

Bachelor's Thesis

Bachelor of Business Administration, Business IT

2020

Vesa-Matti Antero Mäntysaari

# PLANNING AND IMPLEMENTATION OF HONEYPOT SYSTEM

– Building of a bogus Microsoft SQL server

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Bachelor of Business Administration, Business IT

2020 | 32 pages, 8 in appendices

Vesa-Matti Antero Mäntysaari

# PLANNING AND IMPLEMENTATION OF HONEYPOT SYSTEM

- Building of a bogus Microsoft SQL server

The main objective of the thesis' was the building of a prototype bogus Microsoft SQL server, essentially creating a honeypot Microsoft SQL server. Additional machines like the Intra were included to determine how the honeypot Microsoft SQL server could be integrated into a believable honeypot network.

Requirements for the honeypot Microsoft SQL server and honeypot network were established and a constructive research method was selected. The honeypot SQL server would be considered a success if one could connect to it with genuine Microsoft SQL tools, whilst for the honeypot network the number of total machines was set to five, each with their own separate functions.

During the research portion for the required hardware different bare metal and virtualization technologies were examined with the final plan materializing around nested virtualization with the Oracle VM VirtualBox and Proxmox, whilst in the Microsoft SQL server research, the prevalence of Tabular Data Stream protocol on top of TCP was discovered.

The building of the honeypot SQL server in Python proceeded quickly once it was discovered that Tabular Data Stream packets contain the message type as the first byte in the message.

As forensics the logs from the machines inside the honeypot network were analyzed and an attack was confirmed after the machines were breached with either Kali tools or the mere browser, whilst the honeypot SQL server was subjected only to port scan and a connection test with the sqlcmd utility.

In the end, the additional machines which were meant to bring substance to the experiment were a success as the Intra wiki allowed the hiding of critical information believably into the page edit history, while the prototype of a honeypot Microsoft SQL server gave a successful impression of a working database server while also highlighting what the full honeypot database system would need to fool more experienced attackers.

## KEYWORDS:

honeypot, HIDS, Microsoft SQL Server, Tabular Data Stream, Proxmox, nested virtualization, computer forensics

Vesa-Matti Antero Mäntysaari

# HUNAJAPURKKIJÄRJESTELMÄN SUUNNITTELU JA TOTEUTUS

- Microsoft SQL -valetietokantapalvelimen rakentaminen

Opinnäytteen päätarkoituksena oli rakentaa prototyyppi Microsoft SQL -valepalvelimesta, käytännössä siis itsenäinen Microsoft SQL -palvelin hunajapurkki. Jotta pystyttäisiin päättämään, miten hunajapurkin voisi uskottavasti liittää osaksi hunajapurkki ympäristöä, projektiin lisättiin lisäksi koneita, kuten sisäinen wikisivusto.

Asetettiin vaatimukset hunajapurkki Microsoft SQL -palvelimelle ja hunajapurkki ympäristölle. Työssä käytettiin konstruktivistista tutkimusmenetelmää. Hunajapurkki SQL -palvelimen toteutusta pidettäisiin onnistuneena, jos siihen pystyttäisiin yhdistämään aidolla Microsoft SQL -työkalulla. Ympäristölle asetettiin vaatimukseksi viisi tietokonetta, joista jokaisella oli erilainen funktio.

Laitteistovaatimusten tutkinnassa käsiteltiin erilaisia "bare metal" -palvelintyyppisiä ja virtualisointitekniikoita, mutta lopullinen suunnitelma koottiin sisäkkäisen virtualisoinnin ympärille käyttäen Oracle VM Virtual Boxia ja Proxmoxia. Microsoft SQL -palvelimen tutkinnassa havaittiin sen käyttävän Tabular Data Stream -protokollaa TCP-yhteiskäytännössä. Hunajapurkki Microsoft SQL -palvelimen ohjelmointi Pythonilla sujui nopeasti, kun havaittiin, että Tabular Data Stream -paketit sisältävät viestityypin ensimmäisenä bittinä.

Tietoturvaloukkauksen tutkintaosiossa hunajapurkkiverkkoon kuuluvilta tietokoneilta lähetettyjä lokeja analysoitiin ja varmistettiin tapahtunut hyökkäys joko Kalin työkaluilla tai pelkällä selaimella. Hunajapurkki Microsoft SQL-palvelimeen kohdistettiin porttiskannaus ja yhteydentestaus SQLCMD-työkalulla.

Työssä todettiin hunajapurkkiverkkoon liitettyjen lisätietokoneiden tuoneen lisäarvoa ja uskottavuutta projektiin, sillä sisäinen wikisivusto salli kriittisen tiedon piilottamisen uskottavalla tavalla sivun muokkaushistoriaan. Hunajapurkki Microsoft SQL -palvelimen prototyyppi antoi uskottavan vaikutelman toimivasta tietokantapalvelimesta, ja samalla korostui, mitä täysi hunajapurkkietokantajärjestelmä tulisi tarvitsemaan huijatakseen kokeneempiakin hyökkäjiä.

## ASIASANAT:

hunajapurkki, HIDS, Microsoft SQL Server, Tabular Data Stream, Proxmox, nested virtualization, forensikka

# CONTENT

<b>LIST OF ABBREVIATIONS</b>	<b>7</b>
<b>1 INTRODUCTION</b>	<b>8</b>
<b>2 REQUIREMENTS FOR THE HONEYPOT SYSTEM</b>	<b>9</b>
<b>3 FINALIZATION OF THE HONEYPOT PROTOTYPE PLAN</b>	<b>10</b>
3.1 What is nested virtualization	12
<b>4 BUILDING THE HONEYPOT PROTOTYPE</b>	<b>13</b>
4.1 pfSense firewall	14
4.2 Office computer	15
4.3 Log collector	16
4.4 MediaWiki powered Intra server	16
4.5 Bogus Microsoft SQL server	18
<b>5 BREAKING INTO THE HONEYPOT</b>	<b>20</b>
5.1 Office computer	21
5.2 MediaWiki powered Intra server	22
5.3 Bogus Microsoft SQL server	24
<b>6 FORENSICS</b>	<b>26</b>
6.1 Office computer	26
6.2 MediaWiki powered Intra server	27
6.3 Bogus Microsoft SQL server	28
<b>7 CONCLUSION</b>	<b>31</b>
<b>REFERENCES</b>	<b>32</b>

## APPENDICES

Appendix 1. Researching the Microsoft SQL 2017 server

Appendix 2. Building the bogus Microsoft SQL database server

## FIGURES

Figure 1. Network diagram for the honeypot node	11
Figure 2. Nested virtualization shown as levels (Wasserman, 2013)	12
Figure 3. Client to server connection steps shown as a simple diagram ([MS-TDS]: Tabular Data Stream Protocol, 2020)	1
Figure 4. Flowchart of the server states in TDS protocol ([MS-TDS]: Tabular Data Stream Protocol, 2020)	5

## PICTURES

Picture 1. Proxmox VM system settings with enabled nested virtualization support	13
Picture 2. Proxmox networking table	13
Picture 3. Network devices connected to the pfSense in Proxmox	14
Picture 4. The enabled interfaces on pfSense's WebGUI	14
Picture 5. The changed Tunable in pfSense advanced settings	14
Picture 6. pfSense WAN port firewall rules	15
Picture 7. pfSense LAN port firewall rules	15
Picture 8. UFW firewall status showing the open SSH ports	15
Picture 9. Verifying that usage of sudo is not allowed	16
Picture 10. Log Collector up and running as "logpot"	16
Picture 11. Verifying that DNS service works	17
Picture 12. The main page on Intra	17
Picture 13. Verifying that the Apache is not exposing version information to client	18
Picture 14. The scan result for the default gateway	20
Picture 15. THC-Hydra showing the correct password for user pertti	21
Picture 16. The directory listing of Documents folder	21
Picture 17. Todo file printed to the CLI	22
Picture 18. Intra page saying that database should be used to save customer data	22
Picture 19. Edit history of talk page showing the message Jyrki had sent	23
Picture 20. Edit history page for user Pertti showing Jyrki's message	23
Picture 21. The diff page showing the database credentials that were edited out	24
Picture 22. The recon query	24
Picture 23. Unexpected connection dropout	25
Picture 24. Kibana showing the difference between failures and successes	26
Picture 25. Logs showing the brute force and the successful login to Pertti's account	27
Picture 26. Command <b>sudo mkdir checkpoint</b> failed	27
Picture 27. Log entry showing 10.0.2.32 accessing the edit page function	28
Picture 28. Log entry showing the user accessing page's edit history	28
Picture 29. User accessing the diff function on the page that contained credentials	28
Picture 30. The successful binding to port 1433 and the first connection	29
Picture 31. The full connection establishment and authentication for user sa	29
Picture 32. The recon query and the following disconnect	30
Picture 33. Showing the usage of sqlcmd	1
Picture 34. The SQL query and it's result	1
Picture 35. Recorded traffic filtered to show only TDS traffic	2
Picture 36. The environment change from "master" to "tuotteet"	2
Picture 37. Set quoted identifier	3

