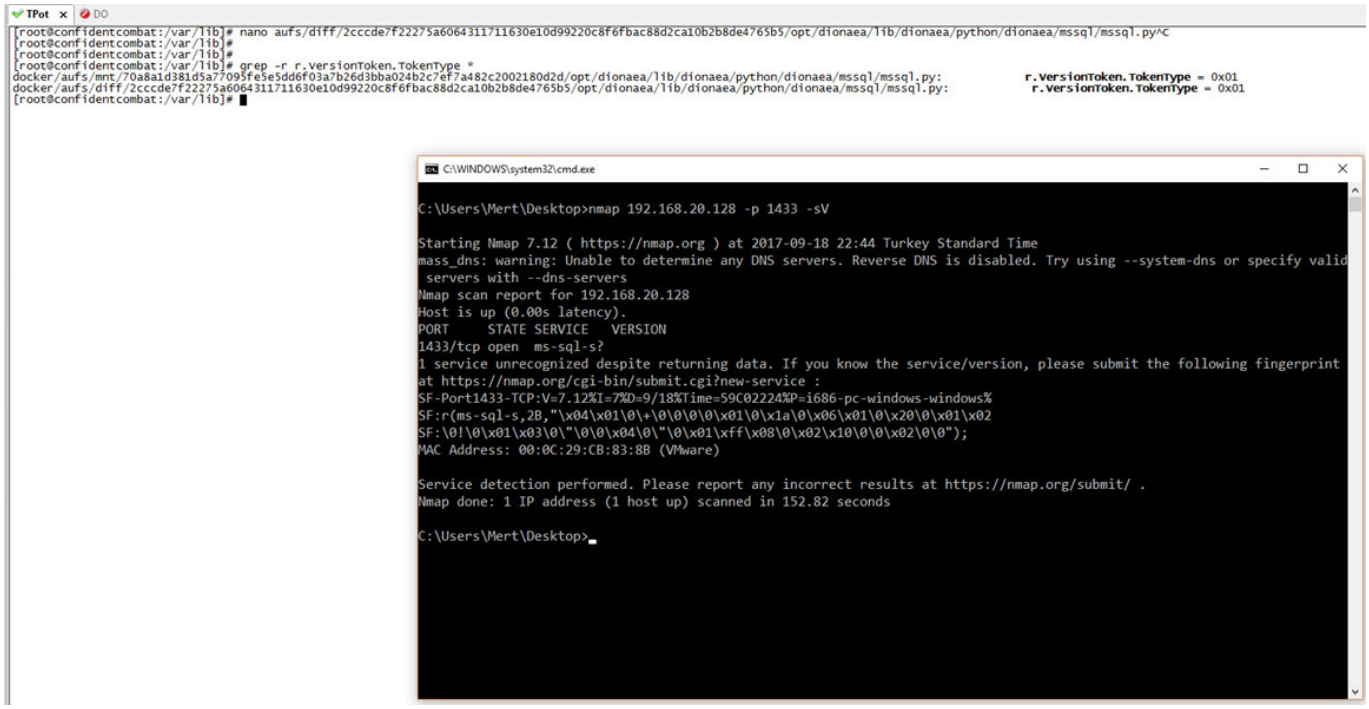
25.09.2008 - MS08-067 - CVE-2008-4250

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.

When it comes to honeypot systems, many people think of Dionaea. As seen in the screenshot above, Dionaea can be easily detected by Nmap when installed with the default settings. However, a quick search on the internet reveals numerous resources (#1, #2, #3) on how to make Dionaea appear undetectable. For example, by changing the “r.VersionToken.TokenType” parameter in the “/dionaea/mssql/mssql.py” file from 0x00 to 0x01, which simulates the MSSQL service, Nmap can no longer detect Dionaea running on port 1433. Since Dionaea simulates vulnerable services (low interaction), making it appear undetectable based on the information found in these resources can make it quite easy to identify Dionaea from the perspective of a cyber attacker. This sparked my interest in researching how easy it actually is to detect an “undetectable” Dionaea.



