



# **Data0: Next generation malware for stealing databases**

**Author:  
Cesar Cerrudo  
(cesar>.at.<argeniss>.dot.<com)**

## Abstract:

This paper it's about Data0, a fictitious (or not) simple PoC of new malware that after it's deployed on a computer in an internal network it will automatically hack database servers and steal their data. Several techniques used by Data0 will be detailed. Data0 will be targeting Microsoft SQL Server and Oracle Database Server two of the most used database servers. While Data0 could be used by the bad guys for evil purposes, it could also be used by security professionals and organizations to determine how strong networks, workstations, database servers, etc. are against this kind of attack.

This paper is not intended to be a cook book for cyber criminals, it's intended to show people that by implementing simple techniques malware can become "smarter" and cause a lot more damage in a very near future.

## Malware, botnets, etc.:

Let's first see what malware means:

"Malware is software designed to infiltrate or damage a computer system without the owner's informed consent. It is a portmanteau of the words "malicious" and "software". The expression is a general term used by computer professionals to mean a variety of forms of hostile, intrusive, or annoying software or program code." [1]

Malware is everywhere as virus, worms, spyware, trojan horses, etc. it's being more and more used for botnets [2], it is used to build botnets and also it's used by the botnets for performing their evil actions: sending spam, launching DoS attacks, installing adware or spyware, etc. Malware is spreading fast and affecting all the world, even Fortune 500 company networks are being affected and spamming people [3].

Luckily malware is not very smart yet but the bad guys are improving fast, lately botnets have incorporated p2p technology and rootkit functionality which makes them more difficult to shutdown, remove, detect, etc. Most affected users are home users that don't have much security protections in place but companies are also being affected, in the later case it is a more serious problem since we are talking about a security compromise in an internal network, in this scenario is where I'm going to focus on, I'm going to show techniques used by Data0 to automatically hack database servers and steal all the data once the malware is deployed on a computer in an internal network.

## Getting malware in:

Malware can be deployed in a computer by opening an email attachment, by downloading and installing software from the web, by exploiting a vulnerability, etc. if you don't have proper protections in place and you are connected to a network then sooner or later you get malware on your computer, this is a fact.

Nowadays you need a lot of protection: antivirus, personal firewalls, anti\* software, etc. but even with all that protection sometimes users are still vulnerable to attack, lots of Microsoft Internet Explorer and Office 0days have plagued Internet lately used by the bad guys for owning computers around the world.

So let's face it, "getting malware in" is not big deal, the thing is what the malware will do after it's on a network and what you can do to protect from it.

## Designing and building Data0:

Once the malware is in, it can start with its evil actions. Let's start to design and build our malware called Data0 with all the needed techniques so it will be able to automatically hack and steal databases. The idea is to get a working PoC code with basic but powerful and enough functionality for performing desired actions.

### Requirements:

- Data0 could install a rootkit to avoid being detected, removed, etc.
- Data0 should be able to get needed information to connect to database servers and also be able to deliver stolen data without needing high privileges.
- Data0 should get information to connect to database servers in all possible ways.
- Data0 should be as stealth as possible, stealth level should be configurable and stealth techniques will be preferred.
- Data0 end goal will be to steal data from database servers and deliver it to attacker in any possible way.
- Data0 should be able to clean itself if needed after performing its evil actions.
- Data0 must target Microsoft SQL Server and Oracle Database Server.

### Data0 functionality can be grouped in four different stages:

- **Discovery**
- **Attacking**
- **Sending data back**
- **Covering tracks**

#### **Discovery:**

In order to hack database servers Data0 needs to know where they are and also it needs to find all the required information to be able to connect to them: user name, password, database name, etc.

1. Getting information from ODBC configured data sources.
2. Getting information from processes connected to database servers.
3. Getting information from network discovery.
4. Getting information by sniffing the network.

#### **1- Getting information from ODBC configured data sources:**

This information can be easily get from Windows registry for User and System DSNs (Data Source Names) and from files for File DSN, the information can include: server, user name, authentication type, etc. The only problem is that Data0 won't be able to get the password in this way so it will have to guess or brute force the password later when trying to connect, but if the database server is SQL Server and it's configured to use Windows authentication then it will be able to connect to the database server just using current user Windows account.

User DSN information can be found in the following registry key:  
*HKCU\SOFTWARE\ODBC\ODBC.INI\ODBC Data Sources*

System DSN information can be found in the following registry key:  
*HKLM\SOFTWARE\ODBC\ODBC.INI\ODBC Data Sources*

File DSN information is saved in files, location of the files can be found in the following registry key:

*HKLM\SOFTWARE\ODBC\ODBC.INI\ODBC File DSN*

## **2- Getting information from processes connected to database servers:**

ODBC data sources are used by applications to connect to database servers, sometimes applications will ask the user for the password and other extra needed information and then connect. What Data0 can do to identify processes that are connecting to database servers is to continuously monitor for outbound connections to known database server ports, when a connection is identified then Data0 can get the process ID that is doing the connection, the process will be running in the security context of current user so Data0 will be able to open that process and read its memory without problems, by reading process memory it can search for user name, password, database name, etc.