Picture 38. Set text size to 4096	3
Picture 39. The select all Transact-SQL query	3
Picture 40. Table returned by the server	3
Picture 41. Determine the packet's meaning with the header type	1

## **TABLES**

Table 1. Nested virtualization support for AMD Ryzen platform	10
Table 2. Machines and their function in the honeypot system	11
Table 3. Scan results gathered	20
Table 4. Machine IPs and hostnames mapped	21

## LIST OF ABBREVIATIONS

Abbreviation	Explanation of abbreviation (Source)
Bare metal	Physical dedicated server (Reynaldo, 2014)
Beats	Open and free data shipper platform (Beats: Data Shippers for Elasticsearch   Elastic, 2020)
HIDS	Host-based intrusion detection system (Wazuh · The Open Source Security Platform, 2020)
Honeypot	Security resource that is meant to be attacked (Spitzner 2002, 58)
TDS	Tabular Data Stream ([MS-TDS]: Tabular Data Stream Protocol, 2020)

# 1 INTRODUCTION

The thesis' aim was to establish the usefulness of a subterfuge technique that is not commonly used, a bogus Microsoft SQL database without an actual database as the backend. In public GitHub there are database honeypots available that either act as a proxy to a dummy database like *MongoDB-HoneyProxy* or as a low to medium-interaction honeypots for non-Microsoft database solutions like *nosqlpot*. (Nazario, 2020) Hence, there was a need for this research.

The bogus database was written in Python and setup in an environment that simulated the office network of a small Finnish metal industry company, with its implementation of a bogus SQL database being considered a success if one can connect to it via genuine Microsoft SQL tools and get an impression of a working server.

To tackle this problem, the constructive research method was used to first determine the requirements for the system (for both the bogus database prototype and the additional machines required to simulate the network) and then, built and tested. In the bogus SQL database's case this meant it was necessary to research how the Microsoft SQL tools and a genuine Microsoft SQL server communicate and then use the acquired knowledge to build a bogus SQL database prototype. The research portion for the additional machines necessitated the specification of what services the machines needed to provide and how those would be used to lead the attacker to the next step.

After these phases were done, the logs were analyzed to verify the points where the monitoring shows possible unusual activity. And finally, in the conclusion it is discussed whether the bogus database prototype reached the given objective and whether the simulated environment provided the research additional value.

The project was primarily focused on the host-based intrusion detection system (HIDS), the bogus MS SQL server, with the simulated environment trying to be as genuine a Finnish (metal industry) company production network as possible in this scope. The project was conducted as a commission.



## 2 REQUIREMENTS FOR THE HONEYPOT SYSTEM

The honeypot should be built as a mix between high and low-interaction principles. High-interaction principles dictate that the attacker should be able to control the systems as they would any regular OS, the resource's goal being the maximum capture of the attacker's interactions (Spitzner 2002, 96). The low-interaction principle, on the other hand, is an opposite to that. The bogus database and Intra wiki will be low-interaction while the Office-PC will be high-interaction.

The bogus SQL database implementation will be considered a success if one can connect to it with the *sqlcmd* utility which comes with the Microsoft Command Line Utilities 15.

The system should consist of more than two virtual machines, the entry point, and the main target with a dedicated log collector machine. The log collector would need to be set behind a strict firewall that only allows traffic that is needed for the log collection.

All machines would need security log handling whilst the bogus database would need a way to output readable logs.

The system GUIs, usernames, bogus data, and other such elements should be in Finnish, to further supplement the simulation of being a Finnish company.

All machines apart from the firewall and the log collector should be part of the same domain named *pertinpelti.local*.

As a bonus the plan will also contain an Intra machine hosting a company wiki. The wiki can be implemented with the MediaWiki and could be used to give the attacker some tips on the bogus database.

The passwords used on the machines should be hard-to-guess long Finnish sentences written together like the following "MustikatsuorastaanPomppivatmustikapiidakkaan", except for "Toimisto-PC" that should have an easily guessable password on the non-privileged user account.

### 3 FINALIZATION OF THE HONEYPOT PROTOTYPE PLAN

Based on the requirements the project could have continued into three directions: two robust machines running a VMware vSphere or an equivalent virtualization platform, five spare machines and a switch, or a virtualization environment inside another non-bare metal virtualization environment, i.e., nested virtualization, which currently works on an AMD Ryzen platform.

Unfortunately, neither five spare machines nor even one robust machine running vSphere was available. Thus, a survey of virtualization solutions allowing the use of nested virtualization with AMD Ryzen was conducted (Table 1).

Table 1. Nested virtualization support for AMD Ryzen platform

	VMware Workstation	Microsoft Hyper-V	Oracle VM VirtualBox	Proxmox
Nested virtualization support as host	Yes, Workstation 8+ [6]	Not in Windows builds older than 19636 and currently no KVM support [7]	Yes, 6.1.4+ [8]	Yes [9]
Nested virtualization support as guest	Yes, Workstation 9+ [6]	Not in Windows builds older than 19636 and currently no KVM support [7]	-	Yes [9]

In the end the combo of Oracle VM VirtualBox and Proxmox was selected, and therefore five virtualized machines in total were needed, each with its own task (Table 2).

Table 2. Machines and their function in the honeypot system

Machine	Services / Function
Toimisto-PC	SSH login with weak password / Password to Intra hidden among regular tasks
Intra	Private MediaWiki / Database credentials hidden in a past page edit
DB	Bogus MS-SQL server / Provide false information
pfSense	Transparent firewall / Connect Log Collector to the main network
Log Collector	Elastic stack (ELK) with Logstash / Collect and index logs centrally sent from clients

As a network diagram it would look like this (Figure 1).

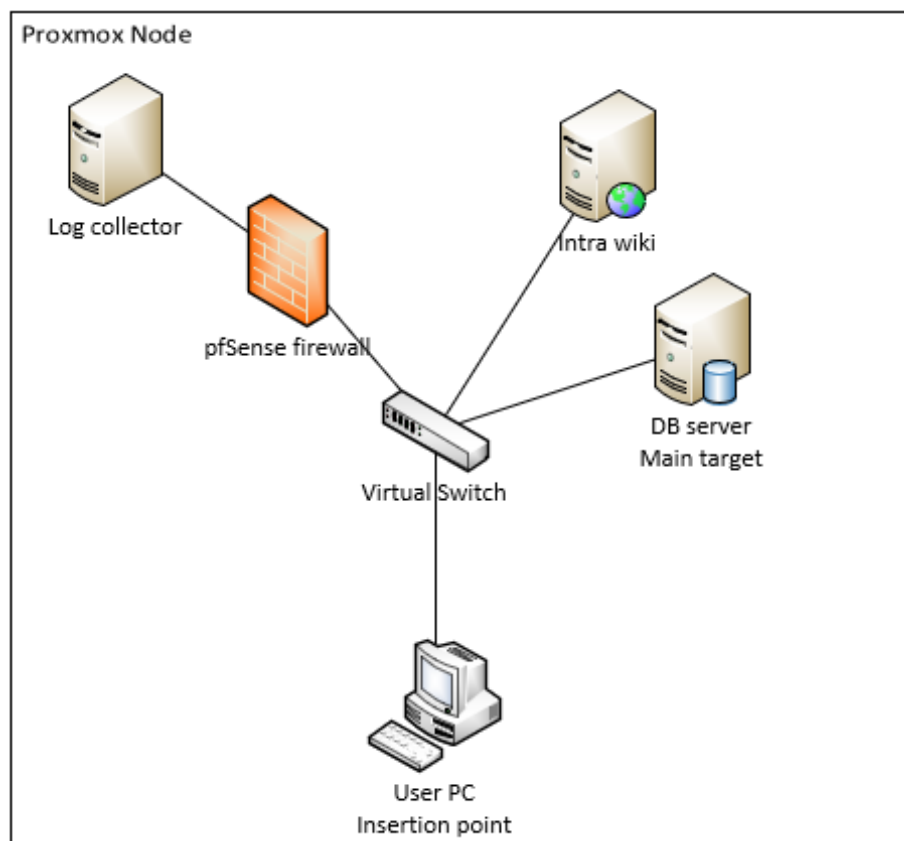


Figure 1. Network diagram for the honeypot node

### 3.1 What is nested virtualization

Nested virtualization allows one to run virtual machines inside other virtual machines. Usually these are described as levels in the nested virtualization diagram, with the lowest number meaning the closest layer to the hardware itself (Wasserman, 2013). In this project this means that the L0 hypervisor will be Oracle VM VirtualBox, whilst L1 hypervisor will be Proxmox and under it (L2) we will be running our honeypot network and its machines (Figure 2).

