Misusing Open Services on the Internet

Jelte Fennema **Ben de Graaff**

University of Amsterdam

Supervisor: Rick van Galen (KPMG)

February 3, 2016

Introduction

Open service: no authentication or default credentials

Relevant: more than **35,000** open MongoDB instances
Exposing **685 TB** (last December [1])

More than just data leaks – example: botnet command and control

A problem for devops *and* software developers

"Memcached does not spend much, if any, effort in ensuring its defensibility from random Internet connections. So you must not expose Memcached directly to the Internet."

– Memcached documentation

"Everybody has privileges to do anything. Neat."

— CouchDB security documentation

Research goals

- What are settings that lead to exploitable services?
- What are the operations required when exploiting an open service as a command & control server?
- What are best practices for default configurations and authentication?

Approach

For various software packages...

- Examine configuration (weaknesses?)
- Tool to scan level of access
- Proof of concept: botnet command & control

Approach

For various software packages...

- Examine configuration (weaknesses?)
- Tool to scan level of access
- Proof of concept: botnet command & control

Scanning the Internet

- Shodan
- ZMap and our own scan tool

Software classes

- **Relational databases:** MySQL, MariaDB, PostgreSQL

Software classes

- **Relational databases:** MySQL, MariaDB, PostgreSQL
- **NoSQL databases:** MongoDB, CouchDB

Software classes

- **Relational databases:** MySQL, MariaDB, PostgreSQL
- **NoSQL databases:** MongoDB, CouchDB
- **Key-value store:** Redis, Memcached

Software classes

- **Relational databases:** MySQL, MariaDB, PostgreSQL
- **NoSQL databases:** MongoDB, CouchDB
- **Key-value store:** Redis, Memcached
- **Message queue:** RabbitMQ

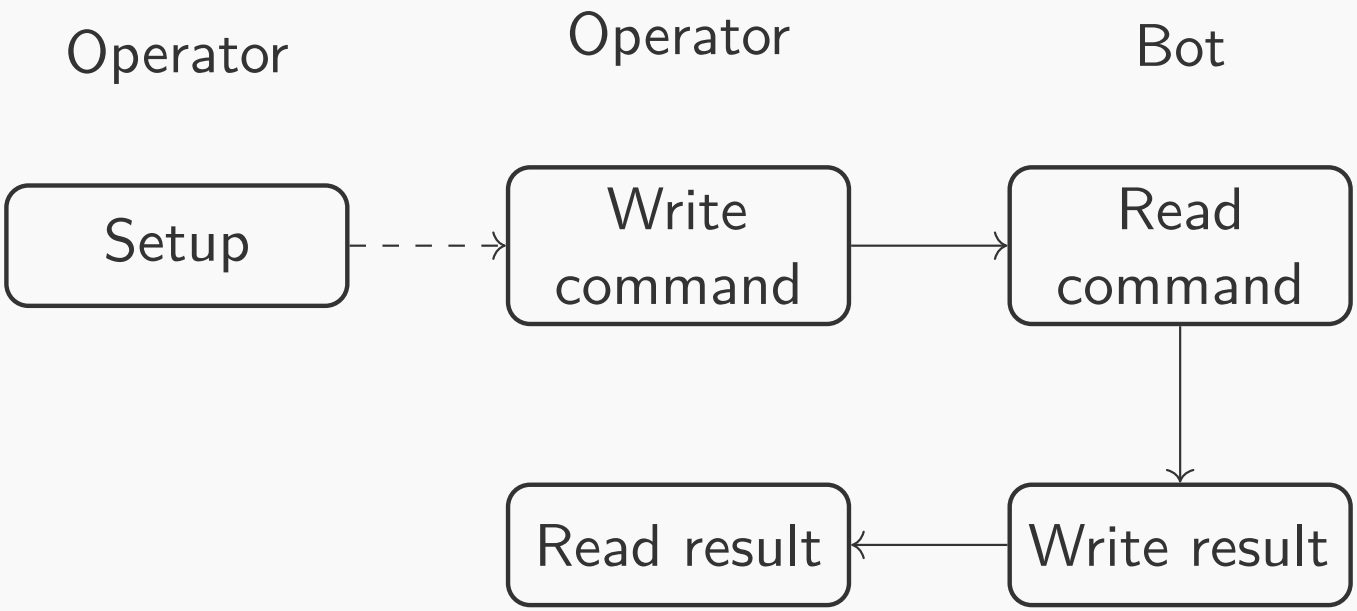
Software classes

- **Relational databases:** MySQL, MariaDB, PostgreSQL
- **NoSQL databases:** MongoDB, CouchDB
- **Key-value store:** Redis, Memcached
- **Message queue:** RabbitMQ
- **Printing protocols:** CUPS (and IPP printers)