If a connection to port 1433 is detected Data0 will know that it's a connection to SQL Server, with the process ID associated with the connection Data0 will open the process and search process memory for the needed connection information.

There is a small problem with Oracle, on default configuration clients don't connect directly to Oracle server port, first the client connect to Oracle Listener that by default listen on port 1521 and then the Listener forward the connection to a dynamic port created by Oracle, at first sight it seems that the technique won't apply to this scenario but that's not true, because a connection is first established to listener the malware can look for connections with TIME\_WAIT status on Listener port or any other status, TIME\_WAIT status remains like 4 minutes so the malware has a good chance to get it, but the PID won't be available so Data0 can get the PID by looking at active connections to the same IP address that had the connection on port 1521, it could get the wrong PID if the local host is connecting to many services at the same IP but it's not common so there are high chances to succeed and to get the right PID. When a process connected to a database server is found it's pretty trivial to look for user name, password, etc. strings in process memory.

In order to get the list of TCP/IP connections and the associated processes the next API is used:

*AllocateAndGetTcpExTableFromStack()*

this API returns the following information: Local IP, local port, remote IP, remote port, status of the connection and process ID associated with the connection.

Connection strings used by applications use the following common parameters:

Data Source, Server or DBQ: for setting the server name, database name, etc.

User ID or UID: for setting the database user name.

Password or PWD: for setting the database password.

All those parameters with their corresponding values are searched in memory on all the applications connected to SQL Server or Oracle, when the values are found they are saved for connecting later when starting to attack.

Another possible advanced technique is to get database connection handles used by applications in data access APIs and use already opened connections, in this way Data0 won't need to get any information nor connect to the server, it will just reuse existing connections. To achieve this goal some data access APIs must be hooked and some code injected in target process, the problem with this is that connections can be closed at any time by the application due to many possible reasons and this mean that more APIs will need to be hooked making more complicated the use of this technique. Also applications can use different data access APIs and all the different APIs must be had in mind when using this technique.

### 3- Getting information from network discovery:

The last and noisy way of getting information is to send database server discovery packets and/or scan the network for database servers. SQL Server has a discovery service listening by default on UDP port 1434 that can be queried for information. Scanning the network for database servers will be very noisy and it should be left as a last resource. Getting only server name or IP doesn't help much since Data0 will have to brute force for default database user names and passwords in order to connect.

### 4- Getting information by sniffing the network:

This is optional and it's not a preferred technique since it will need high privileges and nowadays most networks are switched so capturing useful traffic will be noisy.

### **Attacking:**

If Data0 have enough information then it can connect to database servers, but in case it doesn't have enough information it will have to guess or brute force that information, for instance if it wants to connect to SQL Server using native authentication and it just has user name and the server name or IP then it will need to brute force the password or if it just has the server name or IP it will need to guess or brute force both user name and password. Of course if it can connect using Windows authentication then there is no need to brute force. If it want to connect to Oracle more information such as: user name, password, server name or IP and service name (database name) could be needed so if it doesn't have the service name it will need to guess or brute force it.

Once it connects it needs to determine how much privileges it has in the database server, if it has low privileges then in order to be able to steal all the data available it will need to elevate privileges, version and patch level must be determined for proper selection of exploitation method. If the server is Oracle it will be very easy to own it, a known vulnerability could be exploited or a 0day exploit could be ran and the server be owned no matter if it's up to date with patches, but if the server is SQL Server and it has been patched at least once in the last three years then it will be difficult to elevate privileges by exploiting a vulnerability.

At this stage Data0 can also get information about other database servers by looking at replication servers, linked servers, etc. and then try to connect to those servers by using own database functionality or from Data0 itself. If connection is successful all steps in this stage are repeated in each server.

After getting the highest possible privileges Data0 can start mining the data, it can run several queries for looking for specific data or for just indexing all the available data. It can just steal data by running queries and storing the results for sending them later to attacker, it can also backup a complete database and store it. Data stored in host computer could be encrypted just in case someone finds it, but if the Data0 has installed a rootkit then the files generated by Data0 shouldn't be easy to find.