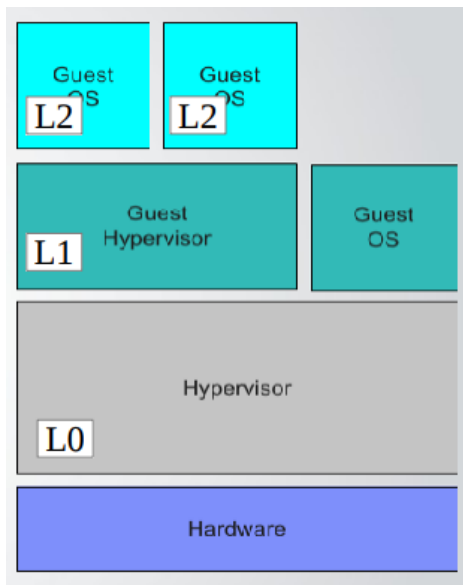
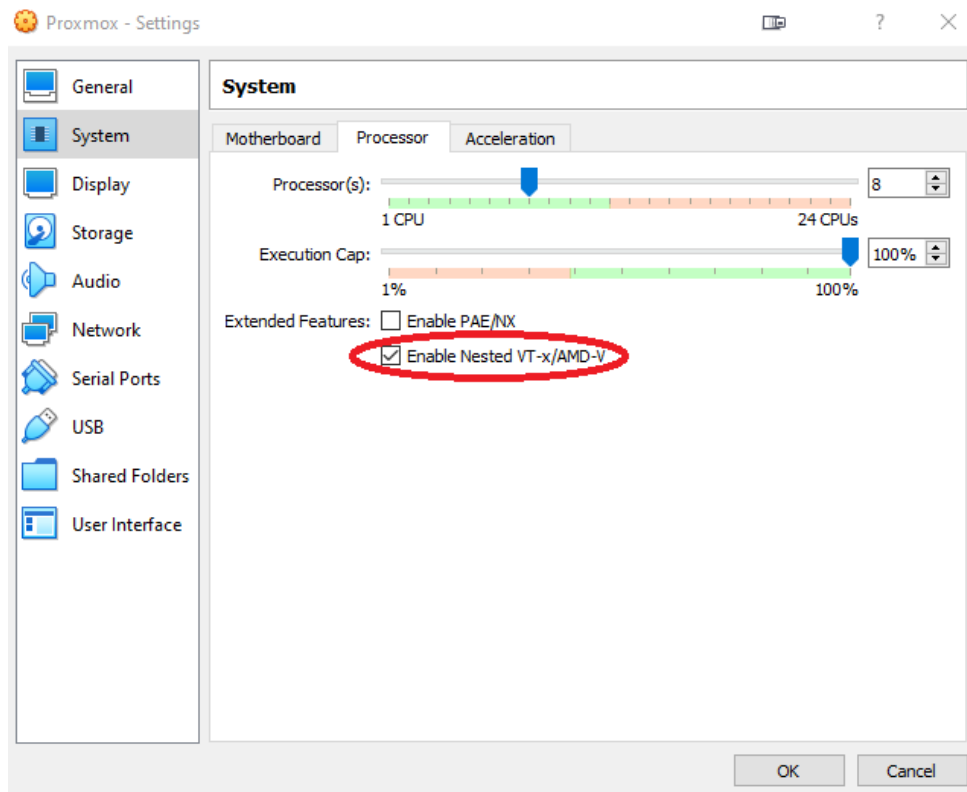


Figure 2. Nested virtualization shown as levels (Wasserman, 2013)

## 4 BUILDING THE HONEYPOT PROTOTYPE

To expose the nested virtualization support to the VM the box “Enable Nested VT-x/AMD-V” must be checked in the VM’s System settings under the Processor tab (Picture 1).



Picture 1. Proxmox VM system settings with enabled nested virtualization support

After the Proxmox installation one extra Linux bridge needs to be added to the Proxmox networking table as the secondary network where the log collector will be setup, since there is one Linux Bridge already the new one gets the name “vibr1” (Picture 2).

Name ↑	Type	Active	Autostart	VLAN aware	Ports/Slaves	B...	CIDR	Gateway	Comment
enp0s3	Network Device	Yes	No	No					
vibr0	Linux Bridge	Yes	Yes	No	enp0s3		10.0.2.15/24	10.0.2.2	Main internal network
vibr1	Linux Bridge	Yes	Yes	No					Aux. network

Picture 2. Proxmox networking table

#### 4.1 pfSense firewall

Configuring pfSense as transparent firewall between the previously mentioned Linux Bridges vmbr0 and vmbr1 will allow the protection of log collector, without exposing the log collector itself. This requires the adding of an additional network device to the pfSense VM with a different Linux Bridge as its link (Picture 3) and the creation of a bridge connection between these devices (Picture 4).

⇒ Network Device (net0)	virtio=72:B6:1E:C2:64:03,bridge=vmbr0
⇒ Network Device (net1)	virtio=82:D7:30:AB:84:1A,bridge=vmbr1

Picture 3. Network devices connected to the pfSense in Proxmox

Interface	Network port
WAN	vtnet0 (72:b6:1e:c2:64:03)
LAN	vtnet1 (82:d7:30:ab:84:1a)
Bridge	BRIDGE0 ("WAN" to "LAN" bridge)

Picture 4. The enabled interfaces on pfSense's WebGUI

To filter the packets moving in the bridge interface the tunable **net.link.bridge.pfil\_bridge** value must be set to 1 at System -> Advanced -> System Tunables (Picture 5).

Edit Tunable	
<b>Tunable</b>	net.link.bridge.pfil_bridge
<b>Value</b>	1
<b>Description</b>	Packet filter on the bridge interfa

Picture 5. The changed Tunable in pfSense advanced settings

Lastly, the firewall rules needed to be changed. In the WAN rule section incoming traffic meant to Logstash was allowed and on top of that the anti-lockout rule to the pfSense WebGUI was created (Picture 6).

Rules (Drag to Change Order)											
<input type="checkbox"/>	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input type="checkbox"/>	✗ 0 / 0 B	*	Reserved Not assigned by IANA	*	*	*	*	*		Block bogon networks	⚙️
<input type="checkbox"/>	✓ 0 / 656 B	IPv4 TCP	WAN net	*	10.0.2.201	5044	*	none		Allow Beats to Logstash port	📌 ✎ 📄 🗑️
<input type="checkbox"/>	✓ 1 / 2.43 MiB	IPv4*	10.0.2.201	*	10.0.2.17	*	*	none		Anti-lockout rule	📌 ✎ 📄 🗑️

Picture 6. pfSense WAN port firewall rules

In the LAN rules it was specified that the log collector can send to any (Picture 7).

Rules (Drag to Change Order)											
<input type="checkbox"/>	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input type="checkbox"/>	✓ 2 / 968 KIB	IPv4*	10.0.2.201	*	*	*	*	none		Allow all outbound from LAN	📌 ✎ 📄 🗑️

Picture 7. pfSense LAN port firewall rules

## 4.2 Office computer

To the Ubuntu 20.04 Office-PC SSH server was installed and the UFW firewall was set to allow the incoming SSH connections (Picture 8).

```
jyrki@Toimisto-PC:~$ sudo ufw status
Status: active

To Action From
--
22/tcp ALLOW Anywhere
22/tcp (v6) ALLOW Anywhere (v6)
```

Picture 8. UFW firewall status showing the open SSH ports

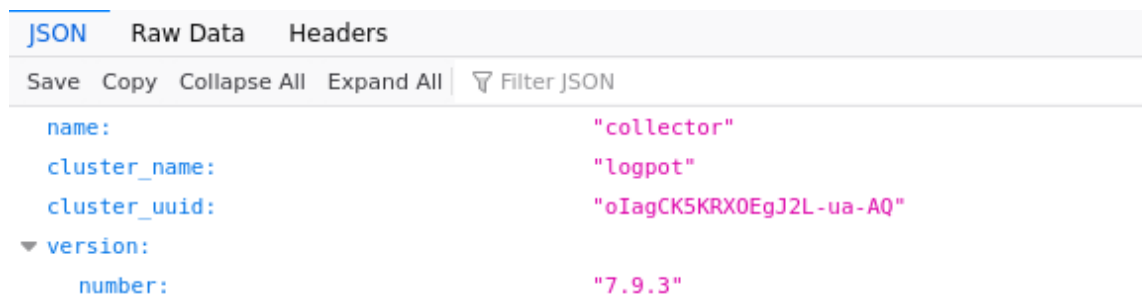
By default, Ubuntu adds the user created on the OS installation to the *sudoers* file and group. However, the first step on the attacker's ladder will not be needing any more privileges than a basic user account gives. Therefore, the additional privileges need to be removed. This can be accomplished with the command **sudo deluser pertti sudo** (Picture 9).

```
pertti@Toimisto-PC:~$ sudo mkdir eionnistu
[sudo] password for pertti:
pertti is not in the sudoers file. This incident will be reported.
```

Picture 9. Verifying that usage of sudo is not allowed

### 4.3 Log collector

The Log Collector will have the Logstash and Elastic stack (ELK) running on Debian 10.6. at the previously mentioned secondary network (Picture 10).



