Hacker Intelligence Initiative, Monthly Trend Report #18

Assessing the Threat Landscape of DBaaS

1. Executive Summary

The threat landscape has changed dramatically over the past decade – from pranksters to extremely sophisticated operations. Today, governments, organized crime, and hacktivists are actively engaged in stealing high-value data. While many organizations keep their prized information stored in their internal networks, others choose to host their data externally, in the cloud. Obviously, neither option guarantees security from cyber criminals.

Database as a Service (DBaaS) offers legitimate businesses a self-service model for provisioning databases, without the cost of setting up servers and burdening their own IT teams. However, DBaaS also offers less legitimate businesses such as criminals a platform for hosting their dubious servers. Using DBaaS is an easy way for someone to set up a Command and Control (C&C) server, store stolen data, and enjoy full anonymity while doing so. The low monthly startup cost is definitely affordable for even small time crooks.

Apart from offering criminals a cheap and safe playground, DBaaS itself introduces new security issues. When an organization's internal data is stored in the cloud, an attacker no longer needs to gain access to the organization's network, before compromising its database. This is compounded when a hacker opens his own account with the same cloud service. By doing that, the attacker can gain privileges or use a vulnerability to compromise all of the hosted data. DBaaS gives illegitimate users (i.e., cyber criminals) easier access to data, from both inside and outside the service.

Key findings in this report are:

1. Database-savvy modular malware platforms are an imminent threat. Malware is now capable of connecting to both local and remote databases to retrieve, manipulate, and ex-filtrate information
2. Some families of malware use DBaaS for botnet management (e.g., Command & Control as well as Dropper functionality)
3. Cloud databases are prone to attacks via both privilege escalation and exposed vulnerabilities, as opposed to on-premise databases, which are mostly compromised via privilege escalation

Our main conclusions are:

1. Cyber criminals have adopted DBaaS as infrastructure for both infection and data exfiltration
2. Risk management for cloud databases needs to be different than for on-premise databases
3. DBaaS, breaches that begin with one customer's database may hop to others

2. DBaaS as a Malware Service

2.1 Overview

In June 2013 we analyzed a malware sample that comes from Brazil. It was referenced from a Phishing email in Portuguese (shared on Google Docs), and, as it turns out, uses a very popular MSSQL hosting service for its C&C functionality as well as its storage ("drop") server. Searching our databases we found an identical malware sample in August 2012 and an earlier version dating back to December 2011. The malware is identified by popular AV engines as Trojan-banker.win32.agent.pqg.

Rather than reverse engineer the sample's code, we took a black box approach, focusing on its communications with the web. We discovered that it was using a cloud database service which led us to examine the contents of the database and, with the help of the service provider, the contents of similar databases.

Overall, we found five different C&C databases and two storage databases hosted with the same service provider. Two of them were found on the same server hosting the original C&C database. The C&C databases contained a list of infected machines and downloadable binaries .We grouped the databases according to the binaries they contained. From here on, we refer to them as CC1 and CC2. We reference the databases used for storing stolen information as Drop1 and Drop2. We are able to correlate machines from CC1 and CC2 with stolen data in Drop1. Drop2 has stolen data from unknown sources, but through the stolen data coming from accounts that belong to the same bank mentioned in the initial phishing email which contained the malware, we can associate it with the same malware family.

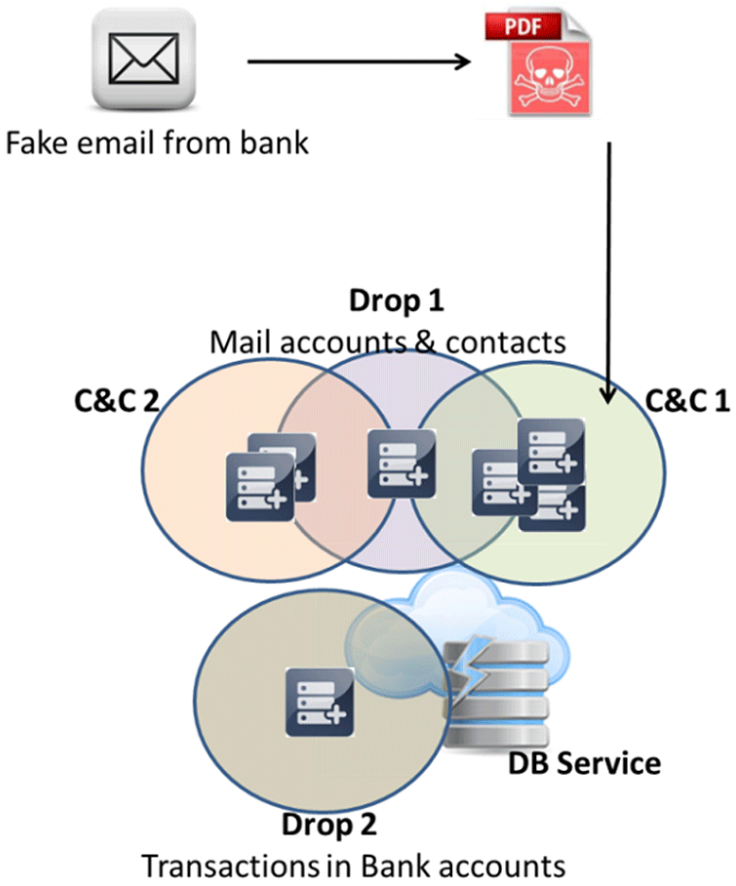
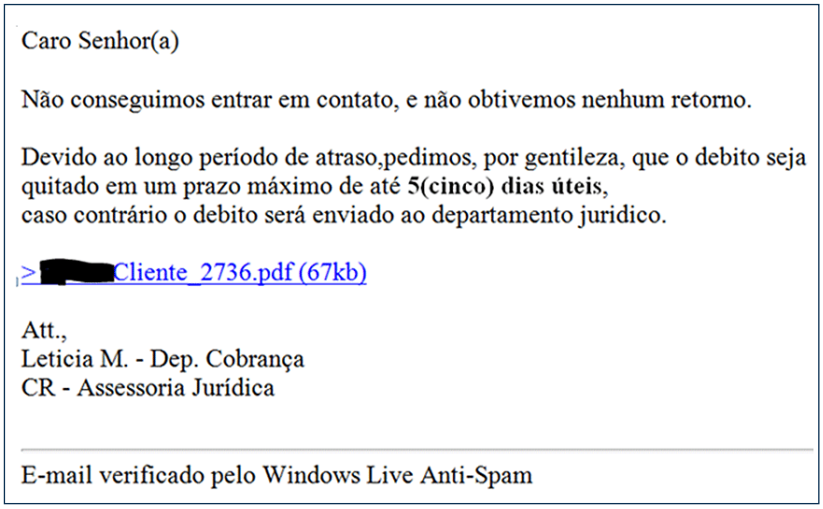


Figure 1: Drop and C&C Servers

2.2 Infection flow

In the following example, the infection starts with some classic social engineering. The victim receives an email (in Portuguese) warning of an unpaid debt to a well know bank in Brazil. To further assure the victim of the "legitimacy" of the message, an "email verified by Windows Live Anti-Spam" statement was added at the bottom of the message. In the message there is a link to an alleged PDF file, presumably detailing the debt.