Proof of concept

Simple botnet simulation (communication channel):

- Botnet operator sends signed *commands* to one bot or all bots
- Bots execute commands, write back encrypted *results*



Impact on the Internet

What is the impact on the Internet?

Is configuration security a factor?

Configuration security

	PostgreSQL	MySQL	CUPS	RabbitMQ	CouchDB	MongoDB	Redis	Memcached
Localhost (in config)	**Y**	**Y**	**Y**	N	**Y**	**Y**	N	N
Not public (default)	**Y**	N	**Y**	N	**Y**	N	N	N

Table 1: Comparison of security settings for the software packages

Configuration security

	PostgreSQL	MySQL	CUPS	RabbitMQ	CouchDB	MongoDB	Redis	Memcached
Localhost (in config)	**Y**	**Y**	**Y**	N	**Y**	**Y**	N	N
Not public (default)	**Y**	N	**Y**	N	**Y**	N	N	N
Authentication by default	**Y**	**Y**	**Y**	**Y**	N	N	N	N
No public creds or anon	**Y**	**Y**	**Y**	**Y**	N	N	N	N

Table 1: Comparison of security settings for the software packages

Configuration security

	PostgreSQL	MySQL	CUPS	RabbitMQ	CouchDB	MongoDB	Redis	Memcached
Localhost (in config)	**Y**	**Y**	**Y**	N	**Y**	**Y**	N	N
Not public (default)	**Y**	N	**Y**	N	**Y**	N	N	N
Authentication by default	**Y**	**Y**	**Y**	**Y**	N	N	N	N
No public creds or anon	**Y**	**Y**	**Y**	**Y**	N	N	N	N
Host-based access control	**Y**	**Y**	**Y**	**Y**	N	N	N	N

Table 1: Comparison of security settings for the software packages

Configuration security

	PostgreSQL	MySQL	CUPS	RabbitMQ	CouchDB	MongoDB	Redis	Memcached
Localhost (in config)	**Y**	**Y**	**Y**	N	**Y**	**Y**	N	N
Not public (default)	**Y**	N	**Y**	N	**Y**	N	N	N
Authentication by default	**Y**	**Y**	**Y**	**Y**	N	N	N	N
No public creds or anon	**Y**	**Y**	**Y**	**Y**	N	N	N	N
Host-based access control	**Y**	**Y**	**Y**	**Y**	N	N	N	N
Authentication always on	N	**Y**	N	**Y**	N	N	N	N

Table 1: Comparison of security settings for the software packages

Configuration security

	PostgreSQL	MySQL	CUPS	RabbitMQ	CouchDB	MongoDB	Redis	Memcached
Localhost (in config)	**Y**	**Y**	**Y**	N	**Y**	**Y**	N	N
Not public (default)	**Y**	N	**Y**	N	**Y**	N	N	N
Authentication by default	**Y**	**Y**	**Y**	**Y**	N	N	N	N
No public creds or anon	**Y**	**Y**	**Y**	**Y**	N	N	N	N
Host-based access control	**Y**	**Y**	**Y**	**Y**	N	N	N	N
Authentication always on	N	**Y**	N	**Y**	N	N	N	N
Minimal steps to make open	3	2	3	1	1	1	0	0

Table 1: Comparison of security settings for the software packages

Configuration security

	PostgreSQL	MySQL	CUPS	RabbitMQ	CouchDB	MongoDB	Redis	Memcached
Localhost (in config)	**Y**	**Y**	**Y**	N	**Y**	**Y**	N	N
Not public (default)	**Y**	N	**Y**	N	**Y**	N	N	N
Authentication by default	**Y**	**Y**	**Y**	**Y**	N	N	N	N
No public creds or anon	**Y**	**Y**	**Y**	**Y**	N	N	N	N
Host-based access control	**Y**	**Y**	**Y**	**Y**	N	N	N	N
Authentication always on	N	**Y**	N	**Y**	N	N	N	N
Minimal steps to make open	3	2	3	1	1	1	0	0
Steps to make public/secure	3	2	3	1	3	3	1	2