The screenshot shows a JSON viewer interface with tabs for 'JSON', 'Raw Data', and 'Headers'. The 'JSON' tab is active, displaying a configuration object. The object contains the following key-value pairs:

- `name:` `"collector"`
- `cluster_name:` `"logpot"`
- `cluster_uuid:` `"oIagCK5KRX0EgJ2L-ua-AQ"`
- `version:` (expanded)
  - `number:` `"7.9.3"`

Picture 10. Log Collector up and running as "logpot"

### 4.4 MediaWiki powered Intra server

After the installation of MediaWiki, onto a Debian 10.5., it is configured to act as private wiki, allowing access only to authenticated users. Three users are created, one administrative and two with the principles of least privileges needed, as the SP 800-53 Rev. 5 (19-23, 36-39) recommends as access control measures.

DNS service is added to the Intra server and the top domain is named *pertinpelti.local* to simulate a small business in the metal industry and to facilitate the resolving of the honeypot hostnames to IP (Picture 11).



```

jyrki@Toimisto-PC:~$ dig @10.0.2.50 intra.pertinpelti.local

; <<> DiG 9.16.1-Ubuntu <<> @10.0.2.50 intra.pertinpelti.local
; (1 server found)
; ; global options: +cmd
; ; Got answer:
; ; WARNING: .local is reserved for Multicast DNS
; ; You are currently testing what happens when an mDNS query is leaked to DNS
; ; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 63575
; ; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

; ; OPT PSEUDOSECTION:
; ; EDNS: version: 0, flags:; udp: 4096
; ; COOKIE: 186f08de2c12bb721d19e36a5f9bfead0dfdd3ecd6780ed3 (good)
; ; QUESTION SECTION:
; ;intra.pertinpelti.local.      IN      A

; ; ANSWER SECTION:
intra.pertinpelti.local. 604800 IN      A      10.0.2.50

; ; AUTHORITY SECTION:
pertinpelti.local.      604800 IN      NS      ns1.pertinpelti.local.

; ; ADDITIONAL SECTION:
ns1.pertinpelti.local. 604800 IN      A      10.0.2.50

; ; Query time: 0 msec
; ; SERVER: 10.0.2.50#53(10.0.2.50)
; ; WHEN: pe loka 30 13:53:17 EET 2020
; ; MSG SIZE rcvd: 130

```

Picture 11. Verifying that DNS service works

The fake information used in the company wiki will supplement the top domain (Picture 12).



Picture 12. The main page on Intra

Before the installation of a Beat the apache2 and OS information from the HTTP headers and 404 web pages is hidden. It was also verified that the PHP version information is hidden. These result on headers that only contain “Apache” (Picture 13).

```

jyrki@Toimisto-PC:~$ wget --server-response --spider http://intra.pertinpelti.local
Spider mode enabled. Check if remote file exists.
--2020-10-30 13:56:32-- http://intra.pertinpelti.local/
Resolving intra.pertinpelti.local (intra.pertinpelti.local)... 10.0.2.50
Connecting to intra.pertinpelti.local (intra.pertinpelti.local)|10.0.2.50|:80... connected.
HTTP request sent, awaiting response...
HTTP/1.1 302 Found
Date: Fri, 30 Oct 2020 11:56:31 GMT
Server: Apache
Location: http://intra.pertinpelti.local/mediawiki/
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1
Location: http://intra.pertinpelti.local/mediawiki/ [following]
Spider mode enabled. Check if remote file exists.
--2020-10-30 13:56:32-- http://intra.pertinpelti.local/mediawiki/
Connecting to intra.pertinpelti.local (intra.pertinpelti.local)|10.0.2.50|:80... connected.
HTTP request sent, awaiting response...
HTTP/1.1 200 OK
Date: Fri, 30 Oct 2020 11:56:31 GMT
Server: Apache
X-Content-Type-Options: nosniff
Content-language: fi
X-Frame-Options: DENY
Vary: Accept-Encoding, Cookie
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
X-Request-Id: 326862780df05c4860672462
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
Length: unspecified [text/html]
Remote file exists and could contain further links,
but recursion is disabled -- not retrieving.

```

Picture 13. Verifying that the Apache is not exposing version information to client

The Picture 13 also confirms that the redirect from site root to the MediaWiki works as intended, an attacker cannot access the site root as its always redirected to the subdirectory where MediaWiki resides.

#### 4.5 Bogus Microsoft SQL server

Due to database server not being genuine the machine only needs the Python 3 which is already installed. Therefore, only the machine’s firewall needed to be configured and it was setup to only allow incoming connections to the TCP port 1433 which is the default for Microsoft SQL server installations.

After the Debian 10.5. machine was ready the research portion for the client and Microsoft SQL 2017 server started in a separate environment (Appendix 1). With the information uncovered during the research it was possible to start the actual programming part in Python (Appendix 2).

## 5 BREAKING INTO THE HONEYPOT

A Kali with the required tools and text files (that contained common Finnish names and passwords) was added to the vmbr0 bridge. The machine got the IP of 10.0.2.32/24 and the firstly network scanning with the tool *Nmap* was done. This gave a nice overview of the machines connected to the network (Table 3).

Table 3. Scan results gathered

IP	Open ports / services
10.0.2.22	22 / SSH
10.0.2.50	80 / HTTP
10.0.2.111	1433 / MS-SQL
10.0.2.201	5044 / Logstash

The scan also showed with a very high certainty that the default gateway 10.0.2.2 is running within a virtualized environment (Picture 14).

```
Nmap scan report for 10.0.2.2
Host is up (0.00050s latency).
Not shown: 995 filtered ports
PORT      STATE SERVICE
135/tcp   open  msrpc
445/tcp   open  microsoft-ds
902/tcp   open  iss-realsecure
912/tcp   open  apex-mesh
3221/tcp  open  xnm-clear-text
MAC Address: 52:54:00:12:35:02 (QEMU virtual NIC)
Warning: OSScan results may be unreliable because we could not find at least 1
open and 1 closed port
Device type: general purpose|bridge|switch
Running (JUST GUESSING): QEMU (99%), Oracle Virtualbox (95%), Bay Networks embe
dded (90%)
OS CPE: cpe:/a:qemu:qemu cpe:/o:oracle:virtualbox cpe:/h:baynetworks:baystack_4
50
Aggressive OS guesses: QEMU user mode network gateway (99%), Oracle Virtualbox
(95%), Bay Networks BayStack 450 switch (software version 3.1.0.22) (90%)
```

Picture 14. The scan result for the default gateway

With the following *Nmap* command **sudo nmap -sn 10.0.2.\*** the IPs were mapped to the correct hostnames (Table 4).