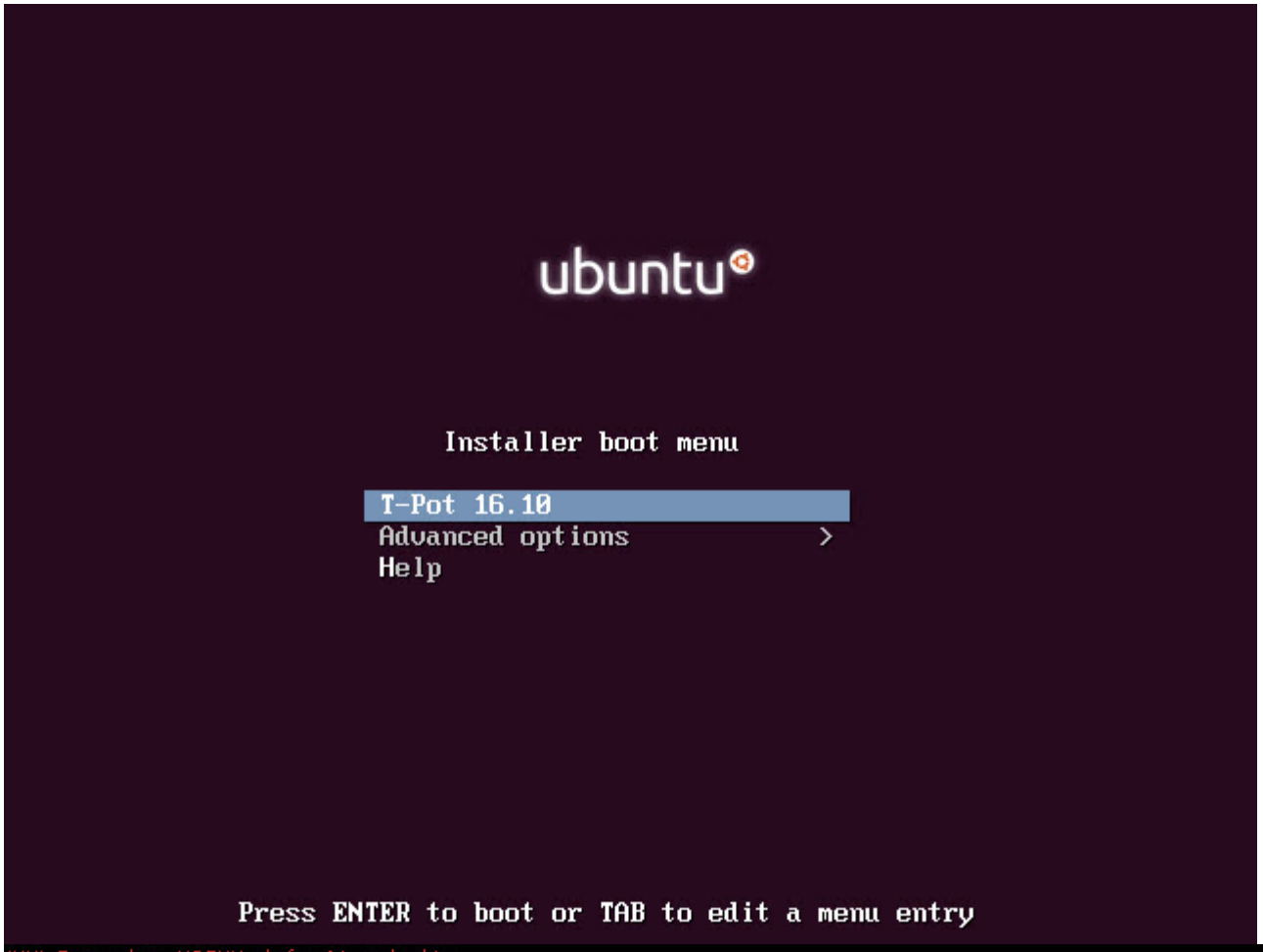
To avoid the hassle of setting up Dionaea, I opted to install T-Pot, a honeypot virtual system developed by Deutsche Telekom that includes numerous honeypot systems, including Dionaea. Considering that a small honeypot system like Dionaea may not fully simulate the MSSQL service (TDS protocol), I decided to start with port 1433.