Table 1: Comparison of security settings for the software packages

Percentage of open services exposed to the Internet

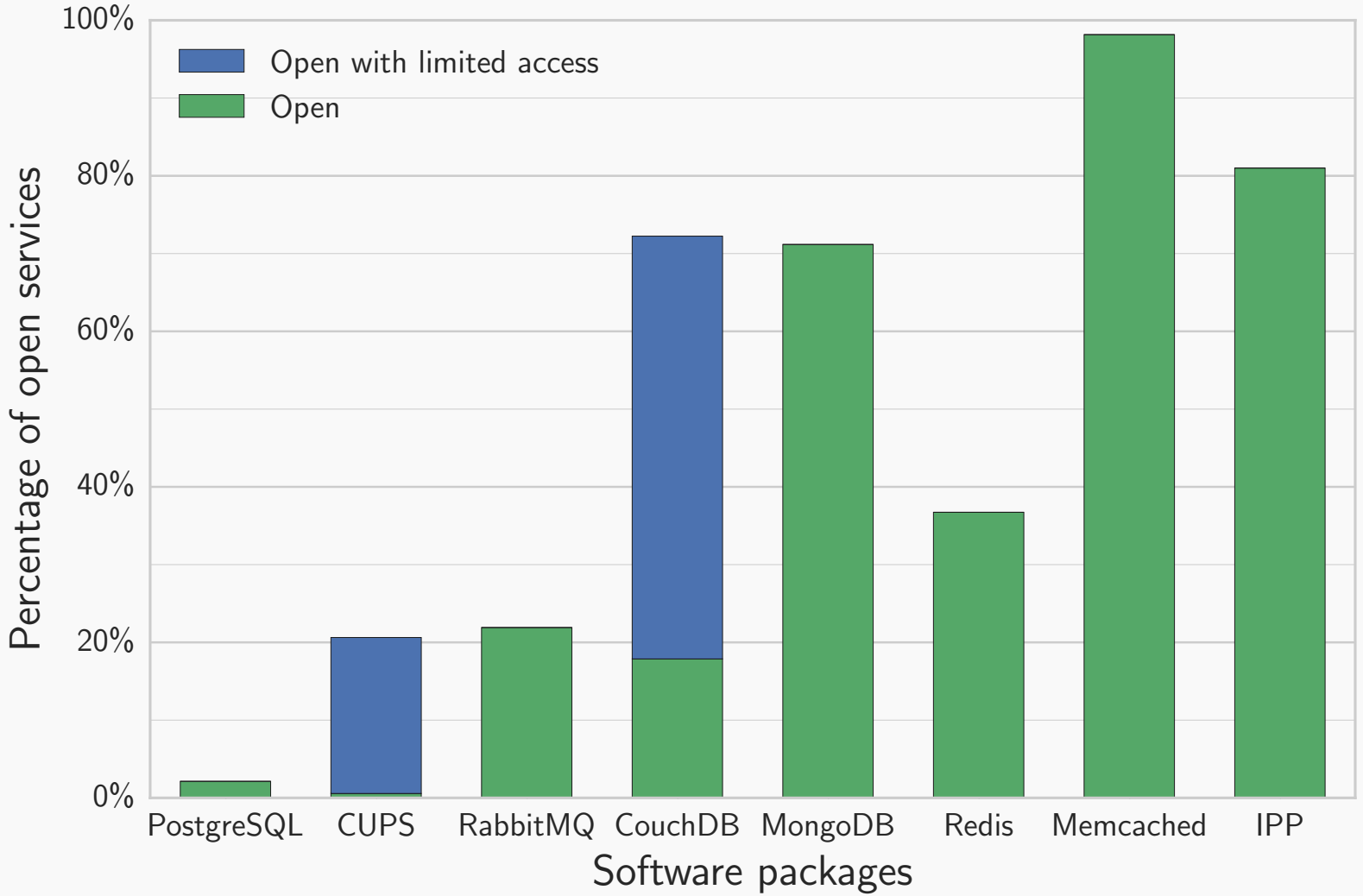


Figure 1: Percentages of open services.