Caro Senhor(a)

Não conseguimos entrar em contato, e não obtivemos nenhum retorno.

Devido ao longo período de atraso,pedimos, por gentileza, que o debito seja quitado em um prazo máximo de até **5(cinco) dias úteis**,
caso contrário o debito será enviado ao departamento juridico.

> [redacted]Cliente_2736.pdf (67kb)

Att.,
Leticia M. - Dep. Cobrança
CR - Assessoria Jurídica

E-mail verificado pelo Windows Live Anti-Spam

Figure 2: Phishing email

The referenced file is however actually a screen saver file, which is practically an executable. The file was created in such a way that it displays an icon similar to a PDF document.

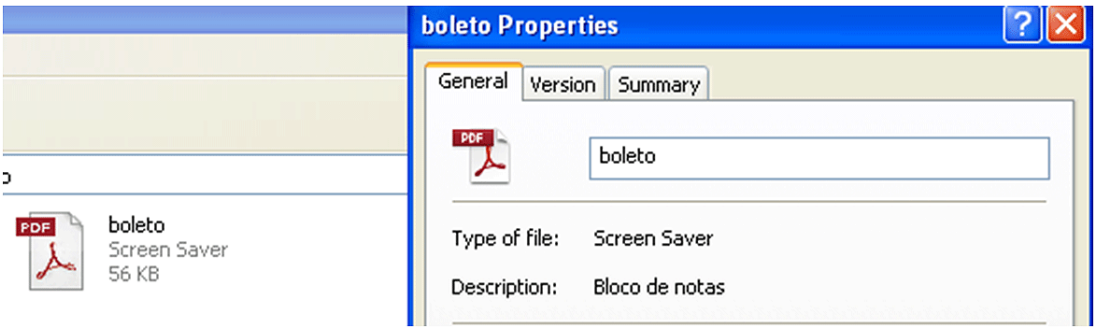


Figure 3: Payload

When the victim downloads and tries to open the PDF file, malicious code is executed. The malware initializes a connection to a remote (hosted) MSSQL database server. The malware uses the local SQLOLEDB provider for this communication. The logon process to the database is done over SSL, making the logon credentials encrypted. As opposed to the host name of a regular web C&C server, the database name is not trivial to extract from the protocol.

Curiosity got the best of us, so we used a man-in-the-middle technique to capture the login credentials used by the malware. Figure 4 shows a database access message sent by the malware, after we applied our MITM technique. Once we obtained those credentials, we were able to connect to the database and explore its contents using a query tool.

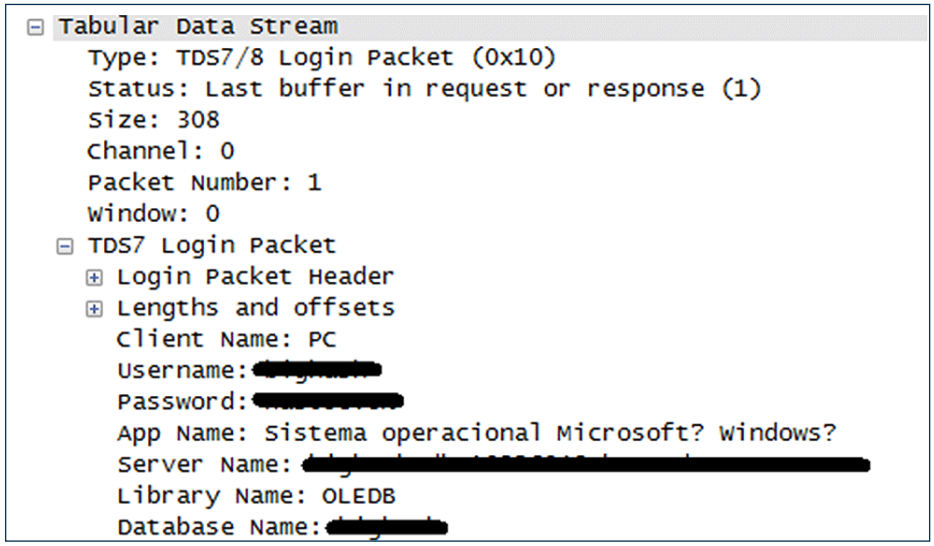


Figure 4: Login Packet

The malware's bootstrap code invokes user defined stored procedures within the database after a connection is established with the DB server. The first stored procedure is shown in Figure 5. It is "retorna_dados" (retrieve data in Portuguese) which retrieves three binary payloads from a table called a "carrega":