System Placement

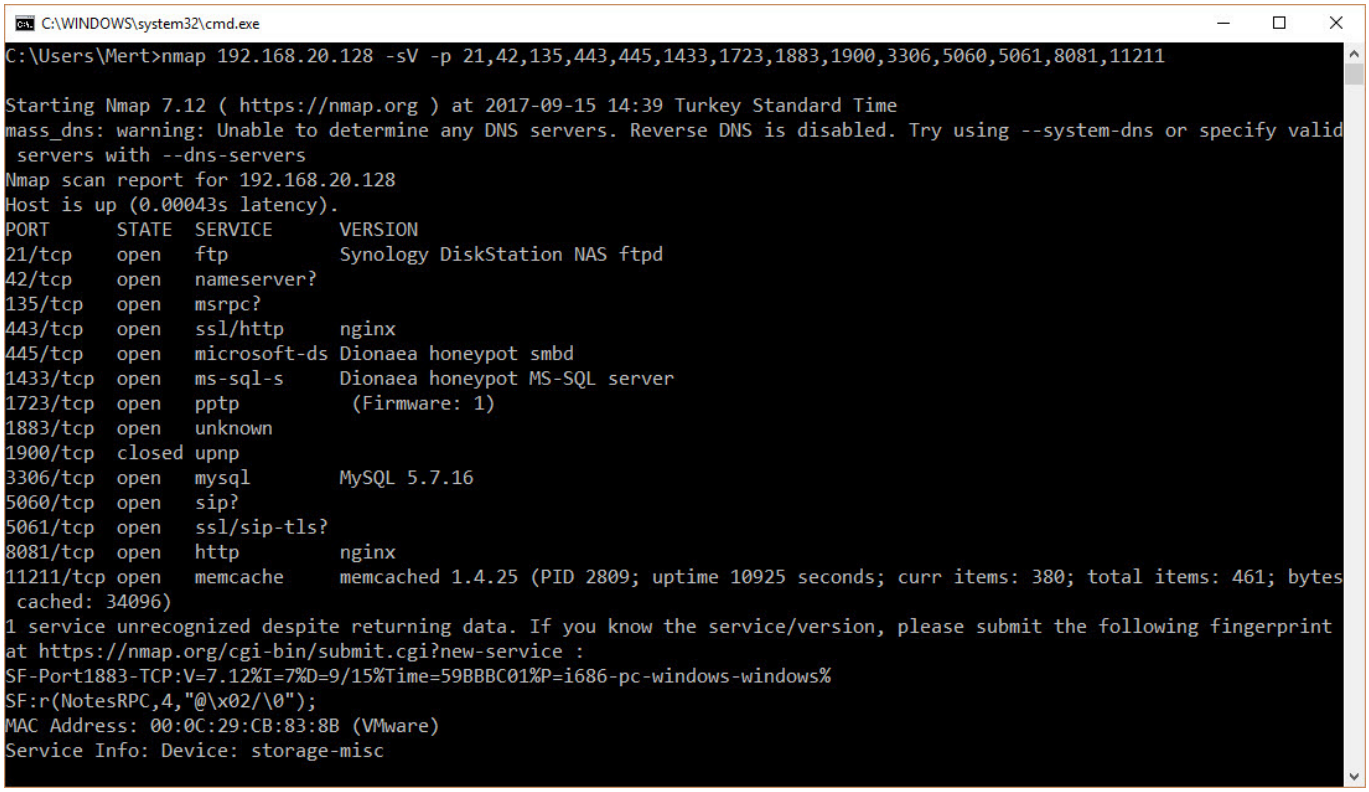
Make sure your system is reachable through the internet. Otherwise it will not capture any attacks, other than the ones from your hostile internal network! We recommend you put it in an unfiltered zone, where all TCP and UDP traffic is forwarded to T-Pot's network interface.