Combined results

	PostgreSQL	CUPS	RabbitMQ	CouchDB	MongoDB	Redis	Memcached	IPP
Localhost (in config)	**Y**	**Y**	N	**Y**	**Y**	N	N	–
Not public (default)	**Y**	**Y**	N	**Y**	N	N	N	–
Authentication by default	**Y**	**Y**	**Y**	N	N	N	N	–
No public creds or anon	**Y**	**Y**	**Y**	N	N	N	N	–
Host-based access control	**Y**	**Y**	**Y**	N	N	N	N	–
Authentication always on	N	N	**Y**	N	N	N	N	–
Minimal steps to make open	3	3	1	1	1	0	0	–
Steps to make public/secure	3	3	1	3	3	1	2	–
Percentage open	2%	21%	22%	72%	71%	37%	98%	81%
With full access	2%	1%	22%	18%	71%	37%	98%	–

Table 2: Combined comparison of software packages

First attempt: Shodan

Shodan:

- Scans the Internet
- Sends command related to that service
- Stores result in plain text

Full text search interface to find open services

Example: `port:5984 couchdb !unauthorized`

Shodan results

Package	Open	Closed
Memcached	100,044	–
MongoDB	47,351	–
Redis	13,455	23,174
RabbitMQ	6,487	23,121
PostgreSQL	6,391	293,481

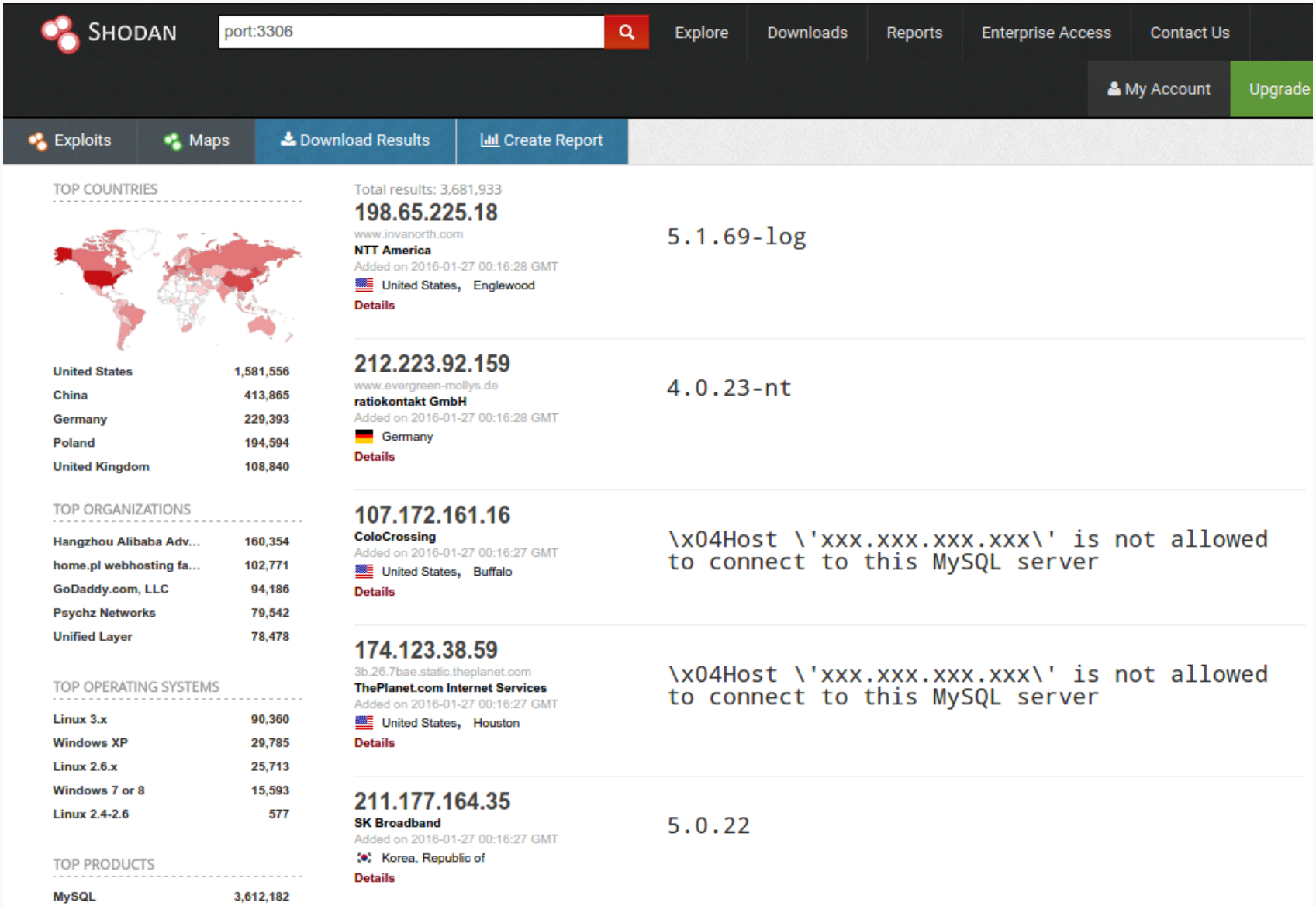
Table 3: Worldwide statistics based on Shodan