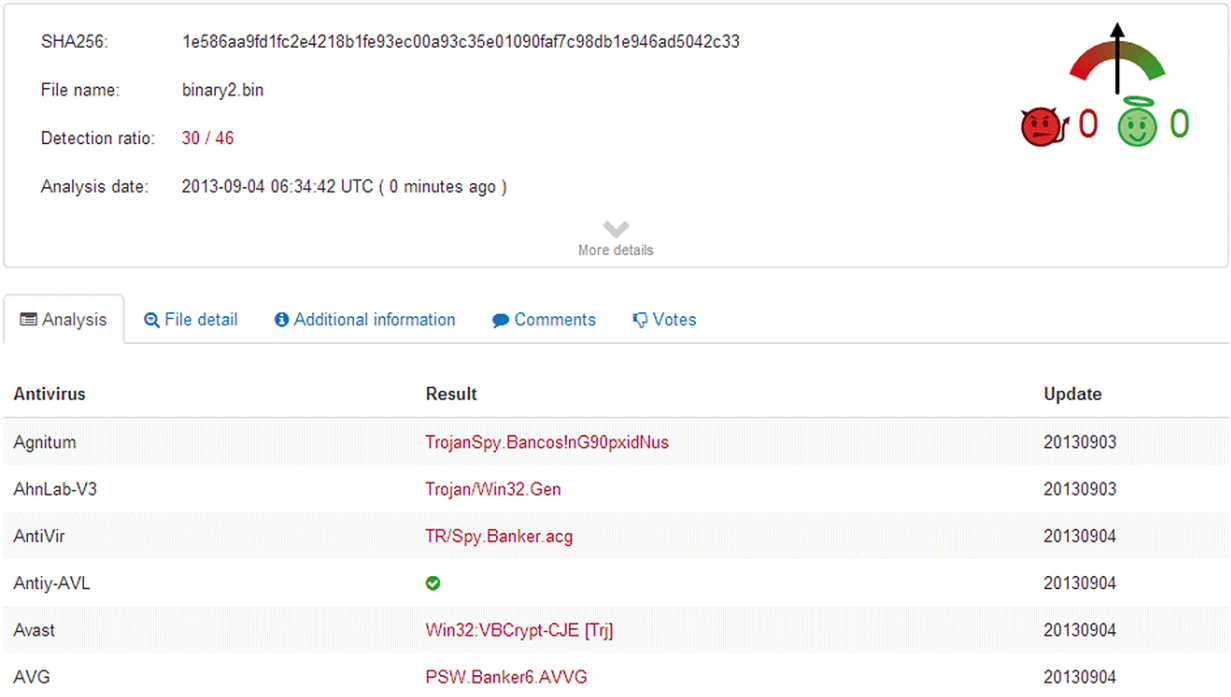
```
begin

        select * FROM tbl_carrega WHERE id = 1
end
```

Figure 5

The table entry contains three different binaries. The bootstrap code selects one of the binaries by its column number and saves it in the %AppData% folder of Windows as govision.dll, for loading and execution later on.

About Half (25/46) scanners in VirusTotal do detect the initial payload as malicious and categorize it as a Trojan Banker module. It was originally submitted on May 2013. We also came across an analysis of a different sample malware from the same family which used a different C&C database on the same hosting service.

Virus Total scanning of govision.dll also showed it to be categorized by some as a Trojan Banker.



SHA256: 1e586aa9fd1fc2e4218b1fe93ec00a93c35e01090faf7c98db1e946ad5042c33

File name: binary2.bin

Detection ratio: 30 / 46

Analysis date: 2013-09-04 06:34:42 UTC (0 minutes ago)

More details

Analysis | File detail | Additional information | Comments | Votes

Antivirus	Result	Update
Agnitum	TrojanSpy.Bancos!nG90pxidNus	20130903
AhnLab-V3	Trojan/Win32.Gen	20130903
AntiVir	TR/Spy.Banker.acg	20130904
Antiy-AVL		20130904
Avast	Win32:VBCrypt-CJE [Trj]	20130904
AVG	PSW.Banker6.AVVG	20130904

Figure 6: Virus Total results

The other two binaries in "carrega" were less notorious, receiving 4/47 and 10/47 detection rate.

The second stored procedure is "add_avs" (code for the stored procedure is shown in Figure 7).

```
begin
        if not exists(select id from tbl_avs where id_pc = @id_pc)
            begin
                insert into tbl_avs VALUES(@id_pc,@versao,@windows,@ff,@ie,@ins001,@ins033,@ins104,@ins341,@ins237,@data)
            end
        else
            begin
                update tbl_avs set versao = @versao,windows = @windows,ff = @ff,ie = @ie,ins001=@ins001,ins033=@ins033,
                ins104=@ins104,ins341=@ins341,ins237=@ins237, data = @data where id_pc = @id_pc
            end
end
```

Figure 7

This stored procedure is used for registering the specific bot agent with the botnet C&C, by sending the following information: identifier, version, Windows OS, browsers (Explorer and FireFox), date and some more ambiguous fields (ins###). The "add_avs" stored procedure looks for an identifier of the agent in the "avs" table, and if that identifier is not found, it adds a new record to the table. We were able to determine that the agent identifier is actually the serial number of the "C" drive volume on the infected machine. Figure 8 lists some records of the "avs" table from the database.

id_pc - vol C	version	windows	FIrefox	Internet Explorer	ins001	ins033	ins104	ins341	ins237	date
[redacted]	0	Microsoft Windows 7 Home Basic (32bits)	1	0	0	0	0	0	0	Jun 10 2013 12:35AM
[redacted]	1.1.0.8	Microsoft Windows 7 Professional - 32bits	0	0	0	0	0	0	0	Jun 9 2013 10:30PM
[redacted]	0	Microsoft Windows 7 Ultimate (32bits)	1	0	0	0	0	0	0	Jun 9 2013 9:26PM

Figure 8: Sample entry in "avs" table

2.3 C&C database contents

By using our MITM technique we were able to analyze the contents of the database used by the botnet. We were also able to extract enough information that would allow the service provider to identify some more databases of the same nature that were hosted on that same server. By collaborating with the service provider, we were able to track down a number of additional databases used by the same family of malware that were active at the same time. The databases had the same table structure and contained the same set of user defined stored procedures. Each database had a different name, and their "avs" table contained mostly disjoint sets of agents. Some agents were found in multiple tables. This could be due to multiple infections of the same machines or explained by the existence of test machines (either by researchers or the criminal group itself). All in all, we were able to analyze the contents of six databases: **CC1.db2**, **CC1.db1**, **CC1.db3**, **CC2.db1**, and **CC2.db2**. We split them into two groups according to the different binaries stored in their "carrega" table:

C&C1: CC1.db1, CC1.db2 and CC1.db3

C&C2: CC2.db1 and CC2.db2

Apart from the binaries that differentiate these tables, minor differences can be observed in the format of the stored data. For example, the **CC1.db3** and **CC1.db2** databases (C&C1) always store the OS CPU type followed by a hyphen, while the **CC2.db2** and **CC2.db1** databases (C&C2) store CPU type in parenthesis. The **CC1.db1** database (C&C1) uses both formats. These minute distinctions are the sign of slight revision changes between the different botnets that make up the same family.

