digital whisper
Technologic Papers

סוגיות אבטחה ב-MongoDB

מאת 5Fingers

הקדמה

בסיס נתונים הוא אמצעי המשמש לאחסון נתונים כלשהם במחשבים בצורה שבה יהיה קל, נוח ופשוט לנהל ולשמור אותם בצורה מאורגנת ואחידה. בשנת 1970 אדגר פ.קוד (Edgar F. Codd) מתמטיקאי בוגר אוקספורד שעבד במעבדת המחקר של IBM בסן חוזה, פרסם מאמר שמראה איך מידע המאוחסן במסדי נתונים יכול להשמר מבלי שבסיס הנתונים ידע את סוג המידע או הדרך שבה יאוחסן.

אדגר אמר פעם:

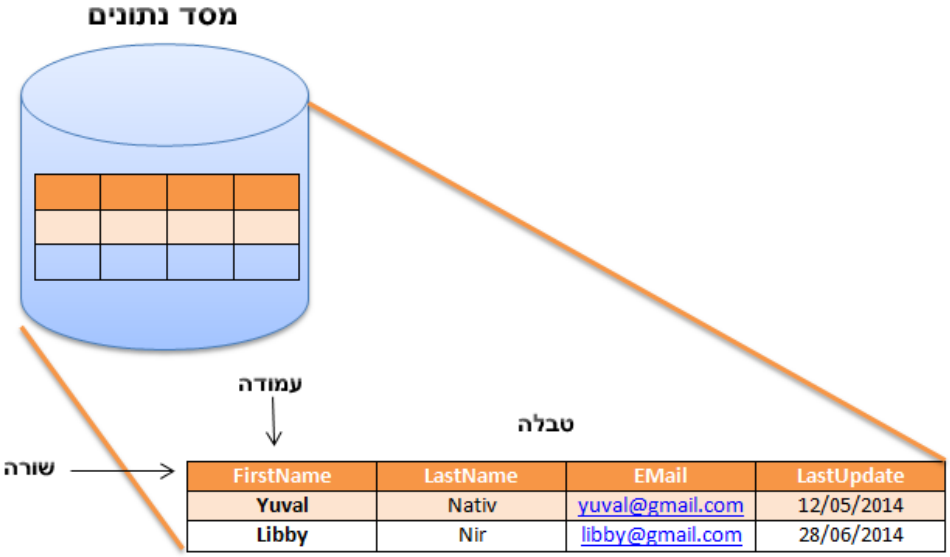
"המודל של בסיס נתונים רלציוני היה מההתחלה שנוי במחלוקת, אנשים חשבו שהמודל פשטני מדיי ושהוא אף-פעם לא ייתן ביצועים טובים".

יש לאחרונה כמות עצומה של באז סביב הנושא של BIG DATA. במאמר זה אנסה לעשות סדר בדברים ולהסביר את ההבדלים בין מסדי נתונים מסורתיים (רלציונים) לבין מסדי נתונים של BIG DATA, פלטפורמות עיבוד הנתונים ואציג כמה מהאתגרים שעולם ה-BIG DATA מביא עימו.

מסדי נתונים רלציונים - מה זה?

מסד נתונים יחסי/רלציוני (Relational Database) הוא אוסף של פריטי נתונים המאורגנים כקבוצה של "טבלאות" (לוחות עם שורות ועמודות) ממש כמו באקסל ובין הטבלאות יכולים להתקיים קשרים המתבססים על מכנה משותף ובכך מתאפשר לנו לאחזר ולשלב נתונים מכמה טבלאות בו זמנית.

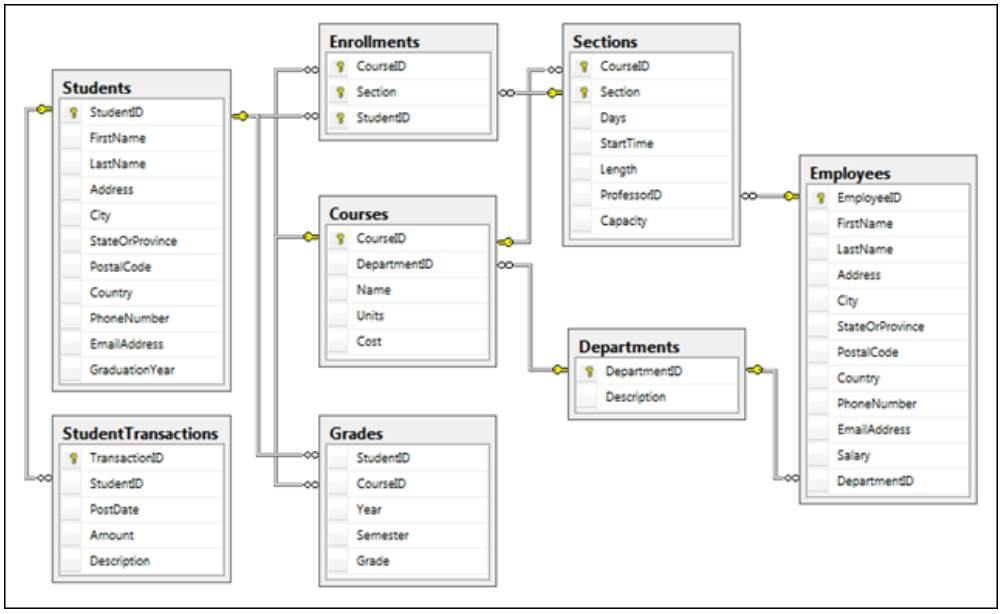
מסדי הנתונים הרלציוניים יוצרים סכמה מובנית עבור ארגון המידע. טבלאות, עמודות, קישורים ואחזור הנתונים. על כל זה ניתן לשלוט באמצעות בשפת השאילתות SQL הידועה לכולנו. מסד הנתונים הרלציוני הוא פתרון מעולה לשמירה ואחזור של נתונים עבור מרבית המערכות הקיימות היום.