Table 4. Machine IPs and hostnames mapped

IP	Hostname
10.0.2.22	Toimisto-PC
10.0.2.50	Intra
10.0.2.111	db
10.0.2.201	-

After those were analysed, the password list was formatted by removing the unnecessary spaces after the passwords and then the brute forcing attack was started on the open SSH port of “Toimisto-PC” using the THC-Hydra tool. It took little over an hour to find the correct password (Picture 15).

```
[ATTEMPT] target 10.0.2.22 - login "admin" - pass "karhu" - 4242 of
[ATTEMPT] target 10.0.2.22 - login "jyrki" - pass "karhu" - 4243 of
[ATTEMPT] target 10.0.2.22 - login "pertti" - pass "karhu" - 4244 of
[22][ssh] host: 10.0.2.22 login: pertti password: karhu
[STATUS] attack finished for 10.0.2.22 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
```

Picture 15. THC-Hydra showing the correct password for user pertti

### 5.1 Office computer

Upon logging into the “Toimisto-PC” the *sudo* command was tried, but it was denied, the current account did not have privileges to use the command. Therefore, as a resort it the machine’s directories were searched for any worthwhile information (Picture 16).

```
pertti@Toimisto-PC:~$ ls -l Documents/
total 16
drwxrwxr-x 4 pertti pertti 4096 loka 30 14:31 prokkikset
drwxrwxr-x 2 pertti pertti 4096 loka 31 19:20 talous
drwxrwxr-x 2 pertti pertti 4096 loka 31 18:45 tuotteet
drwxrwxr-x 2 pertti pertti 4096 loka 31 19:14 yhtiöasiat
```

Picture 16. The directory listing of Documents folder

When searching among the subdirectories of Documents an interesting looking todo file was found with what seemed like an embedded password in the todo objectives. And sure enough, by closely looking at the last objective it was determined that the password



for Intra is either: "Yö Saapuu nopeasti Joka Päivänä" or "YöSaapuu nopeasti Joka Päivänä" (Picture 17).

```
pertti@Toimisto-PC:~$ cat Documents/prokkikset/nykyiset\ prokkikset/todo
1. Siirrä talousasiat ja tuotekatalogi intraan - kesken
2. Muista myös vertailla raakamateriaalien hankintakustannuksia
3. Jyrille lupa tehdä muutoksia firman verkkoon - OK
4. Delegoi "ilmoita työntekijöille työntekörajoituksista koronan takia" - OK Mari otti kopin
5. Hanki Jyrille tarvittavat laitteet - etätöiden mahdollistamiseksi - OK
6. Seuraa pellin työstämiseen tarvittavien laitteiden hintoja - Delegoitu
7. Muista pitää taukoa töistä myös vaikeina aikoina sillä Yö Saapuu nopeasti Joka Päivänä eivätkä työt tule koskaan loppumaan
```

Picture 17. Todo file printed to the CLI

## 5.2 MediaWiki powered Intra server

At Intra all pages redirect to a login prompt, however by presuming the username is the same as on the Ubuntu it is possible to log in easily. Although the password needed to be written without the spaces. Inside the Intra the pages were searched for anything worthwhile manually, and in the end, a page detailing how the database will be used to handle customer data was found (Picture 18).

Sivu **Keskustelu** Lue **Muokkaa** **Näytä historia**  Muut 

## Asiakkaat

Siirretty tietokantaan, tulevaisuudessa käytetään vain tietokantaa asiakkaiden tallentamiseen

Picture 18. Intra page saying that database should be used to save customer data

On the talk page for user Pertti Jyrki (the site admin) had left a message not to save any credentials on the wiki pages (Picture 19).

Käyttäjäsivu Keskustelu Lue Muokkaa Lisää aihe Näytä historia Muut Hae Intrasta

## Sivun ”Keskustelu käyttäjä:Pertti” muutoshistoria Ohjeet

Näytä tämän sivun lokit

▼ Suodata versioita

Eroavaisuuksien valinta: Merkitse niiden versioiden valintaympyrät, joita haluat vertailla, ja paina enter tai alhaalla olevaa nappia.

Selitys: **(nyk.)** = eroavaisuudet uusimpaan versioon, **(edell.)** = eroavaisuudet edeltävään versioon, **p** = pieni muutos.

- (nyk. | edell.) 31. lokakuuta 2020 kello 19.08 Jyrki (keskustelu | muokkaukset) . . (75 tavua) (+75) . . (Ak: Uusi sivu: Hei Pertti, Voisitko tulevaisuudessa olla tallentamatta tunnuksia intraan.)

Picture 19. Edit history of talk page showing the message Jyrki had sent

With the knowledge of basic wiki installations keeping all page content changes in the page histories it would not be far-fetched for the credentials to still be there, wherever that may be. Therefore, the page histories were manually searched for any worthwhile information. After a while, the user page for Pertti was found to contain a change entry by Jyrki with the change's comment telling not to save credentials to Intra (Picture 20).

Käyttäjäsivu Keskustelu Lue Muokkaa Näytä historia Muut Hae Intrasta

## Sivun ”Käyttaja:Pertti” muutoshistoria Ohjeet

Näytä tämän sivun lokit

▼ Suodata versioita

Eroavaisuuksien valinta: Merkitse niiden versioiden valintaympyrät, joita haluat vertailla, ja paina enter tai alhaalla olevaa nappia.

Selitys: **(nyk.)** = eroavaisuudet uusimpaan versioon, **(edell.)** = eroavaisuudet edeltävään versioon, **p** = pieni muutos.

Vertaile valittuja versioita

- (nyk. | edell.)  31. lokakuuta 2020 kello 19.06 Jyrki (keskustelu | muokkaukset) **p** . . (7 tavua) (-35) . . (Älä tallenna tunnuksia intraan) (kumoa)
- (nyk. | edell.)  31. lokakuuta 2020 kello 19.00 Pertti (keskustelu | muokkaukset) **p** . . (42 tavua) (+35) . . (kumoa)
- (nyk. | edell.)  31. lokakuuta 2020 kello 17.38 Pertti (keskustelu | muokkaukset) . . (7 tavua) (+7) . . (Ak: Uusi sivu: Toimari)

Vertaile valittuja versioita

Picture 20. Edit history page for user Pertti showing Jyrki's message

Looking at the diff page between the Jyrki's edit and Pertti's original version we find that the credentials seem to be for the database since the username looks to be *sa*, which is commonly used in Microsoft SQL servers as the default database administrator account (Picture 21).

Käyttäjäsivu [Keskustelu](#)  Lue [Muokkaa](#) [Näytä historia](#) [★](#) Muut  Hae Intrasta

## Ero sivun "Käyttäjä: Pertti" versioiden välillä

<p><b>Versio 31. lokakuuta 2020 kello 17.38</b> (<a href="#">muokkaa</a>) Pertti (<a href="#">keskustelu</a>   <a href="#">muokkaukset</a>) (Ak: Uusi sivu: Toimari)</p>	<p><b>Versio 31. lokakuuta 2020 kello 19.00</b> (<a href="#">muokkaa</a>) (<a href="#">kumoa</a>) Pertti (<a href="#">keskustelu</a>   <a href="#">muokkaukset</a>) <b>P.</b> <a href="#">Uudempi muutos</a> →</p>
<p><b>Rivi 1:</b></p> <div style="border: 1px solid #ccc; padding: 2px; width: fit-content;">Toimari</div>	<p><b>Rivi 1:</b></p> <div style="border: 1px solid #ccc; padding: 2px; width: fit-content;">Toimari</div> <div style="border: 1px solid #ccc; padding: 2px; width: fit-content; margin-top: 5px;">+ sa OotappasKuntalennantänTietueen</div>

---

Versio 31. lokakuuta 2020 kello 19.00

Toimari sa OotappasKuntalennantänTietueen

Picture 21. The diff page showing the database credentials that were edited out

### 5.3 Bogus Microsoft SQL server

With the *sqlcmd* utility already on the Kali machine the command **sqlcmd -U sa -S 10.0.2.111** was used to connect to the database server. However, when trying to log into the database it was found out that the password was written incorrectly to the wiki page. The correct password version did not contain "ä". With the minor setback handled the login completed successfully and it was possible to send a recon query that would show the tables on the database schema (Picture 22).

```
1> SELECT * FROM INFORMATION_SCHEMA.TABLES
2> GO
```

Picture 22. The recon query

After sending the select query it takes a while before the *sqlcmd* utility informs about an unexpected disconnect (Picture 23).



```
TCP Provider: Error code 0x2746  
Communication link failure  
kayttaja@kali:~$ █
```

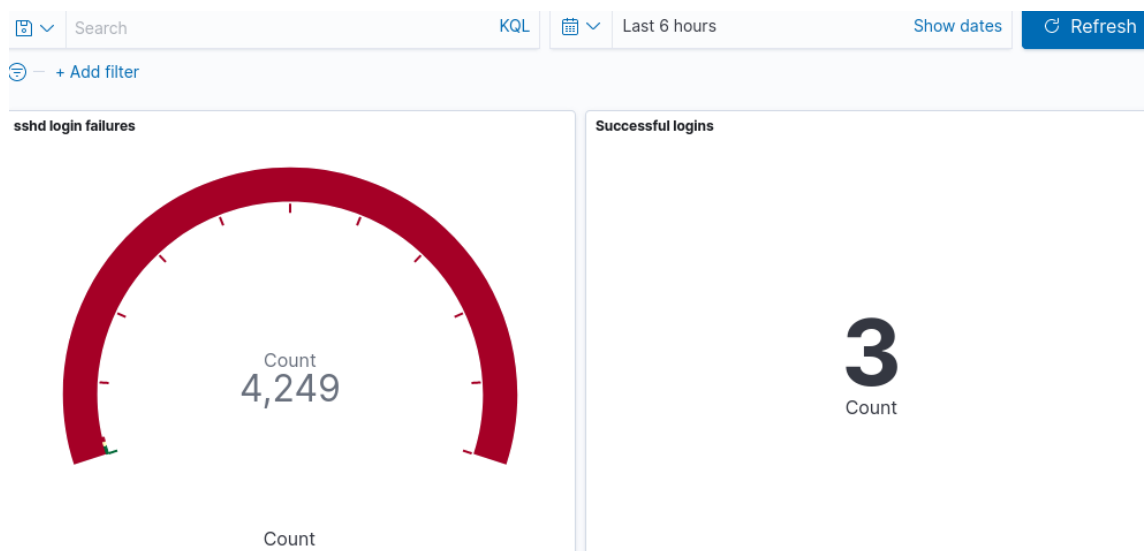
Picture 23. Unexpected connection dropout

## 6 FORENSICS

With the successful completion of penetration testing in the previous section the systems have undoubtedly created very interesting looking logs about the interactions, therefore in this section the objective is to determine what has happened and with whom. It is then determined how much of the information was accessed (if any) and how critical that information was.