For example, we found one machine that was stored across all databases:

version	windows	ff	ie	ins001	ins033	ins104	ins341	ins237	data	database
	0 Microsoft Windows XP Professional(32bits)	0	0	0	0	0	0	0	Jun 10 2013 10:53AM	CC1.db1
	0 Microsoft Windows XP Professional - 32bits	1	6	0	0	0	0	0	Jun 10 2013 6:41AM	CC1.db2
1.1.0.8	Microsoft Windows XP Professional - 32bits	0	0	0	0	0	0	0	Jun 10 2013 2:37AM	CC1.db3
	0 Microsoft Windows XP Professional(32bits)	0	0	0	0	0	0	0	Jun 10 2013 2:49AM	CC2.db1
	0 Microsoft Windows XP Professional(32bits)	1	0	0	0	0	0	0	Jun 9 2013 9:26PM	CC2.db2

Figure 9: Infected machine across all C&C databases

This machine was infected multiple times in a very close range; however, some saved data has changed between infections like FF, IE and version column. These changes can be explained by differences between malware samples that either infected, or were tested on the same machine. It is also possible, in some sort of risk and resource management scheme, that the same malware is configured when it's compiled to reach different databases.

In total, about 350 compromised machines were registered in the databases we've analyzed. All of the infections occurred between February and June of 2013. Figure 10 shows the number of infected machines per database.

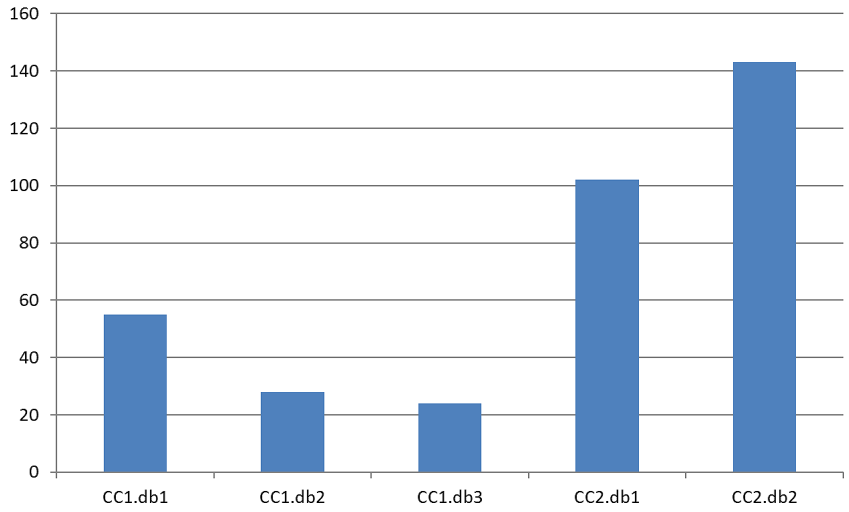


Figure 10: Infections per database

Out of the total number of infected machines, 330 (95%) were infected by this malware between June 3rd and June 10th, suggesting that the earlier infections are actually QA tests performed by the attackers (the databases were removed after June 10th). It seems as though attackers launched a number of simultaneous smaller infection campaigns that week, rather than a single large one. Each of the smaller campaigns probably had a slightly different message format and pointed to a slightly modified version of the malware that uses one of the target databases. This theory is in line with the fact that the infection message we intercepted was not detected by any anti-spam mechanism – suggesting that the campaign size is small enough to fly below the radar. Figure 11 shows the number of infected machines by date (note that the axis is not linear):

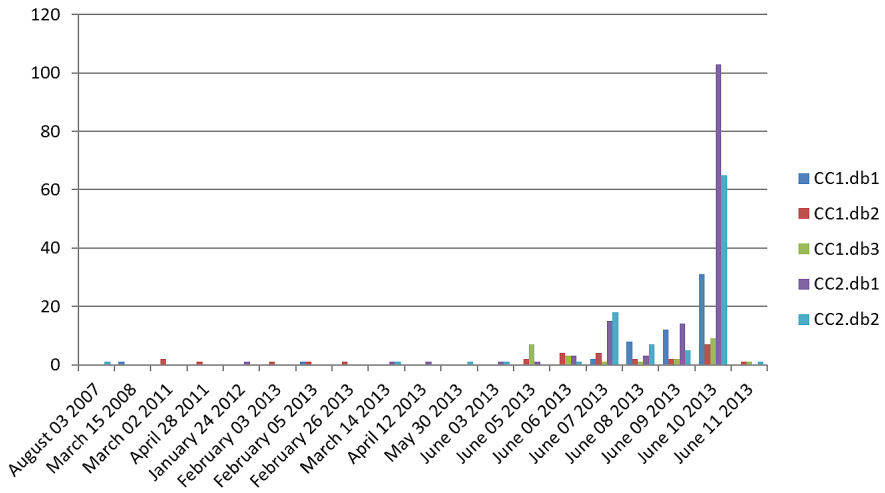


Figure 11: Number of infections by date

Figure 12 shows the distribution of infected operating systems. Oddly enough, about 54% of infected machines are using the old "Microsoft Windows XP" operating system.

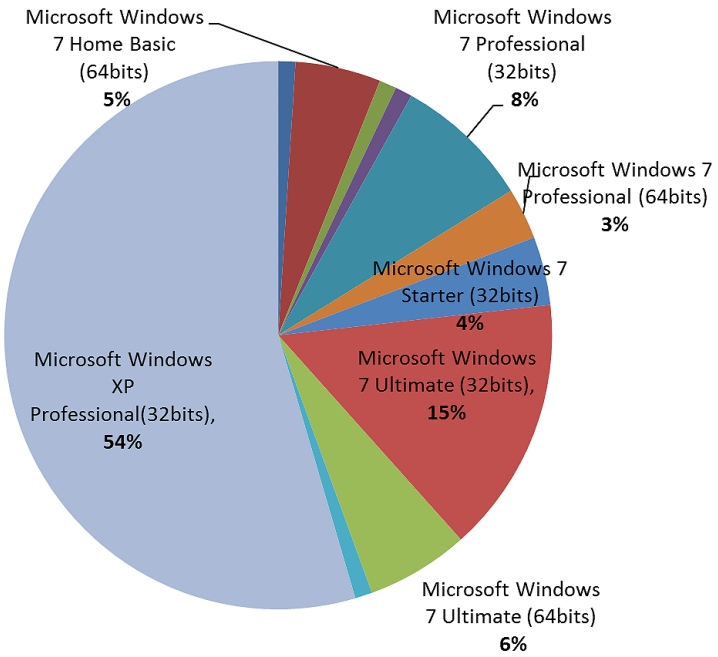


Figure 12: Distribution of infections by OS

Most machines had operating systems indicative of enterprise machines (Figure 13): Windows Server 2003/2008, Windows Vista Business, Windows 7 Enterprise and Windows Professional (XP/7). These compromised operating systems might indicate that internal company resources were also compromised.