[תמונה מס' 1: הדגמה של בסיס נתונים רלציוני]

על מנת להבין את היעילות שמסדי הנתונים הרלציונים מביאים עימם, להלן דוגמא קטנה:

מוסד אקדמי אוניברסיטה / מכללה נאלץ להתמודד מדי שנה עם רשימות גדולות של סטודנטים, שיעורים, מבחנים, ציונים, מרצים וכו'. במידה ותרצה האוניברסיטה לנהל את כל הנתונים האלה בדפי נייר, הרי שמדובר בכמות עצומה של עבודה שבסופו של דבר מתוגרמת לזמן, וזמן = כסף. אז כדי ליעל את העבודה נשתמש במסד נתונים רלציוני כדי לאחסן את כלל הנתונים האלה.



[תמונה מס' 2: דיאגרמה של מסד נתונים של האוניברסיטה]

כפי שניתן לראות פרטי הסטודנטים, הקורסים, הציונים וכו' מנוהלים כעת בטבלאות בדיוק כפי שהיו יכולים להיות מנוהלים באקסל, אך עכשיו בעזרת מסד הנתונים, לא רק שכל הרשימות האלו מאורגנות ומאוחסנות במקום אחד ולא רק שניתן גם לאחזר נתונים בצורה מהירה בניגוד לדפי נייר, אלא שעכשיו ניתן לבצע קורלציה בין כלל הנתונים ולאחזר נתונים העונים למס' קריטריונים ממס' רשימות/טבלאות שונות.

מסדי נתונים רלציונים, הם מה שרובנו (אם לא כולנו) מכירים מחיי היומיום בעבודה (Oracle, MS-SQL וכו') כאשר הם מאפשרים לנו:

- נגישות (Accessibility) מהירה לכל הנתונים (למי שמותר)
- בטחון הגישה לנתונים (Data Security)
- עדכון במקום אחד (מניעת סרבול בעדכון)
- חסכון במקום, מניעת כפילויות (Duplications)
- שלמות הנתונים (Data Integrity)
- אפשרות לעבודה במקביל (Parallel)
- אפשרות לעבודה באופן מבוזר (Distributed)
- קלות בתחזוקה (שדרוג הגדרות/מחשבים/דיסקים/תוכנה)

אך למסדי הנתונים האלה יש גם לא מעט חסרונות:

- סיבוכיות (Complexity):
  - בעיה שיצרני המערכות - Oracle, Microsoft, IBM - מתמודדים כל הזמן.
- ביצועים (Performance):
  - מחיר יקר לעבודה עם התיאורים השונים: פיזי, לוגי.
  - יישומים רגישים לזמן (Time-Critical).
- עלות:
  - רכישה, יישום, הטמעה, תחזוקה שעולים המון אבל המון כסף.
- משאבי חומרה:
  - דורשים הרבה זמן חישוב.
  - לא תמיד יעילים באחזקת הנתונים על הדיסק.
- מורכבות שחזור.

אמנם למסדי נתונים רלציונים יתרונות גדולים, אך הם דורשים הרבה עבודה כדי לארגן את כל המידע בצורה מסודרת. בנוסף השיטה לתשאול מתוכם (שפת SQL) היא שפה מאוד לא ידידותית. בנוסף למסדי הנתונים האלה קשה להתמודד עם Scaling. מסדי נתונים של NoSQL עשו קצת התקדמות בנושא אך זה עדיין לא מושלם.

BIG DATA - מה זה

בין כל ההגדרות המוצעות ל-BIG DATA, התחברתי לזאת שאומרת שאם הנתונים גדולים מדיי ויש צורך להשתמש בהם במהירות ותצורת העבודה עם הכלים הקיימים כיום קשה מדיי - זה BIG DATA.

הנתונים גדולים מדיי - משמעות הדבר היא שיותר ויותר ארגונים מתמודדים עם כמויות עצומות של חומר פטה-בייט (petabyte) שזורומות בשניות, לשמור היסטוריה וכו'.

מהר יותר - משמעות הדבר היא לא רק שהנתונים גדולים, הם גם חייבים להיות מעובדים במהירות לדוג' כאשר נרצה לבצע זיהוי הונאה בנק' מכירה שונות או לקבוע איזו פרסומת להציג למשתמש בדף האינטרנט.

קשה מדיי - סוגייה זו מעט כללית ובעייתית לנתונים שאינם מתאימים בצורה מסודרת לכלי העיבוד הקיימים או שיש צורך בניתוח הנתונים שהכלים הקיימים לא יכולים לספק עבור כמות הנתונים ובזמן מהיר.

BIG DATA-ו NoSQL

הרבה אנשים נוטים לקשור את צמד המילים האלה ביחד מבלי להבין באמת את המשמעות של כל אחד מהם כאשר לא תמיד שימוש באחד מחייב את השני. בואו נעשה קצת סדר בדברים.

מסדי הנתונים הרלציונים מאפשרים לנו שאילתות SQL מורכבות, סיכומים סטטיסטים, התמודדות עם constraints, טריגרים, Stored Procedures ועוד. עכשיו בואו ניקח את כל היכולות האלה ונוותר עליהם לטובת היכולת לטפל ב-Scale עצום.

NoSQL או בשמה המלא Not Only SQL, היא טכנולוגית ניהול נתונים שנועדה לענות על הגדלת הנפח, המהירות, עיבוד, ניתוח ומגוון רחב של סוגי נתונים שחברות/ארגונים מתמודדים.

בהשוואה למסדי נתונים רלציונים, מסדי נתונים NoSQL ניתנים להרחבה יותר ולספק ביצועים מעולים. מסדי נתונים אלה מספקים מס' חידושים:

- Elastic scaling - ארגונים יכולים לבצע scale out בהתאם לצרכי אחסון הנתונים שלהם.
- אין מודל נתונים קבוע (structured and unstructured data) - גמישות זו מעניקה לארגונים לגשת לכמויות הרבה יותר גדולות של נתונים.
- היכולת להתמודד עם כשל חומרה - העובדה שכשלים בחומרה מתרחשים פותח מסד הנתונים NoSQL עם יתירות (redundancy) מראש.

מערכות מסדי נתונים של NoSQL אינן מגיעות עם איזשהו מבנה יחסי/מודל כלשהו לאיחסון הנתונים. הנתונים נשמרים בקבצים בניגוד לטבלאות שיש במסדי הנתונים הרלציונים.