For instance on SQL Server the next queries can be used to get database structure information:

```
--Get linked servers:
```

```
select srvname from master.dbo.sysservers
```

```
--For each linked server try to get all databases:
```

```
select name from server.master.dbo.sysdatabases
```

--For each database get all the tables:

```
select name from server.databasename.dbo.sysobjects where xtype='U'
```

--For each table get all the columns:

```
select name from server.databasename.dbo.syscolumns where id=object_id('tablename')
```

After Data0 knows the database structure it can start looking for interesting data:

--Get tables with interesting names:

```
select name from databasename..sysobjects where xtype='U' and name like = '%customer%'
```

```
select name from databasename..sysobjects where xtype='U' and name like = '%salary%'
```

```
select name from databasename..sysobjects where xtype='U' and name like = '%credit%'
```

```
select name from databasename..sysobjects where xtype='U' and name like = '%payment%'
```

```
select name from databasename..sysobjects where xtype='U' and name like = '%sales%'
```

```
select name from databasename..sysobjects where xtype='U' and name like = '%account%'
```

--Sometimes table names are not descriptive so columns with interesting names can be searched:

```
select object_name(id) TblName, name ColName from databasename..syscolumns where id in
(select id from databasename..sysobjects where xtype='U') and (name like '%user%' or name
like '%pass%')
```

```
select object_name(id) TblName, name ColName from databasename..syscolumns where id in
(select id from databasename..sysobjects where xtype='U') and (name like '%credit%' or
name like '%card%')
```

```
select object_name(id) TblName, name ColName from databasename..syscolumns where id in
(select id from databasename..sysobjects where xtype='U') and (name like '%ccard%' or
name like '%social%')
```

Also specific queries can be built to search and detect columns with credit card numbers, social security numbers, etc.

When a table with interesting data is found it can be dumped and saved or depending on privileges in database server Data0 could backup a complete database and save it for sending it later to attacker. Another option is to just make an index of the database structure that can be used by attacker in next stage for running custom queries.

For keeping access to database server Data0 could also create new users and/or install a backdoor, then deploy database rootkits for hiding new created users and/or installed backdoors [4].

***Sending data back:***

After data is retrieved and stored by Data0, it's needed a way to send it back to attacker, the simplest way it's to send it by HTTP, email, FTP, etc. Another option is to provide to the attacker an interactive connection to the database acting as a proxy between the attacker and the database server so the attacker can run any commands at will and get the results back. It could be possible to use p2p technology so the attacker can send a command to the Data0 p2p net and get results back from all owned databases servers, these commands could be for listing all databases, all tables, columns, etc. for looking for data that matches some expressions such as: \*credit\*, \*pass\*, \*mail\*, \*secret\*, etc. Data0 can be used to build a botnet that will allow control over hundred of compromised databases, think about running a query in hundred of database servers at the same time and get the results back.

An advanced option would be to use a covert channel for sending data back, for instance available IM functionality could be abused.

***Covering tracks:***

Some attacks scenarios could require to clean all possible Data0 related evidence in the host computer after all evil actions are performed. It could be programmed for auto-destruction, this can be done by erasing all malware related files and then wiping free disk space. Activities on database servers can also be cleaned to avoid detection by database administrators but this will require complete control over database servers.

***Data0 PoC:***

Data0 was developed by the author during this research with limited functionality just as a simple PoC to put in practice some of the techniques explained here, because it can be used for evil purposes it won't be released to general public. Anyways it's pretty simple to build a tool that uses the techniques explained here.

***Recommendations:***

In order to protect against this kind of attacks companies must deploy third party personal firewalls on desktop computers and third party real time database activity monitoring products to protect database servers. Also systems must be kept up to date with patches and database servers must be audited periodically for security vulnerabilities.

**Conclusion:**

As we have seen it is not difficult to build a semi “smart” malware that will automatically hack database servers and steal data. Targets of this new kind of malware will be organizations so they should be worried and start investing for securing and protecting their database servers as well as desktop computers. Organizations could also use the same techniques used by this kind of malware on their own networks in a controlled way to determine how much their networks are susceptible to these attacks.

Hopefully all this research will open organizations eyes and make them to increase their defenses to resist future attacks.

**References:**

[1] Malware definition

<http://en.wikipedia.org/wiki/Malware>

[2] Botnet definition

<http://en.wikipedia.org/wiki/Botnet>

[3] Malware on Fortune 500 companies

[http://blog.washingtonpost.com/securityfix/2007/03/fortune\\_500s\\_unwittingly\\_becom.html](http://blog.washingtonpost.com/securityfix/2007/03/fortune_500s_unwittingly_becom.html)

<http://www.support-intelligence.com/blog/>

[4] Hacking Databases for Owning your Data

<http://www.argeniss.com/research/HackingDatabases.zip>

## About Argeniss

Argeniss is an information security company specialized on application security, we offer services such as software auditing, penetration testing and training.

Contact us

Buenos Aires 463  
Parana, Entre Rios  
Argentina

E-mail: [info@argeniss.com](mailto:info@argeniss.com)

Tel: +54-343-4231076  
Fax: 1-801-4545614