### 6.1 Office computer

When looking at the Kibana dashboard for the “Toimisto-PC” an alarming number of failed logins in the last 6 hours were found compared to the successful ones (Picture 24).



Picture 24. Kibana showing the difference between failures and successes

By closely looking at the logs it is discovered that the user pertti was compromised during the brute force attempt (Picture 25).

```

Nov  3 17:45:30 Toimisto-PC sshd[12798]: Failed password for invalid user admin
from 10.0.2.32 port 42258 ssh2

Nov  3 17:45:30 Toimisto-PC sshd[12796]: Received disconnect from 10.0.2.32 port
42256:11: Bye Bye [preauth]

Nov  3 17:45:30 Toimisto-PC sshd[12796]: Disconnected from invalid user root 10.
0.2.32 port 42256 [preauth]

Nov  3 17:45:30 Toimisto-PC sshd[12802]: Connection from 10.0.2.32 port 42262 on
10.0.2.22 port 22 rdomain ""

Nov  3 17:45:30 Toimisto-PC sshd[12802]: Accepted password for pertti from 10.0.
2.32 port 42262 ssh2

Nov  3 17:45:30 Toimisto-PC sshd[12802]: pam_unix(sshd:session): session opened
for user pertti by (uid=0)

```

Picture 25. Logs showing the brute force and the successful login to Pertti's account

After the brute force attack, the logs show the same IP login to Pertti's account, and the first thing that was tried was using the sudo to elevate privileges (Picture 26).

```

Nov  3 17:54:22 Toimisto-PC sudo: pertti : user NOT in sudoers ; TTY=pts/0 ; P
WD=/home/pertti ; USER=root ; COMMAND=/usr/bin/mkdir checkpoint

```

Picture 26. Command **sudo mkdir checkpoint** failed

When that did not work the attacker started perusing the directories that did not require higher privileges. It can therefore be concluded that the user pertti had been compromised and all information that was inside any of the accessed directories is now compromised as well. Next, the apache access logs from the Intra were checked.

## 6.2 MediaWiki powered Intra server

There were only 75 apache access logs for Intra but looking closer at them showed someone from a previously not seen IP 10.0.2.32 had looked and tried editing the "Asiakkaat" page which contained the first clue about the function of the database in this company (Picture 27).

```
10.0.2.32 - - [03/Nov/2020:18:41:56 +0200] "GET /mediawiki/index.php?title=Asiakkaat&action=edit HTTP/1.1" 200 23537 "http://intra.pertinpelti.local/mediawiki/index.php?title=Asiakkaat" "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
```

Picture 27. Log entry showing 10.0.2.32 accessing the edit page function

Reading the logs further it seems the attacker found also the second clue, after which the attacker combed the site looking for the correct page (Picture 28).

```
10.0.2.32 - - [03/Nov/2020:18:44:11 +0200] "GET /mediawiki/index.php?title=K%C3%A4ytt%C3%A4j%C3%A4:Pertti&action=history HTTP/1.1" 200 27856 "http://intra.pertinpelti.local/mediawiki/index.php/K%C3%A4ytt%C3%A4j%C3%A4:Pertti" "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
```

Picture 28. Log entry showing the user accessing page's edit history

After some time, the attacker accesses the diff function on the correct page, getting the database credentials that were not removed correctly (Picture 29).

```
10.0.2.32 - - [03/Nov/2020:18:45:43 +0200] "GET /mediawiki/index.php?title=K%C3%A4ytt%C3%A4j%C3%A4:Pertti&diff=22&oldid=19 HTTP/1.1" 200 20058 "http://intra.pertinpelti.local/mediawiki/index.php?title=K%C3%A4ytt%C3%A4j%C3%A4:Pertti&action=history" "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
```

Picture 29. User accessing the diff function on the page that contained credentials

We can then conclude that the attacker has compromised the user pertti on the Intra wiki and gained access to a page which shows the database credentials. Therefore, we need to check the database logs if the attacker was able to use the stolen credentials.

### 6.3 Bogus Microsoft SQL server

For easier readability, the bogus Microsoft SQL server logs were run through a prettify script which turned the JSON log entries to JSON data that is more spaced out and therefore, easier to read.

The logs showed the program was able to successfully start listening on the default Microsoft SQL port 1433 and client from 10.0.2.32 connected to it (Picture 30) but the client did not have the correct credentials.

```

{
  "event": {
    "@timestamp": "2020-11-13T17:34:45.265",
    "msg": "Bind was succesful!",
    "description": "Listening on port: 1433"
  }
}
{
  "event": {
    "@timestamp": "2020-11-13T17:39:19.029",
    "eventID": 1,
    "description": "Connected with 10.0.2.32:43309"
  }
}

```

Picture 30. The successful binding to port 1433 and the first connection

After the failed authentication, the same IP made a new connection which then authenticates with correct credentials. The connection setup follows the simple method specified in Appendix 2, as such the pre-login message and the authentication request (which succeeds) can be seen. From this it is possible to deduce that the attacker was successful in gaining access to the bogus database. After the authentication, the environment change queries sent by the client are received and logged. (Picture 31)

```

{
  "event": {
    "@timestamp": "2020-11-13T17:40:37.229",
    "eventID": 2,
    "description": "Received pre-login message from 10.0.2.32:39057"
  }
}
{
  "event": {
    "@timestamp": "2020-11-13T17:40:37.229",
    "eventID": 4,
    "description": "Received login7 request from 10.0.2.32:39057"
  }
}
{
  "event": {
    "@timestamp": "2020-11-13T17:40:37.230",
    "eventID": 5,
    "description": "Succesfull login for user sa from 10.0.2.32:39057"
  }
}
{
  "event": {
    "@timestamp": "2020-11-13T17:40:37.230",
    "eventID": 6,
    "description": "Received SQL query SET QUOTED_IDENTIFIER OFF from 10.0.2.32:39057"
  }
}
{
  "event": {
    "@timestamp": "2020-11-13T17:40:37.246",
    "eventID": 6,
    "description": "Received SQL query SET TEXTSIZE 4096 from 10.0.2.32:39057"
  }
}
}

```

Picture 31. The full connection establishment and authentication for user sa

After the TDS connection has been established and authenticated it the client sends a recon query meant to show information about what tables the currently selected database has. However, since the server is not genuine it simply ignores the query by not responding to it and writes the full query to the log after which it closes the connection to the remote client. (Picture 32)

```
{
  "event": {
    "@timestamp": "2020-11-13T17:46:51.108",
    "eventID": 6,
    "description": "Received SQL query SELECT * FROM INFORMATION_SCHEMA.TABLES from 10.0.2.32:39057"
  }
}
{
  "event": {
    "@timestamp": "2020-11-13T17:46:51.109",
    "eventID": 11,
    "description": "Closing socket connection to 10.0.2.32:39057 in 21 seconds"
  }
}
```

Picture 32. The recon query and the following disconnect

In this instance the attacker was not able get any useful information even though the stolen credentials gave the rights to login to the bogus database.

## 7 CONCLUSION

The additional machines in the project were successful as they brought credibility and substance to the network experiment in the form of more services and information relaying options such as the MediaWiki powered Intra with its wiki pages filled with fake and mostly useless information. Their implementation went smoothly, although the original plan needed to be modified for the Office-PC by removing the rate limiting in UFW firewall since with it set, the machine was too well protected against the SSH brute force attack.

Although the implementation for the honeypot Microsoft SQL 2017 server had some setbacks such as the need to use tools, Python modules and transmission protocols that were not familiar to the author beforehand, the planning and implementation of the honeypot database server was still a success since one was able to connect to it with the genuine Microsoft SQL tool *sqlcmd* and send a Transact-SQL query, which was then logged for further analysis.