יש מגוון רחב של סוגי בסיס נתונים של NoSQL כגון:

- Key-Value-DB
- Column-DB
- Document-DB
- Graph-DB
- Object-DB

Document Oriented DB

מסד נתונים מונחה מסמך שנועד לאחסון, אחזור, וניהול מסמך מכוון, או נתונים ללא מבנה (unstructured data). זהו הרעיון המרכזי של מסדי נתונים NoSQL. למרות שכל מסמך במסד הנתונים יכול להיות מוגדר בצורה שונה משאר המסמכים, באופן כללי, כל המסמכים משמשים לשמירת הנתונים באותו פורמט. קידוד המסמך יכול להיות ע"י XML, YAML, JSON ו-BSON, בצורה בינארית כמו PDF ומסמכי Microsoft Office - Word, Excel, וכך הלאה.

נשתמש לרגע בדוגמא של האוניברסיטה ממקודם:
במידה ומוסד אקדמי אוניברסיטה/מכללה רוצה לייצג רשומות היסטוריות רבות: לדוגמא סיכום של כלל הסטודנטים בפקולטה מסויימת. אם נשתמש בסיס נתונים רלציוני ומנורמל תהיה לנו טבלת קורסים (נאמר 20) המקושרת לטבלת מבחנים (נאמר מיליון בשנה, עשר שנים = 20 מיליון) ואחד מכללי האצבע בבסיסי נתונים הוא: שמעל **10** מיליון רשומות בטבלה בסיס הנתונים מתחיל להגיב לאט. כלומר - לא מעשי.

OK הבנו, אבל איך כל זה קשור לאבטחת מידע?

SQL Injection - זה כ"כ 90'

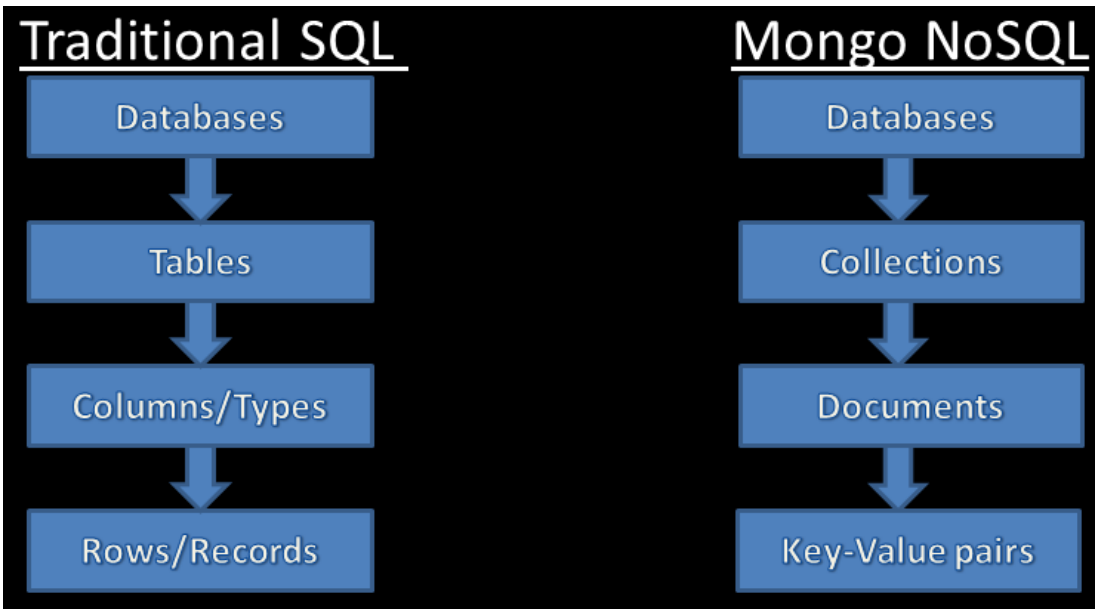
מכיוון שמסדי הנתונים עברו שינוי בתצורת העבודה שלהם, משתנה גם דרך הפריצה אליהם. הזרקת SQL - היא אחד ממנגנוני ההתקפה הרבים בשימוש ע"י האקרים כדי לגנוב נתונים מארגונים או מאנשים פרטיים. זו אולי אחת מטכניקות התקיפה הנפוצות ביותר בשכבת האפליקציה בשימוש כיום. זוה סוג של התקפה שמנצל קידוד לא תקין של יישומי תוכנה, המאפשר להאקר להזריק פקודות SQL (למשל טופס התחברות) כדי לאפשר לו לקבל גישה לנתונים שנשמרו בתוך מסד הנתונים שלך.

על טכניקה זו אין צורך להרחיב כיוון שהאניטרנט מוצף דוגמאות והסברים על נושא זה.

NoSQL = No Injection?

בדוגמאות שמיד הציג בסיס הנתונים הנבחר הינו: MongoDB. mongoDB הינו פרויקט קוד פתוח (open source) למסד נתונים מבוסס **Document Oriented**. המודל שלפי עובד mongoDB הינו שבכל מסד נתונים יש מס' אוספים ובכל אוסף יש מס' מסמכים כאשר כל מסמך מכיל צמדים של Key-Value. (מיד המושגים האלה יובהרו).

מודל נתוני המסמך שמקבל MongoDB הוא למעשה JSON בינארי כלומר BSON, דבר אשר מאפשר תחזוקה וניהול מאוד נוחים. ההבדל הקונספטואלי במבנה, בין MongoDB למשל, לבין מסדי נתונים מבוססי SQL, הוא כדלקמן:



[התמונה לקוחה מהמצגת "Making Mongo Cry: NoSQL for Penetration Testers" של Russell Butturini]

אז, ב-mongoDB יש אוסף (טבלה) והוא מחזיק מסמך אחד או יותר (רשומות). כל מסמך יכול להיות שדה אחד או יותר (עמודות).

אז אחרי שנכנסו למים, בואו נצלול טיפה.

NoSQL Injection במסדי נתונים מבוססי MongoDB:

בבסיסי הנתונים הרלציונים הידועים לכולנו כאשר אנחנו מעוניינים לתשאל את ה-DB אנחנו נעשה זאת ע"י שאילתת SQL בצורה הבאה:

```
SELECT * FROM users WHERE username = '$username' AND password = '$password'
```

אם השאילתא לא נכתבה כראוי, תוקף יכול להציב את הערך ' or '1'='1' -- בשדה ה-username ובכך לעקוף או לרמות את השאילתא המקורית:

```
SELECT * FROM users WHERE username = ' ' or '1'='1' -- AND password = ''
```

ב-MongoDB הסיפור קצת אחרת.