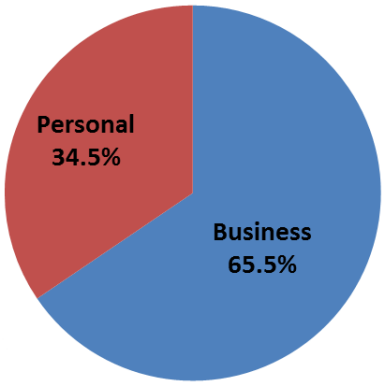


Figure 13: Business vs Private OS

2.4 Drop server contents

While monitoring the malware and its behavior, we were able to obtain the contents of two of the drop servers, we call them DROP1 and DROP2.

The DROP1 database contains a table named "dados". The "dados" table stores compromised email accounts information, detailing user name and password, SMTP & POP3 servers and a list of contacts. The account information stored in the table was extracted from either Outlook or Outlook Express installed on the compromised machine. A sample entry can be seen in Figure 14. We handpicked a few stolen accounts and found out that some of them were blocked, due to spamming by their mail service provider.

A sample entry:

```
Account Name: pop.*****.com
SMTP Display Name: c*****.as******2@*******.com.br
SMTP Email Address: c*****.as******2@*******.com.br
SMTP Server: smtp.googlemail.com
POP3 Server: pop.googlemail.com
POP3 User Name: c*****.as******2@*******com.br
POP3 Password2:  *********
Account Name: Active Directory
Account Name: Serviço de diretório na Internet Bigfoot
Account Name: Serviço de diretório na Internet VeriSign
Account Name: Serviço de diretório na Internet WhoWhere
CONTATOS
.....List of contact emails.....
```

Figure 14

The stolen information dates to the period from April 2013 to June 10th 2013. It refers to roughly 600 infected machines and 767 compromised accounts. It contains thousands of stolen contact records. We cross-referenced the agent identifier values from this database, with those found in the C&C databases and were able to find seven matches (throughout the C&C databases). This correlation reveals a few things about the structure of the botnet. First, it shows that these "drop servers" are in fact related to the C&C servers and that this is indeed a closely related family of botnets operated by a single organization. Second, since we were not able to correlate all infected machines with our C&C databases we suspect that the actual size of the botnet (number of C&C databases) is much larger than we have been made aware. Last, but not least, it shows that the botnet employs many more C&C databases than drop servers. Agents operated by different C&C databases send the stolen data to a shared drop server. This is in line with our impression that infection is achieved by multiple smaller campaigns rather than a single large one. Such architecture makes it much harder to take down the entire botnet, as it is highly dispersed between multiple C&C databases.

Other important data the email accounts can give us is the geographical distribution of the malware:

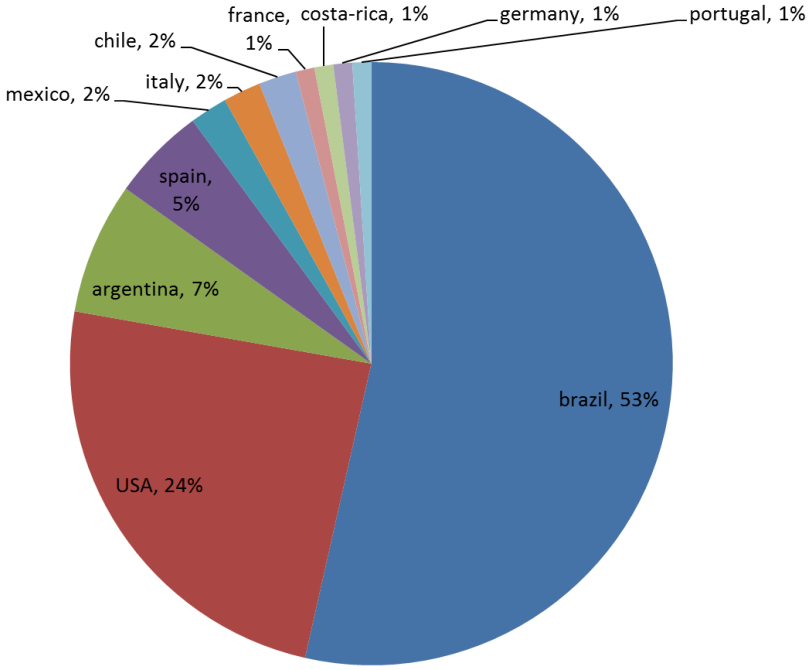


Figure 15: Stolen email accounts by country

Not surprisingly, South America, mainly Brazil and Argentina, comprise about 60% of infections. Additionally many other countries in South America are represented in the data set, including Chile, Venezuela, Colombia, Uruguay, and Peru.

The DROP2 database contains 2 tables: "Juju" and "Titulo". The "Juju" table contains stolen banking activity information taken from infected machines. The information was obtained from the same banking application that was targeted by the Phishing campaign that prompted our research. Each record includes a serial number, some identification of the machine, unstructured data, and capture timestamp. As can be seen in Figure 18, some of the records contain machine details in a formatted manner, including the disk "C" volume identifier we saw in the other databases. We were not able to correlate any of the machines to the contents we obtained from the C&C databases.

The database consists of more than 400 entries from 12 different machines. All machines were associated with the same Brazilian bank, but it is worth noting however, that the bank itself was not breached. This bank is dedicated to corporate accounts. Targeting corporate accounts is not surprising because they hold greater financial rewards for offenders.

Timeline of those entries is depicted in Figure 16.