The building of a prototype honeypot Microsoft SQL server emphasizes that the full honeypot can be implemented in Python and the findings gained in Appendix 2 give pointers what a genuine Microsoft SQL server consists of, in terms of responding to client messages and the handling of said messages.

The possibilities for a bogus Microsoft SQL database are endless. One could for example build one where the server procedurally generates database structures for each day, authenticated connection and so on, a database with randomized values, or one housing data that seems real but is fake.

## REFERENCES

- [1] Reynaldo, M., 2014. Bare metal vs. virtual servers: Which choice is right for you?. Thoughts on Cloud, Available at: <https://www.ibm.com/blogs/cloud-computing/2014/07/25/bare-metal-vs-virtual-servers-choice-right/> Accessed 8 November 2020.
- [2] Elastic. 2020. Beats: Data Shippers For Elasticsearch | Elastic. Available at: <https://www.elastic.co/beats/> Accessed 8 November 2020.
- [3] Wazuh. 2020. Wazuh · The Open Source Security Platform. Available at: <https://wazuh.com> Accessed 8 November 2020.
- [4] Nazario, J., 2020. Awesome Honeypots. GitHub. Available at: <https://github.com/paralax/awesome-honeypots#readme> Accessed 30 October 2020.
- [5] Spitzner, L. 2002. Honeypots: Tracking Hackers. Boston: Addison-Wesley.
- [6] Mattson, J., 2016. Running Nested Vms |Vmware Communities. Communities.vmware.com. Available at: <https://communities.vmware.com/docs/DOC-8970> Accessed 31 October 2020.
- [7] Huybregts, C., 2020. AMD Nested Virtualization Support. TECHCOMMUNITY.MICROSOFT.COM. Available at: <https://techcommunity.microsoft.com/t5/virtualization/amd-nested-virtualization-support/ba-p/1434841> Accessed 31 October 2020.
- [8] Docs.oracle.com. 2020. 2.34. Nested Virtualization. Available at: <https://docs.oracle.com/en/virtualization/virtualbox/6.0/admin/nested-virt.html> Accessed 31 October 2020.
- [9] Pve.proxmox.com. 2020. Nested Virtualization - Proxmox VE. Available at: [https://pve.proxmox.com/wiki/Nested\\_Virtualization](https://pve.proxmox.com/wiki/Nested_Virtualization) Accessed 31 October 2020.
- [10] Wasserman, O. 2013. Kvm Forum – Red Hat Inc. Available at: <http://www.linux-kvm.org/images/e/e9/Kvm-forum-2013-nested-virtualization-shadow-turtles.pdf> Accessed 12 September 2020.
- [11] SP 800-53 Rev. 5. Security and Privacy Controls for Information Systems and Organizations. Maryland: National Institute of Standards and Technology.
- [12] 2020. [MS-TDS]: Tabular Data Stream Protocol. 30th ed. Microsoft. Available at: [https://docs.microsoft.com/en-us/openspecs/windows\\_protocols/ms-tds/b46a581a-39de-4745-b076-ec4dbb7d13ec](https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-tds/b46a581a-39de-4745-b076-ec4dbb7d13ec) Accessed 2 November 2020.



## Researching the Microsoft SQL 2017 server

The research environment had Windows Server 2016 Standard running the SQL 2017 Developer hence to be referred to as only SQL 2017 and the latest Kali version in a host-only networking mode on Oracle VM VirtualBox. Both machines were in the same 192.168.56.0/24 IP block.

On the SQL 2017 an account “\_jyrki” was made and setup as one of the database administrator accounts whilst on the Kali the *msodbcsql17* and *mssql-tools* packages from Microsoft’s Debian package depository was installed. The first package is a backend driver for the SQL which is used by the latter package of tools to connect to the SQL server.

Firstly, research needed to determine what the network traffic between a genuine SQL server and client looks like, therefore Wireshark was set to record the traffic between Kali and SQL 2017 whilst the SQL 2017 was logged into using the *sqlcmd* utility found among the *mssql-tools* package (Picture 33).

```
kayttaja@kali:~$ sqlcmd -U _jyrki -S 192.168.56.200
Password:
```

Picture 33. Showing the usage of sqlcmd

After a successful login, a basic select all query was done on a previously made table (Picture 34).

```
1> SELECT * FROM dbo.pellit
2> GO
ProductID  ProductName          ProductSize          Price
-----
          1 Liukuhihna pelti    30x50cm              34.0000
(1 rows affected)
```

Picture 34. The SQL query and it’s result

After these actions, the connection was ended by logging out, whilst the logging of network traffic was stopped and then saved. By looking at the saved traffic one could deduce that the SQL transmissions use Tabular Data Streams (TDS) to send and receive the data (Picture 35) with the actual login being hidden even for Wireshark.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000398235	192.168.56.107	192.168.56.200	TDS	154	TDS7 pre-login message
5	0.000723436	192.168.56.200	192.168.56.107	TDS	114	Response
7	0.030910333	192.168.56.107	192.168.56.200	TDS	380	TDS7 pre-login message
8	0.032176514	192.168.56.200	192.168.56.107	TDS	1250	TDS7 pre-login message
10	0.032518144	192.168.56.107	192.168.56.200	TDS	167	TDS7 pre-login message
11	0.032927106	192.168.56.200	192.168.56.107	TDS	125	TDS7 pre-login message
13	0.033040360	192.168.56.107	192.168.56.200	TDS	318	TLS exchange
14	0.033782652	192.168.56.200	192.168.56.107	TDS	540	Response
16	0.053515664	192.168.56.107	192.168.56.200	TDS	146	SQL batch
17	0.074599974	192.168.56.200	192.168.56.107	TDS	108	Response
19	0.084550390	192.168.56.107	192.168.56.200	TDS	130	SQL batch
20	0.084798581	192.168.56.200	192.168.56.107	TDS	103	Response
24	35.551072867	192.168.56.107	192.168.56.200	TDS	146	SQL batch
25	35.551809852	192.168.56.200	192.168.56.107	TDS	250	Response

Picture 35. Recorded traffic filtered to show only TDS traffic

Upon closer inspection one can see that the SQL 2017 sent an environment change message to the client (Picture 36) after the authentication has happened behind closed doors.

```

▼ Token - EnvChange
  Token length: 31
  Type: Database (1)
  New Value Length: 8
  New Value: tuotteet
  Old Value Length: 6
  Old Value: master
▼ Token - Info
  Token length: 122
  SQL Error Number: 5701
  State: 2
  Class (Severity): 0
  Error message length: 39 characters
  Error message: Changed database context to 'tuotteet'.
  Server name length: 15 characters
  Server name: WIN-QIMQ95A3855
  Process name length: 0 characters
  Line number: 1

```

Picture 36. The environment change from “master” to “tuotteet”

After the client has confirmed the receiving of the environment change message from SQL 2017 it sends two of its own environment change messages. First of them asks not to use quoted identifier (Picture 37) and second one to set the variable *textsize* to 4096 (Picture 38), to facilitate the writing of long Transact-SQL queries.

- ▼ TDS Query Packet
  - > Packet data stream headers
    - Query: SET QUOTED\_IDENTIFIER OFF

Picture 37. Set quoted identifier

- ▼ TDS Query Packet
  - > Packet data stream headers
    - Query: SET TEXTSIZE 4096

Picture 38. Set text size to 4096

After these it is possible to see the client's select all query (Picture 39), to which the SQL 2017 responds with a table (Picture 40).

- ▼ TDS Query Packet
  - > Packet data stream headers
    - Query: SELECT \* FROM dbo.pellit\n

Picture 39. The select all Transact-SQL query

- ▼ Tabular Data Stream
  - Type: Response (4)
  - > Status: 0x01, End of message
    - Length: 184
    - Channel: 53
    - Packet Number: 1
    - Window: 0
  - > Token - ColumnMetaData
  - ▼ Token - Row
    - ▼ Field 1 (1)
      - Data: 1
    - ▼ Field 2 (Liukuhihna pelti)
      - Length: 16
      - Data: Liukuhihna pelti
    - ▼ Field 3 (30x50cm)
      - Length: 7
      - Data: 30x50cm
    - ▼ Field 4
      - Length: 8
      - Data: 34,0000
  - > Token - Done

Picture 40. Table returned by the server

## Building the bogus Microsoft SQL database server

Before the research, the fact of default Microsoft SQL installs listening on port 1433 was known by the author, however what was not known was that the server uses Tabular Data Stream (TDS) as a message carrier over TCP connections, therefore also the bogus server needed to handle TDS data. In the simplest instance there are two messages, from both client and server, before the client can query the database (Figure 3).