כאשר נרצה לכתוב את השאילתא שלעיל עבור mongo, נעשה זאת כך:

```
db.users.find({username: username, password: password});
```

הסבר: גש וחפש בתוך ה-collection של users האם קיימים ה-username וה-password האלה.

כפי שניתו לראות הדרך בה אנחנו מתשאלים את בסיס הנתונים השתנתנה מעט. אין יותר את שפת השאילתות SQL יש פונקציות. למעשה mongoDB מספקת לנו שכבת ניהול כזו שמקלה מאוד על איחזור הנתונים , אם ניקח את הדוגמא הקודמת נוכל להכניס תנאים בתוך הפונקציה find ממש כמו where clause בשפת SQL:

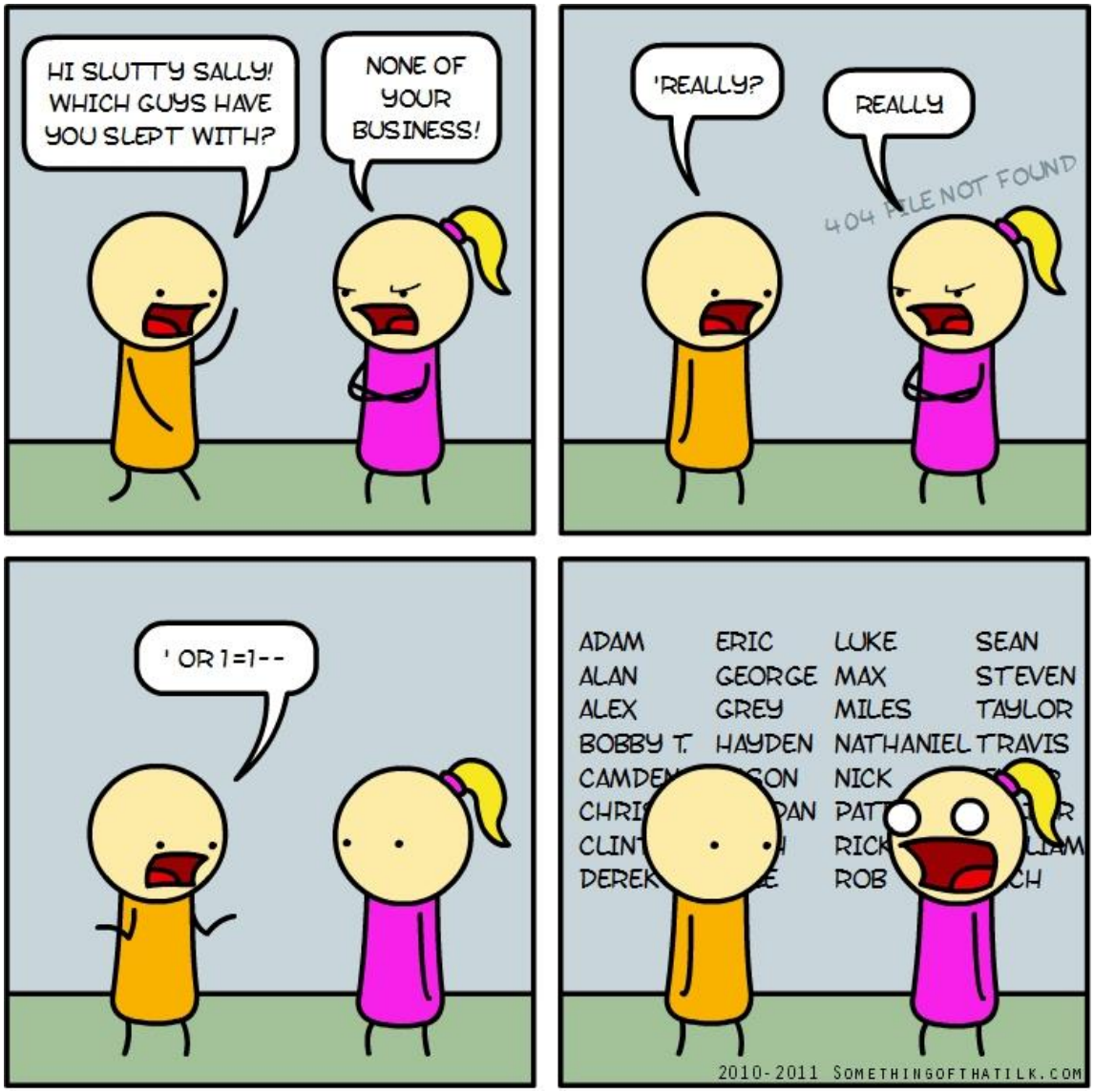
```
db.users.find( { username: { $in: [ 'root, 'admin' ] } } )
```

בדוג' הנ"ל אנחנו מבקשים לאחזר רשומות ש-username שווה ל-root **או** ל-admin. אז נכון זו לא שפת שאילתות, כך גם בה יש חולשות וגם לה אפשר לבצע הזרקה. הזרקה במקרה שלנו יכולה להיות ממומשת כך:

בפרמטר username נשים את הערך: "//'})root"=username

```
db.users.find({username: 'root'}) //,password: password});
```

בדוגמא הנ"ל לא מבצעים ואלידציה על תקינות הקלט להשדות username ו-password ובכך ביטלנו את הפרמטר password.