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

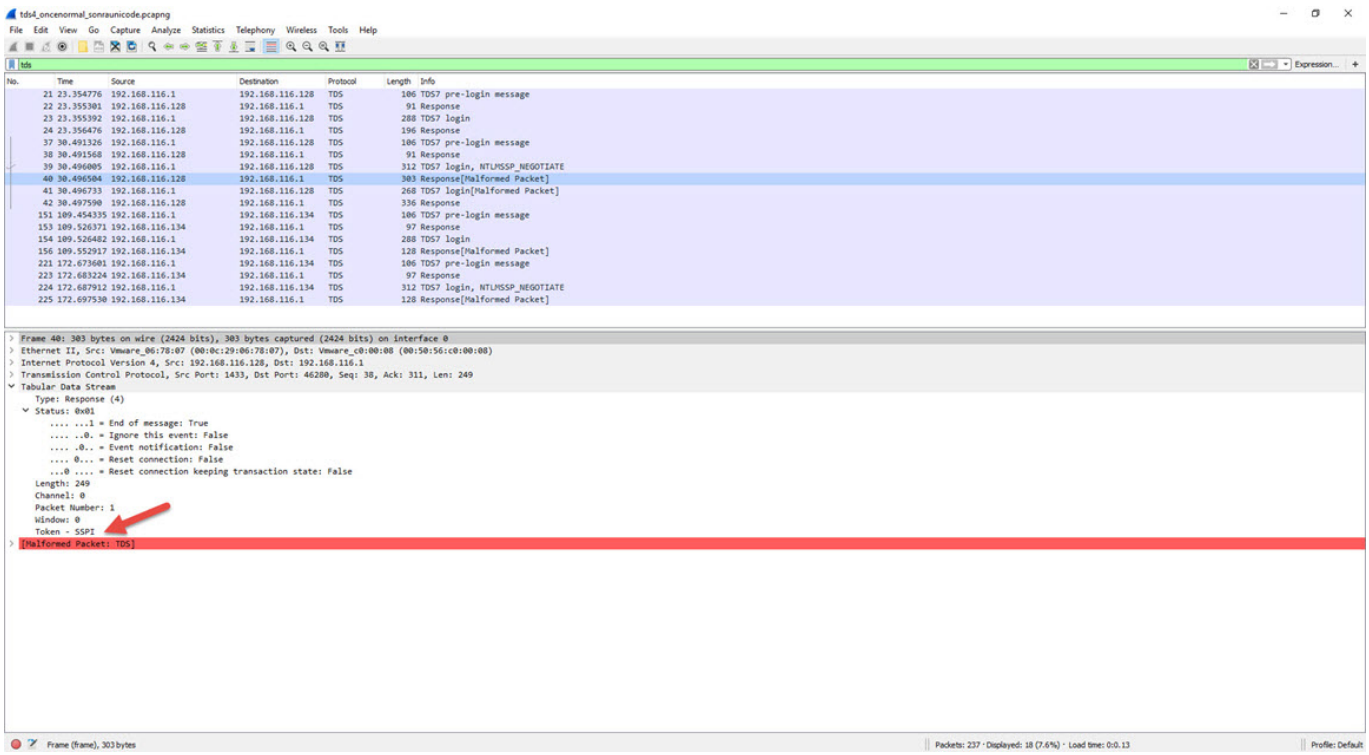



```
### Removing NGINX default website.
### Waiting a few seconds to avoid interference with service messages.
### Please choose your install type and notice HW recommendation.

    [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:
```