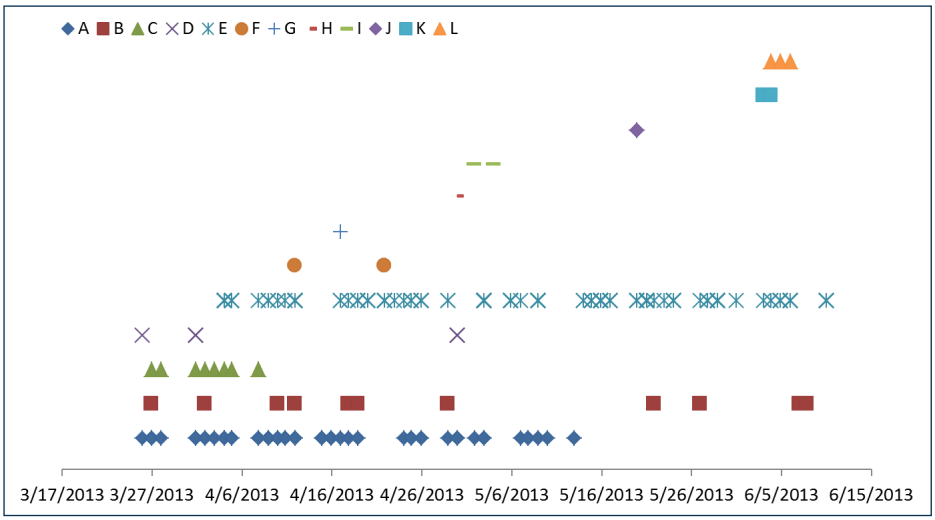


Figure 16: Juju entries by date

Only 56 entries contain actual stolen records (the accounts' transactional information), others just contain the machine's information along with a URL, as depicted in Figure 18. The records reflect five different versions of the malware agent software: 118, 126, 127, 128 and 129. All but one machine had consistently the same version of malware in all records. Only one machine (machine "I" in Figure 16) "evolved" from one version over time (from 128 to 129). This might indicate that the agent can have its software upgraded after initial infection. However, this machine had very few entries coming from only two different dates. Another explanation can be simply that different versions of the malware created multiple infections of the same machine. This hypothesis seems consistent with data displayed in Figure 17 that shows the version number entries over time.

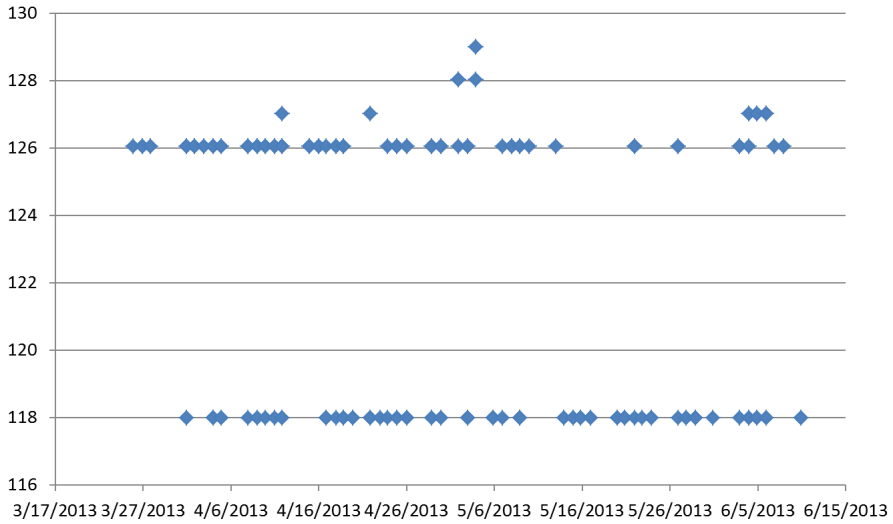


Figure 17: Juju malware version entries by date

All records taken from the same machine contain the same user name indicating that these are personal, non-shared workstations. The entries data is from May 26th to June 10th. Some machines provide stolen data throughout the entire period, suggesting that they have been compromised for months without their owners detecting the infection.

The "dados" column in each record contains the really interesting information. Some records contain mostly identification information about the machine and its user, while others contain bank account information, such as transaction history. Interestingly enough, none of the records seem to contain actual credentials for the banking application. We have noticed that all records taken from the same machine at the same timeframe carry the same value in a field called "CTRL" or "CONTROLE" which seems to be the session identifier used by the banking application itself. When actual data exists it is in a stripped down HTML-like format, suggesting that the malware took a copy of the page being displayed and sent it over to the drop server. All the accounts where information appears in the "juju" table are business accounts belonging to small organizations in Brazil. We were not able to determine the mechanism that allows the malware to select only these accounts, as we believe that private individuals have been infected as well (see Figure 13 above). Figure 18 shows a redacted version of one of the data records in the table.

```
188,1******_B************F49_06**********83,Macs:
IntelR PRO100 VE Network Connection - Miniporta do agendador de pacotes -
00********0
Sistema Operacional: Windows XP Professional Build 2600, Service Pack 2
Versao IE: 8
Nome Computador: BARBARA
Nome Usuario: SPM
Serial Volume HD: 1******
Serial Fixo HD: 06**********83
Serial CPU: B************F49
Versao Programa: 200 1 126
Url: https:*********empresa.com**********.aspCONTROLE438650629645078684&CTRL4386
50629645078684&DATAUACES
```

Figure 18: Juju entry

Transactional entries contain information about actual banking transactions:

```
***** HOTEIS E TURISMO LTDA
R*****h H***a
AGNCIA: **** CONTA: ***** - *
Saldos e Extratos Conta-Corrente , Conta-Corrente , Consulta
```

Addressed company with its account number

```
AGNCIA: **** CONTA: ***** - *
ULTIMOS 5 DIAS TEIS - CONTA CORRENTE
DATA HISTRICO DOC TO CREDITO DEBITO SALDO
120313 SALDO ANTERIOR  174,94
140313 DEPOSITO EM DINHEIRO
O PROPRIO FAVORECIDO 862108 230,10
.....
```

List of transactions in the last five days: deposits, withdrawals, checks, and current balance.

```
Quarta, 27 de maro de 2013, 08:**:**
E-mail      Sobre o Net Empresa      FAQ
Controle: 06****************94 Tela : 03 ** ** ** 02
```

Same control number as in previous entry

Figure 19: Juju transactional entry

We were not able to get a good understanding of the contents of the "Titulo" table, as it only contains five structured records. One additional record in the table holds what looks like a configuration script for the malware itself, indicating when to trigger capture and what contents to capture.

3. Reflections on Malware and Database Access

The analysis brought in the previous section discusses how malware is using a database server hosted in the cloud for C&C and drop server functionality. It shows that the technology for accessing relational databases is already embedded in modern malware. The TrendMICRO's analysis of database accessing malware is a clear example. TrendMICRO's analysis looks at a malware sample that contains the code which enables database connectivity.