[במקור: http://www.somethingofthatilk.com/index.php?id=271]

דוגמאות לפגיעויות במסדי נתונים מבוססי MongoDB:

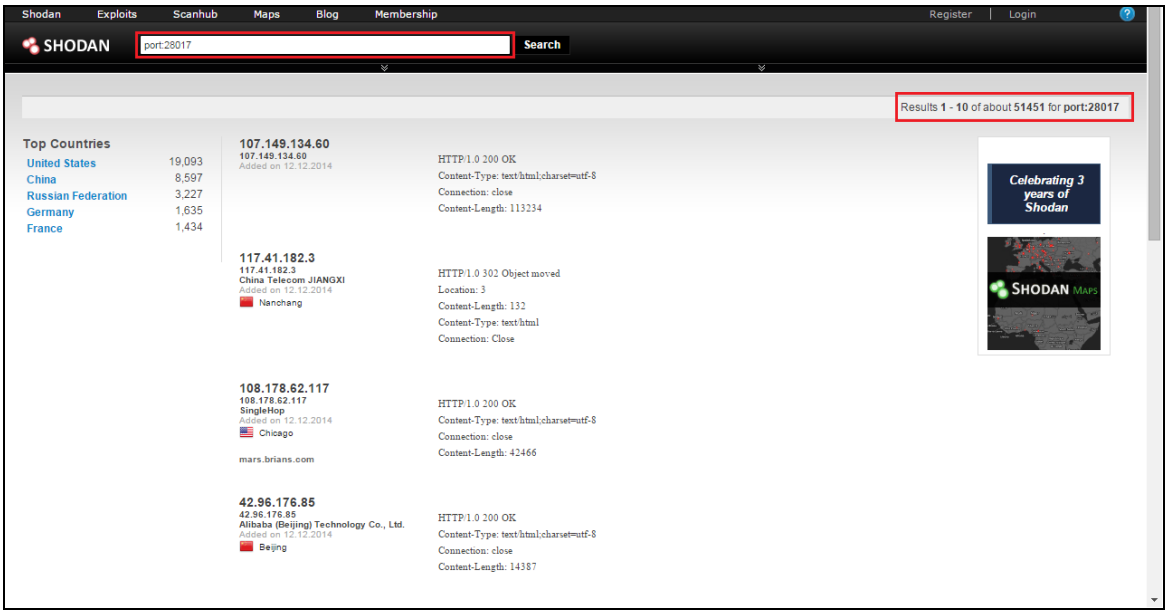
1. **Default Ports** - ל-MongoDB יש מס' פורטים שלרוב נשארות קבועים:

- **27017** - זוהי יציאת ברירת המחדל של mongod ו-mongos. זהו השירות (service) של MongoDB ואפשר ואף רצוי לשנות אותו ע"י הפקודה: port או port-- או ע"י שינוי קובץ הקונפיגורציה. (/etc/mongodb.conf):

```
fingers@5fingers: ~
# Where to store the data.

# Note: if you run mongodb as a non-root user (recommended) you may
# need to create and set permissions for this directory manually,
# e.g., if the parent directory isn't mutable by the mongodb user.
dbpath=/var/lib/mongodb

#where to log
logpath=/var/log/mongodb/mongodb.log

logappend=true

bind_ip = 127.0.0.1
port = 27017
~
~
~
```

- **27018** - זוהי יציאת ברירת המחדל כאשר מריצים עם: shardsvr--
- **27019** - זוהי יציאת ברירת המחדל כאשר מריצים עם: configsvr--
- **28017** - זוהי ברירת המחדל (אולי המוכרת מכולן) דרך הדפדפן.

ריצה על פורט כברירת מחדל כשלעצמה הינה מצביעה על כשל אבטחה, אבל זה, בנוסף לשאר הסעיפים שיגיעו בהמשך (ובייחוד לסעיף 2) - מדובר בקטסטרופה, סריקה ב-Shodan מראה כי לפחות 51,451 שרתים מאזינים על פורט זה:



2. **Authentication Weakness** - כברירת מחדל בסיס הנתונים נוצר **ללא סיסמא**. כשקוראים את המדריך של mongoDB מגלים שהמפתחים החליטו שעניין האבטחה צריך להישאר בצד האפליקציה בלבד. נשמע מוזר בייחוד לאור מקרים כה רבים בהם עניין אבטחה בבסיסי נתונים RDBMS הוכח כנחוץ מאוד. כדי למנוע זאת, ראשית עלינו ליצור יוזר עבור בסיס הנתונים הרלוונטי ולתת לו ורק לו את ההרשאות הרלוונטיות אליו, לדוג':

```
fingers@5fingers: ~
> show dbs
5Fingers        0.078GB
Mongo-Shmongo   0.078GB
admin           0.078GB
digitalwhisper  0.078GB
local           0.078GB
> use digitalwhisper
switched to db digitalwhisper
> db.createUser({'user':'Mika', 'pwd':'qwer123', roles:['readWrite']});
Successfully added user: { "user" : "Mika", "roles" : [ "readWrite" ] }
>
```