Shodan inconclusive results

Package	Unknown	Closed
MySQL/MariaDB	1,767,930	2,231,132
Non-CUPS IPP	23,948	1,664
CouchDB	2,783	513
CUPS	5,591	29,387

Table 4: Inconclusive statistics based on Shodan

Shodan unknowns



Second attempt: Our own scan

Shodan incomplete for some services

Setup

- Permission to scan all Dutch IPs
- Not allowed to log in to any service (required for MySQL, PostgreSQL)

Second attempt: Our own scan

Shodan incomplete for some services

Setup

- Permission to scan all Dutch IPs
- Not allowed to log in to any service (required for MySQL, PostgreSQL)

How

- ZMap to port scan (5.5 minutes for 4.6 million IPs)
- ~20,000 hits per port
- Scanner modified for concurrency (7 minutes at 500 concurrent requests)

Our scan results

Service	Open		Closed	Admin
Memcached	**98%**	3,725	70	–
IPP	**81%**	260	61	–
CouchDB	**72%**	190	73	47
MongoDB	**71%**	1,859	753	–
CUPS	**21%**	474	1,824	13

Table 5: Dutch statistics based on our scan

Type of IP range per service

CUPS/IPP

- Consumer networks
- Some universities
- Businesses

Memcached, MongoDB, CouchDB

- Mostly hosting services

Other quirks we found

CUPS:
Print jobs can contain arbitrary attributes

. . . turning a printing job into a key-value store

Other quirks we found

CouchDB:
Default security policy empty (writable by anyone)

. . . including database containing user credentials

Other quirks we found

Memcached:
Authentication requires *different*, binary protocol

. . . not even supported by all clients

Conclusion

- Open services still an issue
- Exploitation is *incredibly easy*:
  any service you can write data to and read it back later
- Bad (default) configuration can lead to exploitable services

Best practices

- Prefer **localhost** access, **require** authentication for remote access

Best practices

- Prefer **localhost** access, **require** authentication for remote access
- Secure **defaults**!

Best practices

- Prefer **localhost** access, **require** authentication for remote access
- Secure **defaults**!
- Clear **documentation** and warnings (also in configuration)

Best practices

- Prefer **localhost** access, **require** authentication for remote access
- Secure **defaults**!
- Clear **documentation** and warnings (also in configuration)
- **Simplify** configuration of authentication (good configs, tools)

References

- John Matherly. *It's Still the Data, Stupid!* 15th Dec. 2015. URL: https://blog.shodan.io/its-still-the-data-stupid/ (visited on 27/01/2016).
- *Shodan: the world's first search engine for Internet-connected devices.* 2009. URL: https://www.shodan.io/ (visited on 04/01/2016).
- Zakir Durumeric, Eric Wustrow and J. Alex Halderman. "ZMap: Fast Internet-Wide Scanning and its Security Applications". In: *Proceedings of the 22nd USENIX Security Symposium*. Aug. 2013.