Moreover, we have seen that attackers are already packaging database drivers into commercial malware modules. What does that mean for our internal database servers?

In a M-Trends 2012 report by Mandiant we are made aware of how intruders are using publicly available Remote Access Tools (RATs) to poke into enterprise databases. These programs are downloaded into compromised machine after the initial infection and are operated manually by the intruder.

Other incidents over the past three years show how attackers breach the integrity of enterprise databases using autonomous malicious code. The first publicly disclosed incident is that of the Stuxnet worm which tampered the configuration of SCADA devices by manipulating the MS SQL database server of the management application. The malware exploited databases for which default access credentials were still in place. A Symantec analysis shows a query issued by Stuxnet used to upload a Stuxnet DLL into the database, which later can be saved locally on disk:

```
CREATE TABLE sysbinlog ( abin image ) INSERT INTO sysbinlog VALUES(0x…)
```

Figure 20

Not long after this, we were introduced to the Narilam malware that targeted the database of Iranian financial software. Narilam updates MS SQL databases accessible by OLEDB in order to tamper stored data.

We reverse engineered one of the modules of the Kulouz malware family (a module that we have seen in the wild throughout 2012 and 2013) and discovered that it was linked with the SQLite library. It performed queries against browser data repositories in order to extract information, such as stored credentials (example against Mozilla FireFox is show in Figure 21). Common to all these is the fact that the malware agent targeted a database installed locally on the infected machine.

```
SELECT hostname, encryptedUsername, encryptedPassword, usernameField FROM moz_
logins
```

Figure 21

Given all the evidence, it seems that criminal hackers are only a small step away from using off-the-shelf malware for generic database access inside the enterprise. Once their motivation and business model becomes clear, whatever they lack in terms of technology they are certain to achieve. At that point, internal data stores of many more organizations are going to be part of the attack surface. These include organizations of all sizes and verticals – not only large defense contractors. Based on our observations and analysis, we expect the first generation of such tools to use standard SQL access to servers relying on default or stolen credentials for initial access. We expect this first generation of tools to target standard database structures of well-known applications (e.g. SAP, PeopleSoft).

4. Some Thoughts about Database Vulnerabilities

While analyzing the malware and its use of external database hosted in the cloud, our research went in different directions. In particular we came to ponder the risk posed to enterprises by database vulnerabilities.

In general, although database vulnerabilities draw a lot of media attention (e.g. the usual commotion around Oracle's quarterly CPU release) the risk they actually represent, for organizations who manage their internal data stores, is small. None of the data breaches of the past decade involving internal databases was attributed to a specific vulnerability in the database software (the last one we can recall was the Slammer worm break-out in January 2003). This is related to the fact that internal breaches are usually carried out by non-technical people using (or rather abusing) privileges that they rightfully own.

It all changes when we consider cloud databases and especially those in shared hosting. Cloud databases are directly exposed to the web and are accessible, from a network perspective, from anywhere in the world. Thus, they are affected by vulnerabilities that might be too complex to use by internal agents, but are the bread and butter of many criminal hackers. This is probably the reason why vulnerability CVE-2013-1899 in the open source database PostgreSQL invoked a commotion last year in the security world, and, in particular, in some of the biggest services that provide PostgreSQL as a service. The incident went as far as giving cloud platform provider Heroku early access to the patch, which by itself spurred a rather heated discussion (as seen here, and with PostgreSql trying to answer here). By successfully exploiting this vulnerability, an attacker can cause damage to shared service ranging from denial of service, to full overtaking of the host computer. All that is possible without a need for legitimate credentials to target system. Due to a bug in the server parsing, a connection request containing a database name that begins with "-" may be crafted to damage or destroy files within a server's data directory. The argument "-r" in the connection request specifies a file where all server output is to be sent. By using this option, an attacker can, in the first scene, easily point the gun of the attack by corrupting the server configuration file 'pg_hab.conf'. In the next scene, when the server will be restarted, the gun will shoot and the desired denial of service will be achieved.

Exploit (for DoS attack):

```
psql --host 10.1.1.1 --dbname="-rpg_hab.conf" -user="aaaaaaa"
```

A little more complex exploit can inject arbitrary code to the postgresql "user .profile" file. Once successful, the injected code will be invoked when 'su – postgresql' command is executed. The full and detailed POC (een here) was made public only four days after PostgreSQL published their security update, which is probably not enough time for patching the servers.

Vulnerability CVE-2011-5049 in the MySQL server allows a client to send to the server during the authentication message a charset value which exceeds the valid charset array. This then, causes a crash in server. Again, fully described POC can be found on the web (here). This is another example of vulnerability which doesn't require an attacker to have valid credentials and therefore, anyone can bring down any cloud service based on vulnerable MySQL.

MySQL vulnerability CVE-2012-5611, is stack-based buffer overflow vulnerability which allows remote authenticated users to execute arbitrary code via a long argument to the GRANT command. The denial of service exploit is as simple as this statement:

```
GRANT select ON
MYSQsssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssLqqqq
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.* TO 'user11'@'%'
```

Although this last example does require having valid credentials, it's not really a barrier. The attacker just needs to open an account. In many cases services allow opening free accounts, lowering the bar even lower.

Reviewing the above example and numerous others, we came to realize that the exploits are extremely easy to execute. Accordingly, we must change our perception of the risk posed by such vulnerabilities on shared hosting environments. In other words, given the analysis in Malware Analysis section above, how do you feel about sharing your database with a criminal? It is not enough that the database server is exposed to protocol layer attacks from cybercrooks. These miscreants can have their own databases hosted on the same server, thus granting them further access and allowing them to exploit privilege elevation vulnerabilities. Perhaps we are misinterpreting the severity score given to vulnerabilities (which used as a base for further risk analysis). For example, according to the industry accepted CVSS 2.0 standard any vulnerability that requires authentication is immediately rated not more than the base score of 9.0. For example, MySQL vulnerability on Windows, CVE-2012-0552, has the following CVSS metrics:

CVE#	Component	Protocol	Sub-component	Remote Exploit without Auth.?	CVSS VERSION 2.0 RISK (see Risk Matrix Definitions)						
					Base Score	Access Vector	Access Complexity	Authen-tication	Confiden-tiality	Integrity	Avail-ability
CVE-2012-5611	MySQL Server	MySQL Protocol	Server Privileges	No	9.0	Network	Low	Single	Complete	Complete	Complete

Figure 22

The only metric that lowers the CVSS score from 10.0 to 9.0 is the need for single authentication. However, for databases hosted in a shared environment, obtaining credentials and going through the authentication process is trivial. Thus a vulnerability whose severity CVSS score is 9.0 should actually be considered to have score 10.0 in a shared hosted environment. Complexity and impact should also be treated differently in shared cloud services, due to the exposure to the much wider attacker audience and multi-tenant nature of the impacted service.