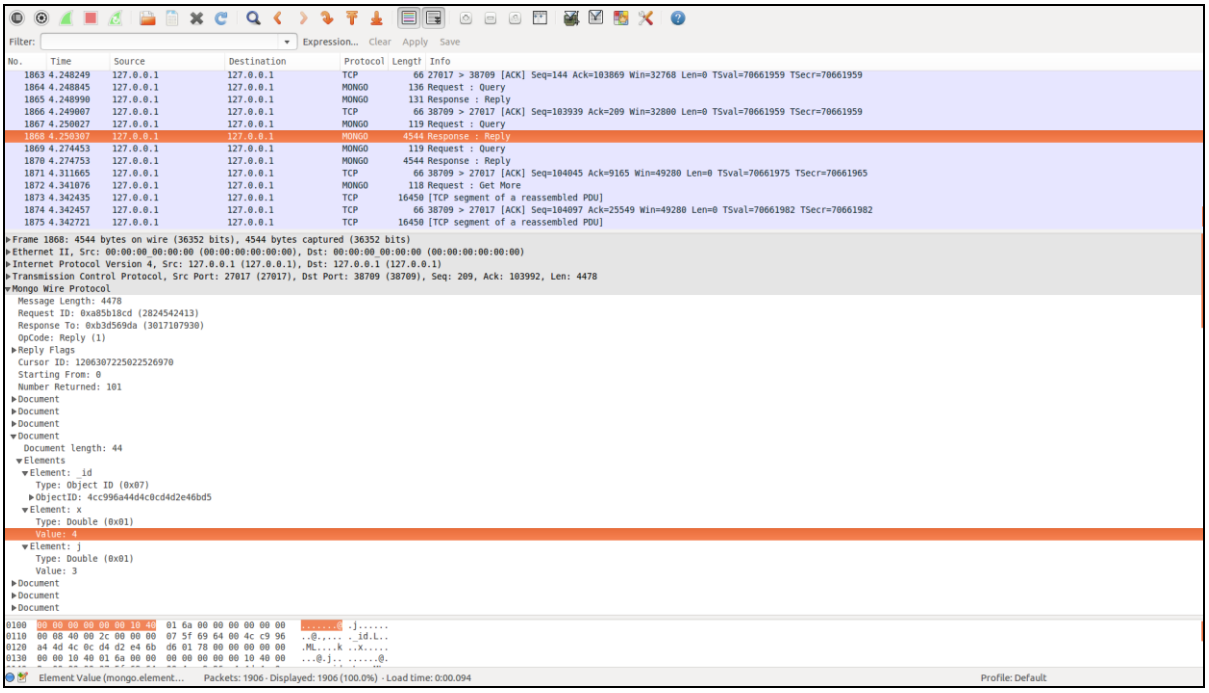
ליוזר Mika יש הרשאות קריאה/כתיבה רק! לבסיס הנתונים של digitalwhisper. (במקרה של הראשאת קריאה בלבד יש לרשום read). לאחר מכן, בקובץ הקונפיג (שהוזכר בסעיף הקודם) ישנו פרמטר נוסף הנקרא auth,

```
fingers@5fingers: ~
# mongod.conf

#where to log
logpath=/var/log/mongodb/mongod.log

logappend=true

# fork and run in background
fork=true

#port=27017

dbpath=/var/lib/mongo

# location of pidfile
pidfilepath=/var/run/mongodb/mongod.pid

# Listen to local interface only. Comment out to listen on all interfaces.
bind_ip=127.0.0.1

# Disables write-ahead journaling
# nojournal=true

# Enables periodic logging of CPU utilization and I/O wait
#cpu=true

# Turn on/off security.  Off is currently the default
#noauth=true
#auth=true
```

יש לאפשר אותו ע"י מחיקת #. כעת כל התחברות אל בסיס הנתונים תחוייב לעבור אימות.

3. **Authorization Weaknesses** - חולשה ביצירת משתמשים חדשים. כל משתמש חדש שנוצר מקבל כברירת מחדל הרשאת קריאה לכל בסיס הנתונים. דמיינו מצב בו משתמש עם הרשאות נמוכות יכול לקרוא את הסיסמא של היוזר Admin. כפועל יוצא של שני החולשות שהוזכרו זה עתה, ניתן להבין שיוזר עם הרשאות **admin** (קריאה/כתיבה) נוצר מלכתחילה ללא סיסמא.

4. המידע נשלח **Clear Text** - כל הנתונים נשלחים באופן שאינו מוצפן, כך ניתן ללכוד את הנתונים בהתקפת ARP Poisoning או בהתקפת MITM כזאת או אחרת:



5. אין הצפנה של הנתונים - נכון לעכשיו אין הצפנת נתונים ב-mongoDB. הנתונים ב-mongoDB נשמרים במערכת קבצים יעודית הנקראת GridFS ונתיב ברירת המחדל של הקבצים הינו: /data/db/ (או בספיריה C:\data\db כאשר מריצים ממערכת הפעלה WINDOWS). למעשה אם נריץ את הפקודה:

```
ps -xa | grep mongod
```

```
fingers@5fingers: ~
fingers@5fingers:~$ ps -xa | grep mongod
 5018 ?        Ssl    1:20 /usr/bin/mongod --config /etc/mongod.conf
 5453 ?        Sl     0:35 wireshark /tmp/mongodb.pcap
 8713 pts/0    S+     0:00 grep --color=auto mongod
fingers@5fingers:~$
```

נוכל לראות שאין איזכור ל-dbpath-- כלומר אנחנו מוכוונים אל נתיב ברירת המחדל. על מנת לשנות את נתיב ברירת המחדל יש להריץ את הפקודה:

```
mongod --dbpath /usr/local/mongodb-data
```

כעת הנתונים יאוחסנו בנתיב: /usr/local/mongodb-data. אך מכיוון שהנתונים לא מוצפנים עכשיו נשאלת השאלה האם משתמש שלא מורשה לגשת בצורה ישירה (דרך mongoDB) אל הנתונים, יכול בכל זאת לגשת אליהם?

אם Mongo או כל כלי אחר שמתחבר לmongos או mongod מנוהל ללא sslCAFile--, כשננסה לבצע ייצוא של נתונים זה לא יתבצע אימות משתמש, כלומר כל משתמש יכול למעשה לבצע export לנתונים.