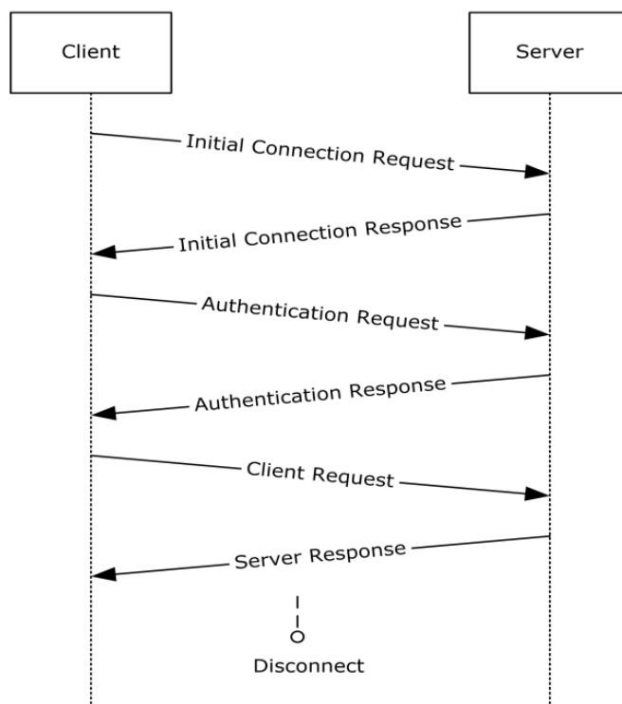


Figure 3. Client to server connection steps shown as a simple diagram ([MS-TDS]: Tabular Data Stream Protocol, 2020)

The technical documentation for the protocol states that all TDS packets contain a header type value as the first byte in the message, therefore the server must differentiate between these packets by looking at that value (Picture 41).

```

if recMsg[0] == 18 or recMsg[0] == 12:
    preloginHandler(recMsg, conn)
elif recMsg[0] == 16:
    loginHandler(recMsg, conn)
elif recMsg[0] == 1:
    sqlHandler(recMsg, conn)
  
```

Picture 41. Determine the packet's meaning with the header type

Inside these handler functions the received message is mapped into a Python dictionary object (which resembles JSON format), this allows the quick and easy to understand debugging and response generation to the client machine(s). For example, the mapped pre-login message from client would look like the following JSON dump.

```
{
  "type": 18,
  "status": 1,
  "lengthC": [
    94,
    0
  ],
  "spidC": [
    0,
    1
  ],
  "packetID": 0,
  "window": 0
}
{
  "version": 0,
  "pl_offset1C": [
    36,
    0
  ],
  "pl_option_length1C": [
    6,
    1
  ],
  "encryption": 0,
  "pl_offset2C": [
    42,
    0
  ],
  "pl_option_length2C": [
    1,
    2
  ],
  "instOpt": 0,
  "pl_offset3C": [
    43,
    0
  ],
  "pl_option_length3C": [
    1,
    3
  ],
}
```

Continues

```

"threadID": 0,
"pl_offset4C": [
  44,
  0
],
"pl_option_length4C": [
  4,
  4
],
"MARS": 0,
"pl_offset5C": [
  48,
  0
],
"pl_option_length6C": [
  1,
  5
],
"traceID": 0,
"payload": [
  49,
  0,
  36,
  6,
  0,
  85,
  0,
  1,
  255,
  17,
  6,
  0,
  1
]
}

```

The JSON dump shown above is cut into two sections the first one is the packet header whilst the second is the data section of the packet. The short explanation is as follows: in the case of the header's status field being 1 means that this packet is the end of the message, whilst the length tells how much data the whole TDS packet has in it (as bytes), this includes the header as well.

The data packet itself contains options like version and encryption, however when client sends the first pre-login message these and other such options are set to 0. Worth a note is also that when a client sends the pre-login message the traceID is serialized into the

payload portion of the packet, whilst in server response messages the payload contains the individual settings (on/off) for the previously mentioned options.

After experimenting with the response for the author can formulate a response with the encryption flag set to off and server TDS version as 14.0.1000. This response was then hardcoded into the program.

In the login part the server must compare allowed credentials and the credentials received, if those check out it sends a response containing an environment change with also the server name. However, before the comparing can even begin the password needs to be unscrambled since the TDS protocol requires the client scrambles the password before sending it. According to the MS-TDS: Tabular Data Stream Protocol's technical documentation (2020, 66) the client does this by first swapping "*the four high bits with the four low bits*" and then running XOR on the result with the binary 10100101, which is 165 as a decimal. Therefore, in the bogus server this needs to be run in reverse to get a password which can be compared to the one allowed.

After the authentication is finished the client can send its own environment change requests, like for example **set textsize to 4096**. These can be easily handled by the *sqlHandler* portion, since they came as SQL batch messages. However, this was just a small part of the TDS protocol (Figure 4) and further development would surely be needed to fool experienced attackers.

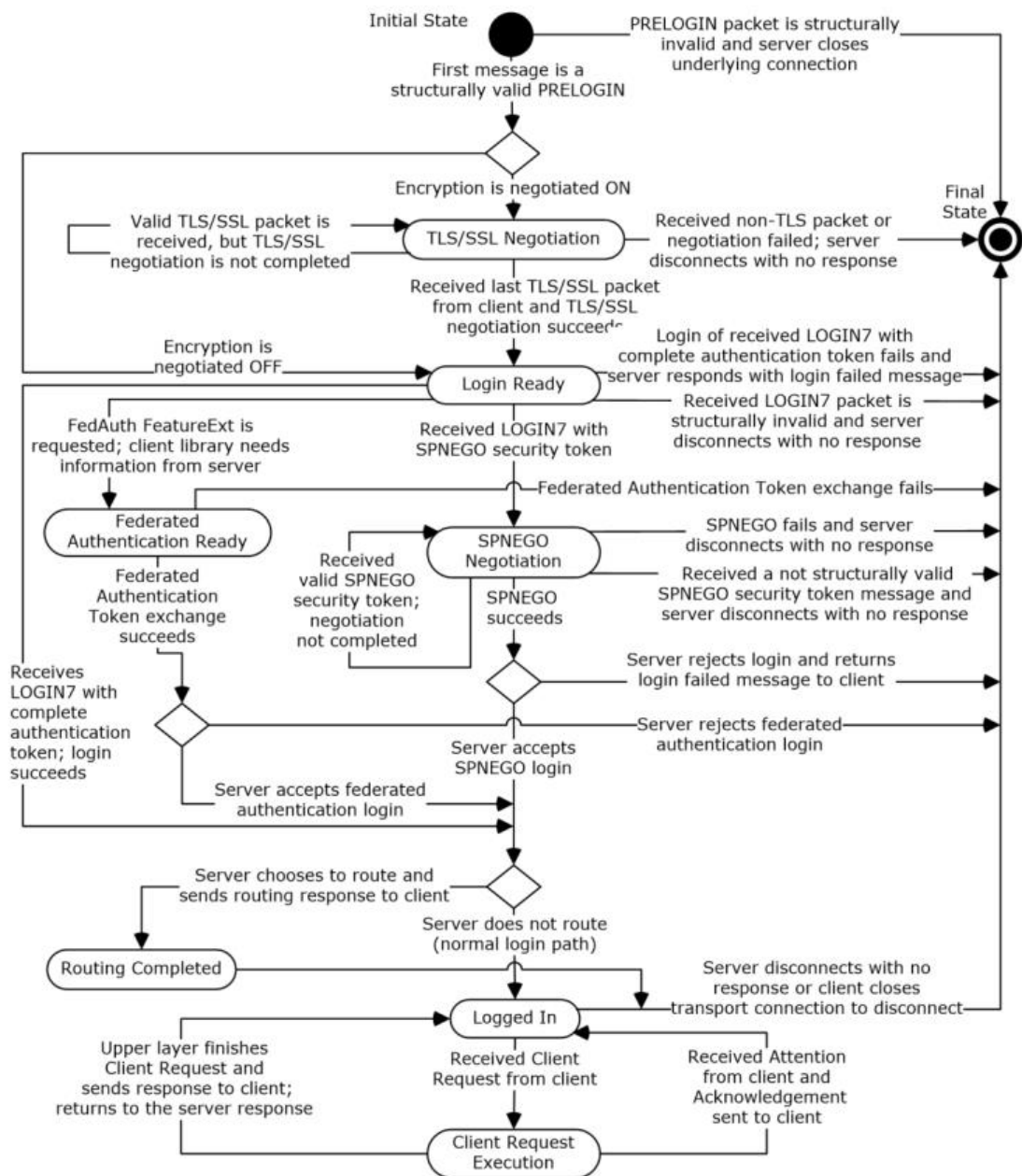


Figure 4. Flowchart of the server states in TDS protocol ([MS-TDS]: Tabular Data Stream Protocol, 2020)