Reviewing the way PostgreSQL vulnerabilities are published makes our point even clearer. There are four levels of scores as following (and shown here):

Class	Description
A	A vulnerability that is exploitable for privilege escalation *without* requiring a prior login.
B	A vulnerability that is exploitable for denial-of-service *without* requiring a prior login.
C	A vulnerability that is exploitable for privilege escalation, but requiring a valid prior login.
D	A vulnerability that is exploitable for denial-of-service, but requiring a valid prior login.

Figure 23: PostgreSQL scoring standard

The only metric that lowers the score from A to C, or from B to D is the need for authentication. Thus a vulnerability with a severity score of C and D, should actually, in a shared hosted environment, be considered to have score of A and B. Figure 25 demonstrates the scoring class percentage of PostgreSQL vulnerabilities since 2009 according to PostgreSQL scoring standard. The left part presents the percentage of each scoring class. As we see it, the right part of Figure 24 shows present risk scoring in a shared environment. Our perspective is that on shared environments, when a login to the attacker database on the server can be easily gained, we should set class types C, D to A, B.

While on the original scoring only 17% of the vulnerabilities belong to class 'A' and can be exploited for privilege escalation without requiring login, in the right chart about 79% of the vulnerabilities belong to class 'A'.

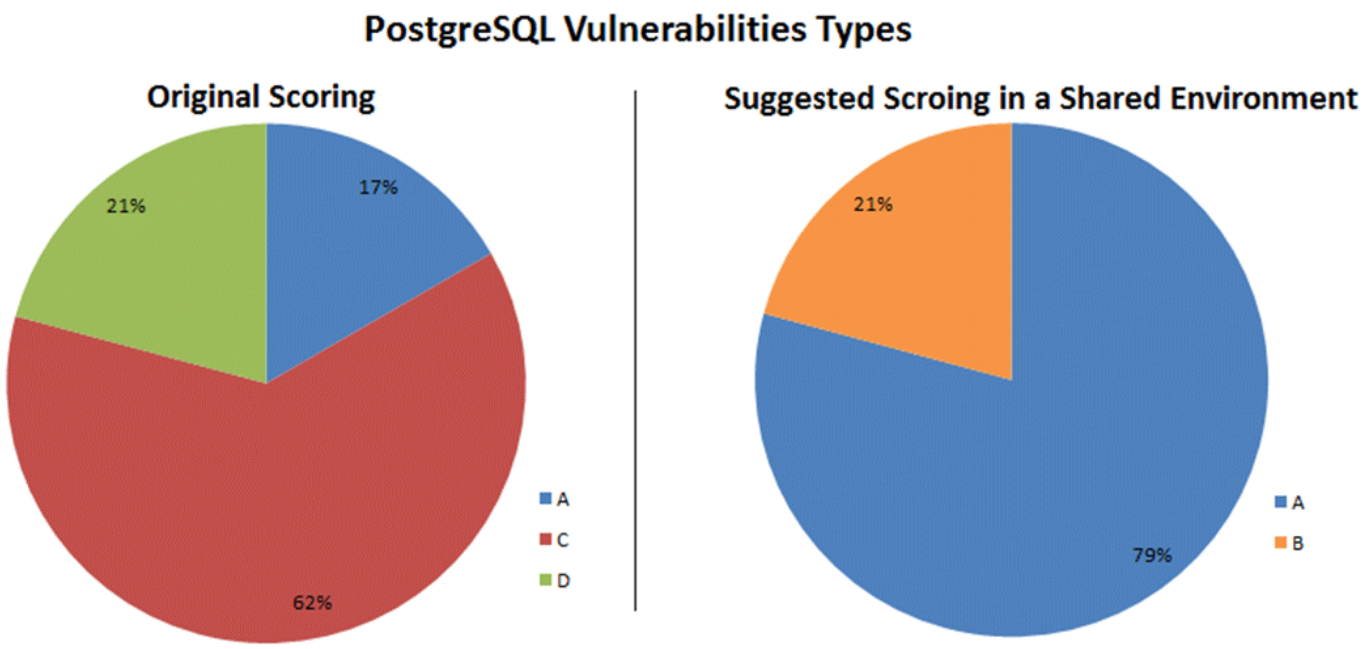


Figure 24: Distribution of vulnerability types in PostgreSQL since 2009. On the left, according to PostgreSQL scoring, and on the right, according to the scoring appropriate for shared environment.

5. Summary and Conclusion

We have started by analyzing a malware that uses a shared hosting database service for its C&C and drop server. We have seen how criminals are abusing this type of infrastructure for their malicious endeavors. This led us in two different directions with respect to analyzing the risk to organizations.

First and foremost, the technology used by the malware, together with additional evidence taken from other malware samples, suggest that very soon, we will see autonomous malware targeting internal databases within organizations. Once again, we claim (based on our experience with this malware and previous research) that infection is inevitable, and compromise of a portion of workstations within a network should be considered an inherent condition. Thus, organizations must improve controls around data stores as a mitigation strategy, focusing on technologies like database audit and DAM.

Second it seems that organizations that host their data in a cloud service are exposing it to higher risks than originally perceived. Due to the exposure of the database to technically savvy attackers and to the ease of obtaining a legitimate foothold on such a server, risk factors are increased. This can quickly be turned into a privilege escalation attack. This change in how we perceive the risk should be taken into consideration by organizations when they decide which data they want to store externally It should also serve as a wake-up call for service providers to look for deploying virtual patching solutions.

Hacker Intelligence Initiative Overview

The Imperva Hacker Intelligence Initiative goes inside the cyber-underground and provides analysis of the trending hacking techniques and interesting attack campaigns from the past month. A part of Imperva's Application Defense Center research arm, the Hacker Intelligence Initiative (HII), is focused on tracking the latest trends in attacks, Web application security and cyber-crime business models with the goal of improving security controls and risk management processes.

Imperva
3400 Bridge Parkway, Suite 200
Redwood City, CA 94065

Tel: +1-650-345-9000
Fax: +1-650-345-9004
www.imperva.com