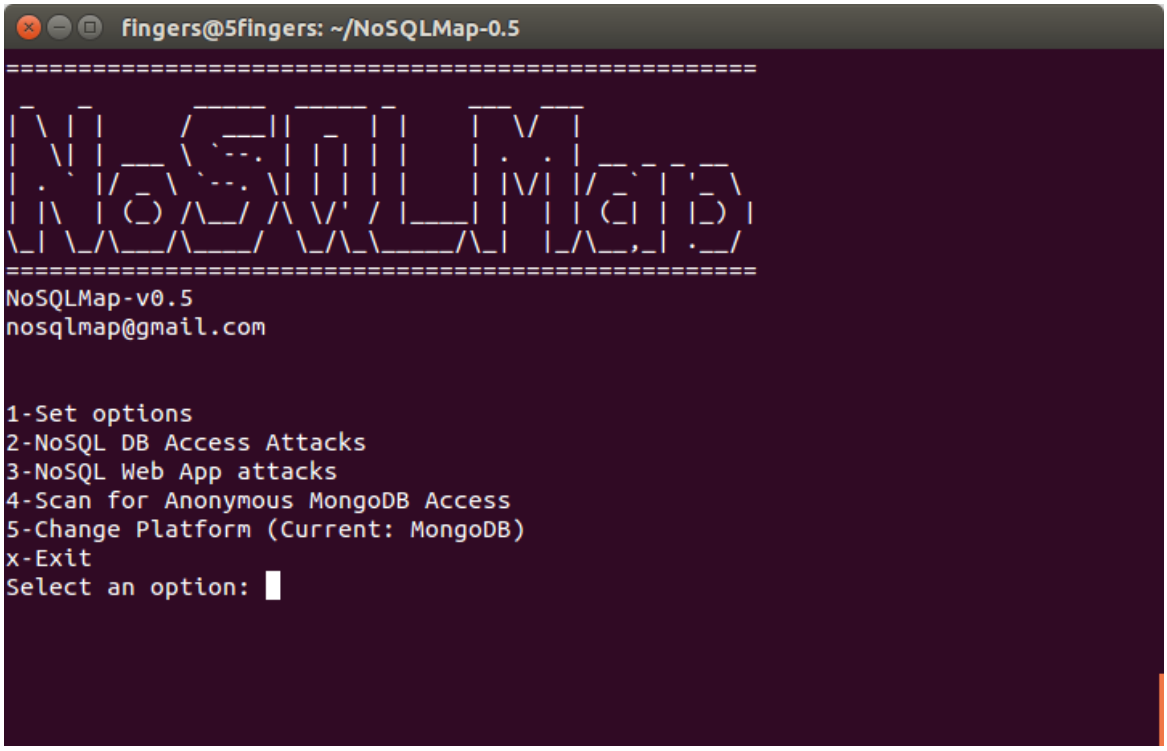
```
mongoexport -d digitalwhisper -c MyData --out /usr/local/myRecords.json
```

```
fingers@5fingers: ~
{ "_id" : "01001", "userName" : "bill_c", "FirstName" : "Bill", "LastName" : "Clinton" }
{ "_id" : "01002", "userName" : "Carolus69", "FirstName" : "Carolus", "LastName" : "Magnus" }
{ "_id" : "01003", "userName" : "5Fingers", "FirstName" : "some", "LastName" : "one" }
{ "_id" : "01004", "userName" : "Lobengulu", "FirstName" : "Lobengulu", "LastName" : "King of zulu" }
{ "_id" : "01005", "userName" : "tisf", "FirstName" : "Yuval", "LastName" : "Nativ" }
"/usr/local/myRecords.json" 5 lines, 458 characters
```

NoSQLMap:

NoSQLMap הינו פרויקט קוד פתוח מבית היוצר של ברנרדו דאמל (Bernardo Damele) ומירוסלב סטאמפר (Miroslav Stampar) הידועים כמי שהביאו לנו את SQLMap. הפרויקט החדש שלהם NoSQLMap נכתב בפייטון ומטרתו לאגד התקפות הזרקה למסדי נתוני של NoSQL ולנצל את חולשות ברירת המחדל שלהם.

בואו נראה זאת הלכה למעשה על בסיס הנתונים שלנו ב-mongoDB. ראשית, את הכלי ניתן להוריד כאן. לפני הרצתו וודאו שיש ברשותכם התקנה של Metasploit Framework ו-PyMongo. כעת, ברוכים הבאים:



שימו לב לפרט חשוב כבר במסך הפתיחה, mongoDB מוגדר כפלטפורמת ברירת המחדל, נכון לגרסא האחרונה ניתן לבחור בין mongoDB ל-CouchDB. בעתיד האפשרויות יתרחבו.

לאחר שמסך הפתיחה הופיע, יש לבחור באפשרות הראשונה (לחיצה על המספר 1) על מנת לצפות באפשרויות:

```
fingers@5fingers: ~/NoSQLMap-0.5
 _   _       _____  _____ _     ___  ___
| \ | |     /  ___||  _  | |    |  \/  |
|  \| | ___ \ `--. | | | | |    | .  . | __ _ _ __
| . ` |/ _ \ `--. \| | | | |    | |\/| |/ _` | '_ \
| |\  | (_) /\__/ /\ \/' / |____| |  | | (_| | |_) |
\_| \_/\___/\____/  \_/\_\_____/\_|  |_/\__,_| .__/
=====================================================
NoSQLMap-v0.5
nosqlmap@gmail.com


1-Set options
2-NoSQL DB Access Attacks
3-NoSQL Web App attacks
4-Scan for Anonymous MongoDB Access
5-Change Platform (Current: MongoDB)
x-Exit
Select an option: 1



Options
1-Set target host/IP (Current: Not Set)
2-Set web app port (Current: 80)
3-Set App Path (Current: Not Set)
4-Toggle HTTPS (Current: OFF)
5-Set MongoDB Port (Current : 27017)
6-Set HTTP Request Method (GET/POST) (Current: GET)
7-Set my local MongoDB/Shell IP (Current: Not Set)
8-Set shell listener port (Current: Not Set)
9-Toggle Verbose Mode: (Current: OFF)
0-Load options file
a-Load options from saved Burp request
b-Save options file
x-Back to main menu
Select an option: 1
Enter the host IP/DNS name: 10.10.10.2
```

הזנתי את ה-IP עליו נמצא mongoDB וכפי שאפשר לראות NoSQLMap מספק לי רשימת אפשרויות תקיפה עבור web application וכמובן שגם עבור תקיפה של מערכת הניהול של mongoDB כפי שהוזכר במאמר זה קודם לכן, כברירת מחדל מותקן mongoDB ללא סיסמא וכל אחד יכול לגשת לכל מקום.

לכלי אפשרויות רבות, לא נרחיב עליהם במאמר זה, במידה ותרצו, גשו לאתר הכלי:

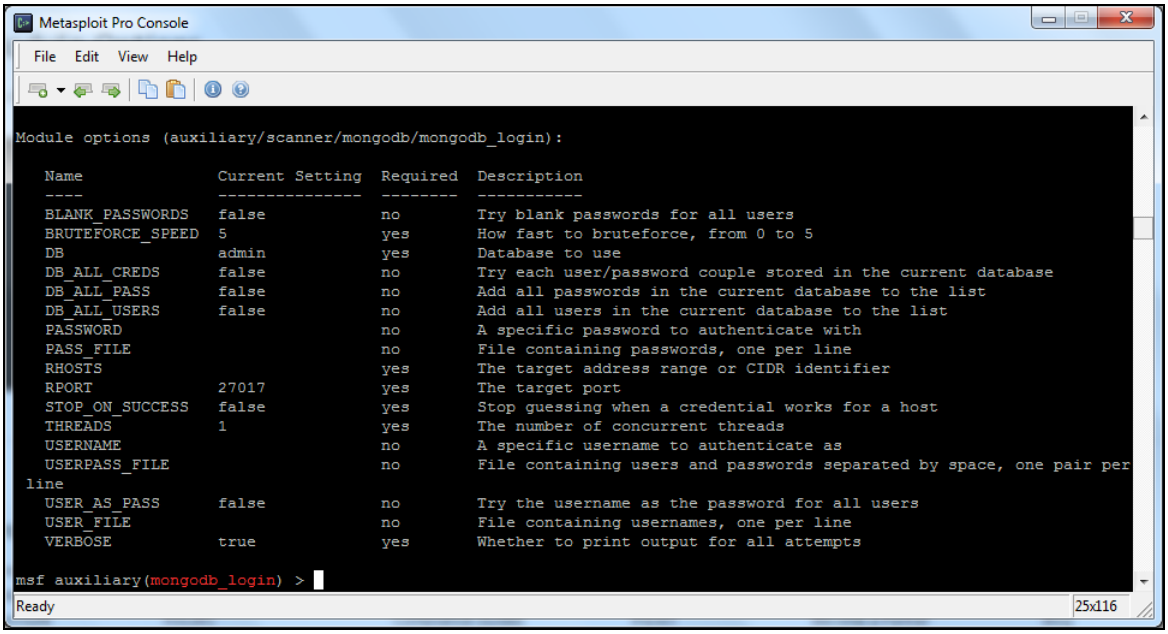
http://www.nosqlmap.net

ובו תוכלו ללמוד על כלל האפשרויות של הכלי.

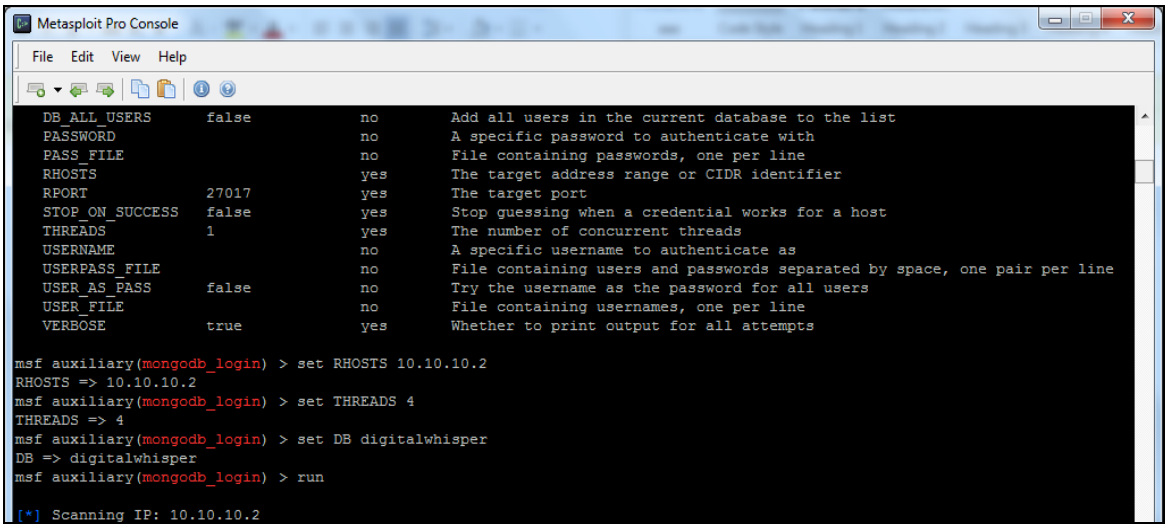
כלי נוסף הקיים לתקיפת מסדי נתונים מבוססי MongoDB קיים ב-Metasploit. המערכת מכילה כלי עזר מאוד שימושי בשם "MondoGB_Login", הוא ממוקם ב:

```
use auxiliary/scanner/mongodb/mongodb_login
```

הכלי נועד לבדיקת סיסמאות בעת שלב האימות:



לאחר בחירת ה-auxiliary שימו לב לאפשריות שיש בו.



סיכום

בסיסי נתונים מסוג NoSQL הם לא תחליף לבסיסי הנתונים הרלציונים הקיימים היום. המוטיבציה להשתמש בכל אחד מהם שונה לחלוטין ושניהם ימשיכו ללכת יד ביד עוד הרבה מאוד שנים.

בסיסי נתונים מסוג NoSQL מתחילים לצבור תאוצה בקרב חברות וארגונים רבים וצריכים להיות מודעים ולתת דגש רב לנושא האבטחה לבסיסי נתונים אלה בייחוד כשהתחום עוד יחסית בתחילת בתנופה שלו.

במאמר זה נגענו בלא מעט נקודות הקשורות לאבטחת מידע בכל הנוגע ל-MondoDB, נקודה נוספת ששווה להסתכל עליה הינה הקישור הבא:
http://blog.mongodirector.com/10-tips-to-improve-your-mongodb-security/

מדובר בריכוז של כעשרה צעדים שניתן לבצע על מנת להקשיח ולהעמיק את האבטחה בשרת ה-Mongo על מנת להגיע לרמת אבטחה גבוה יותר ממה שהמערכת מציעה כברירת מחדל.

תודות

- תודה ענקית לאישתי המדהימה נטלי - כל מה שאני אגיד עכשיו יגמד לכמה שאני מעריך את העזרה שלך, אז תודה רבה אהובתי.
- תודה ענקית ליובל (tisf) נתיב על הדרבון, העזרה והמוטיבציה שנתת לי לאורך כתיבת המאמר.

מקורות מידע / קישורים לקריאה נוספת

- http://2012.zeronights.org/includes/docs/Firstov%20-%20Attacking%20MongoDB.pdf
- http://blogs.adobe.com/security/files/2011/04/NoSQL-But-Even-Less-Security.pdf
- https://www.youtube.com/watch?v=Cy8-EYS3HeM
- https://media.blackhat.com/bh-us-11/Sullivan/BH_US_11_Sullivan_Server_Side_WP.pdf
- http://blog.spiderlabs.com/2013/03/mongodb-security-weaknesses-in-a-typical-nosql-database.html
- http://blog.mongodirector.com/10-tips-to-improve-your-mongodb-security/