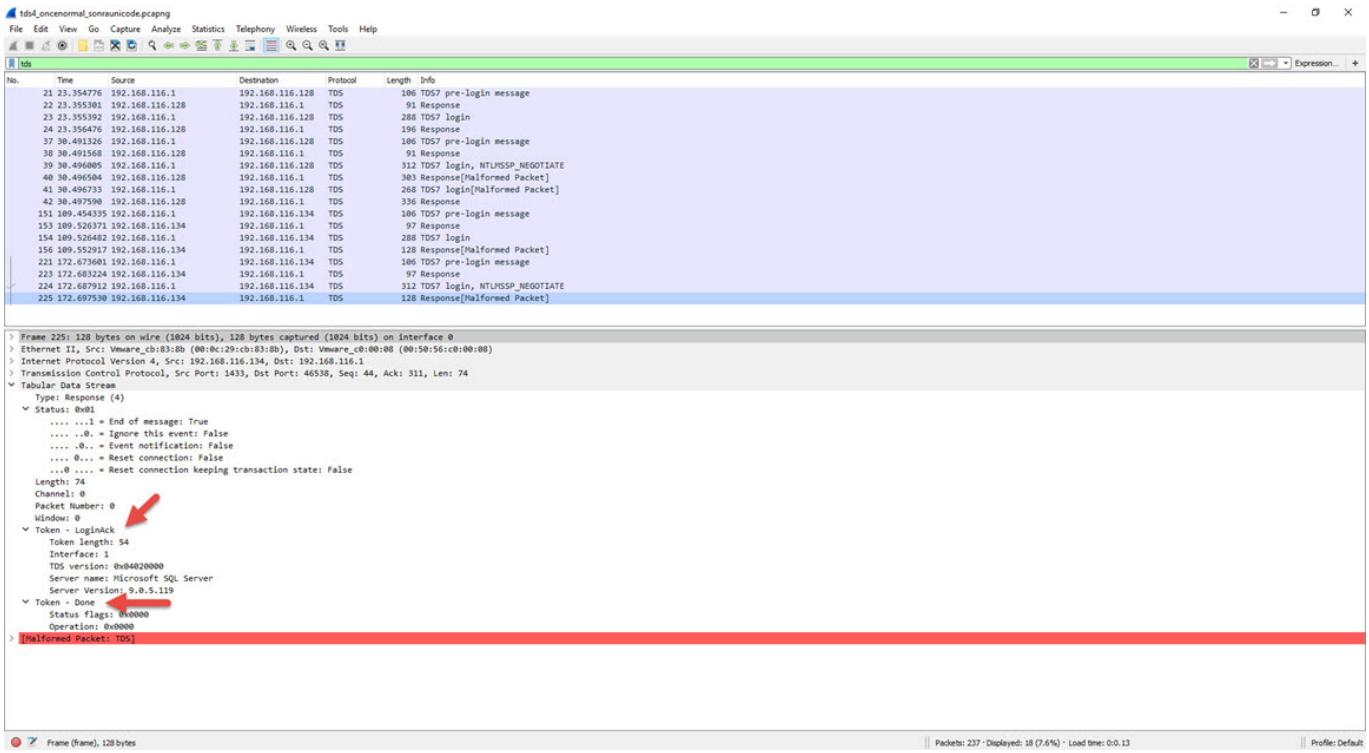


To establish communication between an MSSQL server and a client at the application level, the TDS (Tabular Data Stream) protocol must be used. The TDS protocol supports two types of login methods that have been available since the beginning. The first is login using a username and password, and the second is login using Windows authentication (NTLM). Normally, when attempting to log in with a username and password using the TDS protocol, the response from the MSSQL server should include the LOGINACK_TOKEN (0xAD) token, and when attempting to log in with Windows authentication, it should include the SSPI TOKEN (0xED) token. However, Dionaea returns the same result for both types of requests. :)

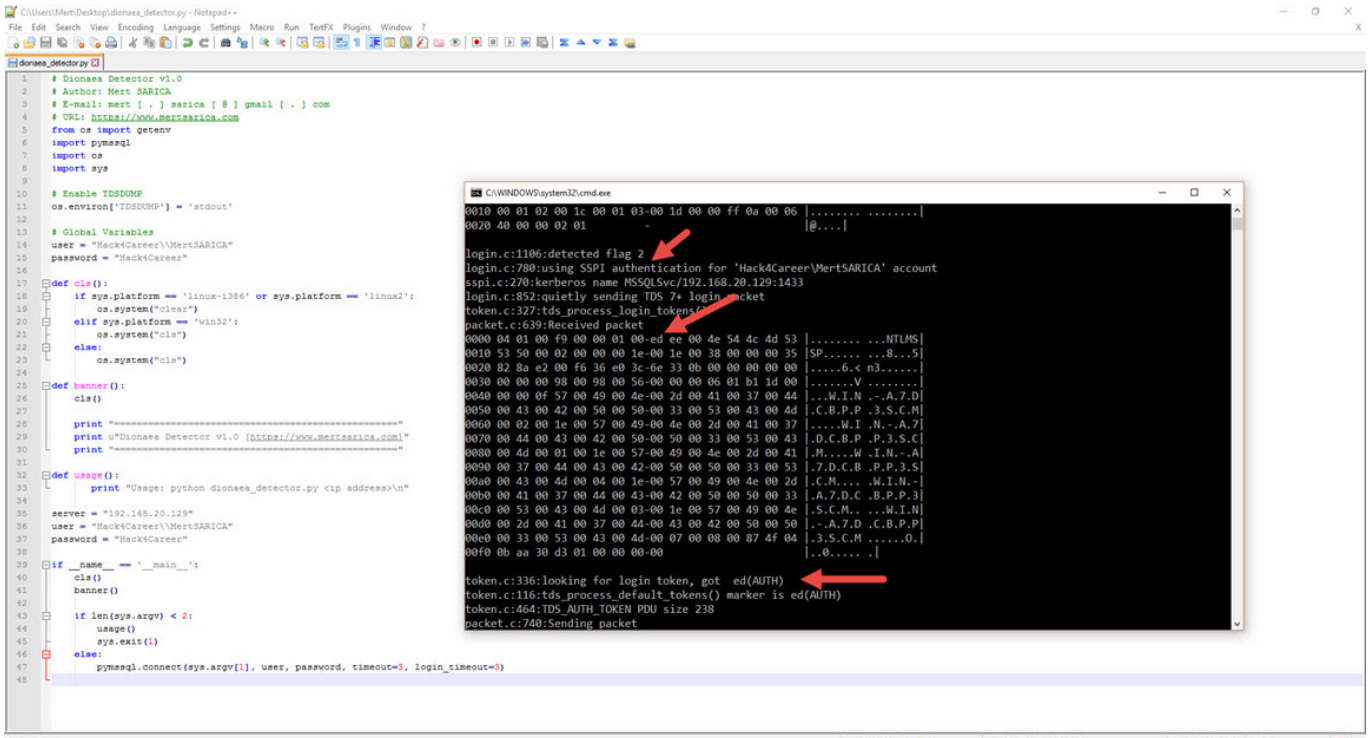
Response from Microsoft SQL 2008 Server Express version to a Windows authentication request

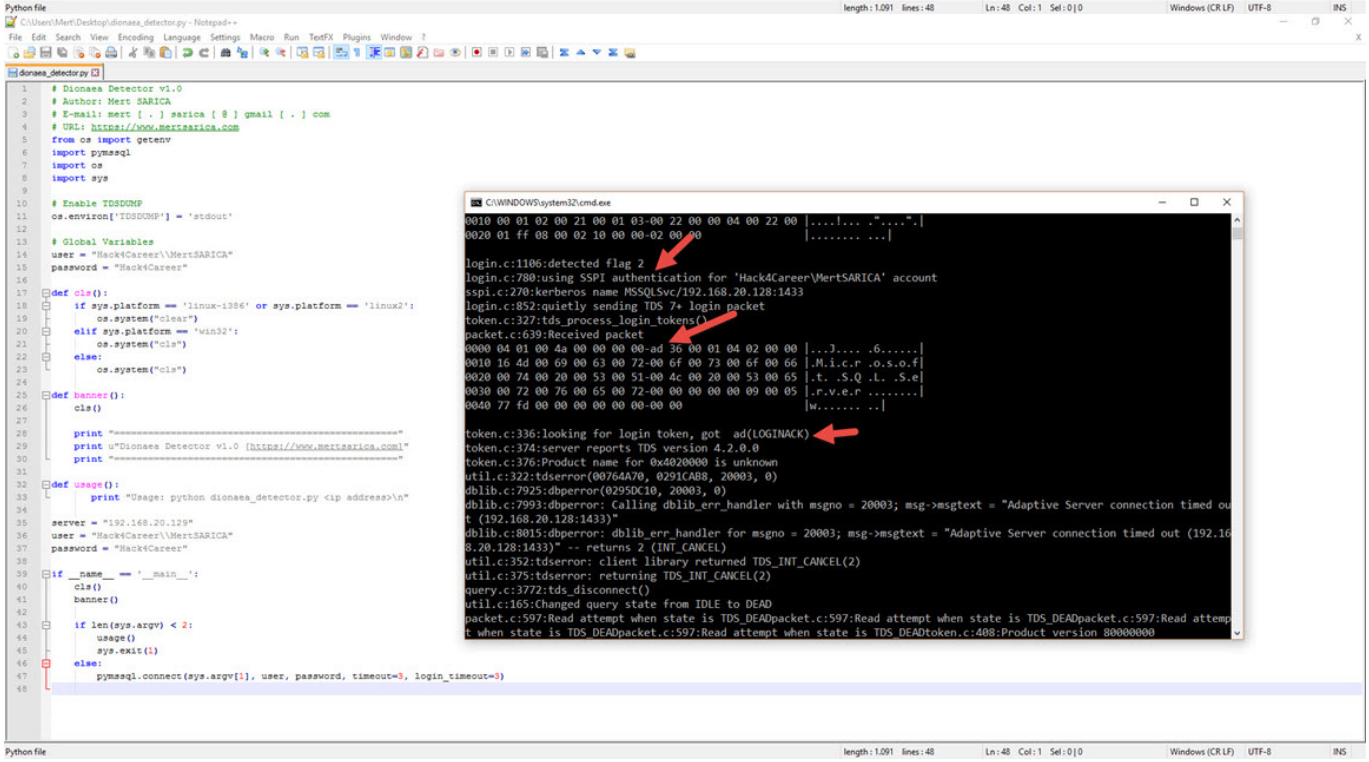


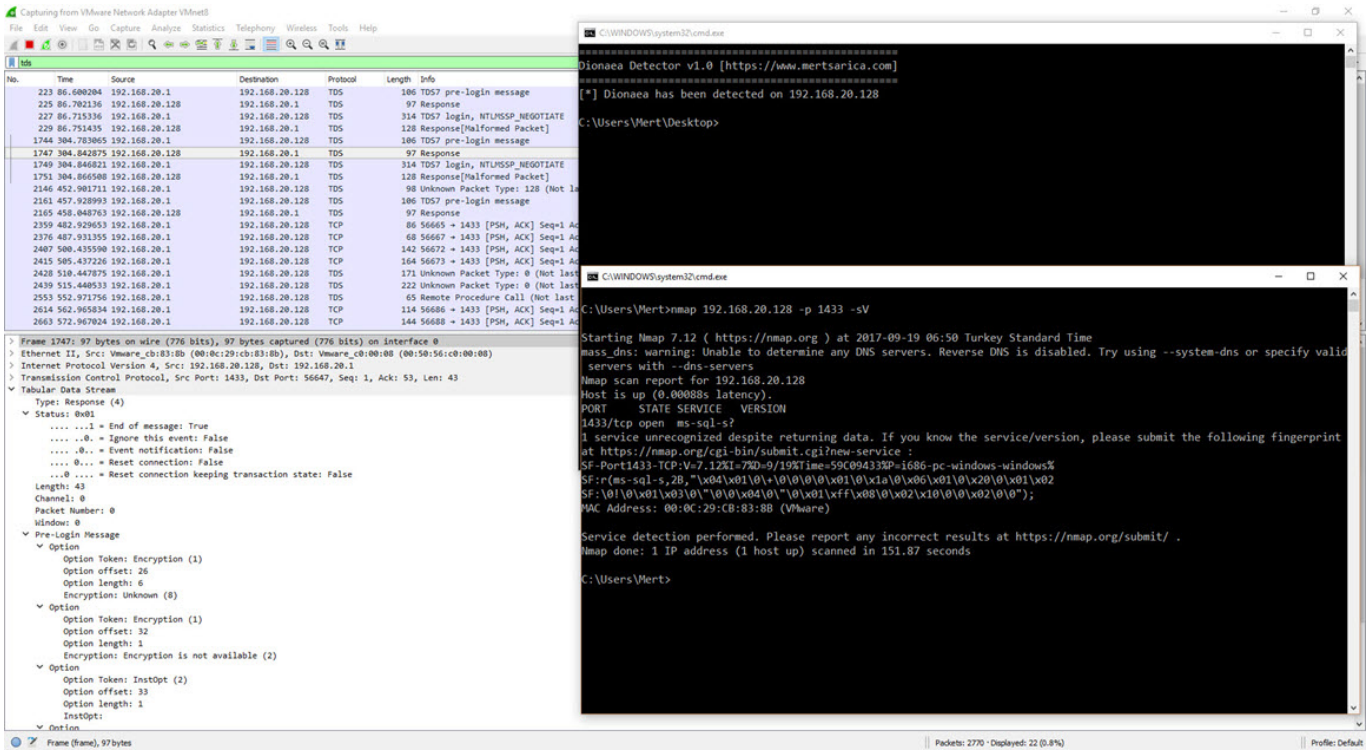
Response from Dionaea to a Windows authentication request



In such a situation, I quickly prepared a simple tool named “dionaea_detector.py” using the pymssql library in Python, which can detect this difference. With this tool, I was able to identify the Dionaea honeypot system, which Nmap couldn’t detect, through a simple check. By doing this, I learned how easily malicious individuals can practically detect it.


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 ou
t (192.168.20.128:1433)"
dblib.c:8015:dbperror: dblib_err_handler for msgno = 20003; msg->msgtext = "Adaptive Server connection timed out (192.16
8.20.128:1433)" -- 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 attemp
t 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



Exactly, before using a honeypot system, it is important to thoroughly evaluate the advantages and disadvantages of high and low interaction honeypots. Choosing the one that is difficult to detect by cyber attackers or making modifications to existing systems to make it more challenging can be beneficial for you or your organization. By considering these factors and implementing the appropriate measures, you can enhance your security and gain valuable insights into the tactics and techniques used by malicious actors.

Hope to see you in the following articles.

Note: Although the screenshots belong to T-Pot version 16.10, it has been confirmed that the dionaea_detector.py tool can successfully detect Dionaea in the latest version of T-Pot, which is version 17.10 sürümü.