BBQSQL

Ben Toews
Scott Behrens
Who are we?

- Ben Toews
  - Security Consultant / Researcher at Neohapsis
- Scott Behrens
  - Security Consultant / Researcher at Neohapsis
Why are we here?

● **BBQSQL**
  ○ New dog, old trick
    ■ Exploits Blind SQL Injection
  ○ New dog, new trick
    ■ Fast
    ■ Easy
    ■ Gets those hard to reach spots
SQL What?

- **Structured Query Language (SQL)**
  - Language for interacting with database
- **SQL Injection**
  - Inject syntax into an application's SQL queries
Basic SQL Injection

Normal Case:
UNAME = "mastahyeti"
PASS = "s3cret"
QUERY = "select * from users where pass=md5('"+PASS+"') and uname='"+UNAME+"'";

QUERY evaluates to:
select *
from users
where pass=md5('secret')
and uname='mastahyeti'
Basic SQL Injection

SQL Injection Case:
UNAME = "pwned' or '1'='1";
PASS = "pwned";
QUERY = "select * from users where pass=md5('"+PASS+"') and uname='"+UNAME+"'";

QUERY evaluates to:
select *
from users
where pass=md5('pwned')
and uname='pwned' or '1'='1"
Blind SQL Injection

- Still trying to alter SQL syntax
- Dumping database
- More complex SQL syntax
Blind SQL Injection Case:
UNAME = "' or (ASCII(SUBSTR(SELECT user(), 1,1))>63) --";
PASS = "";
QUERY = "select * from users where pass=md5('"+PASS+'') and uname='"+UNAME+"'";
QUERY evaluates to:
select *
from users where pass=md5('')
and uname='" or (ASCII(SUBSTR(SELECT user(), 1,1))>63) --'
Blind SQL Injection

select *
from users where pass=md5('') and
  uname=''
  or (ASCII(
    SUBSTR(
      SELECT user()
    ,1,1)
  )>63
  ) --'

Blind SQL Injection

- Binary (or other) search for each character
- One character at a time
- Time consuming
Blind SQL Injection

- Lots of excellent tools out there
  - sqlmap, sqlninja, BSQL Hacker, the Mole, Havij, ...
- Lots of great features
  ^^^^^^^  good job guys...
- If these tools don't work
  - You end up writing a custom script, test, debug, test, debug...
- What if there was a way to simplify tricky Blind SQL Injection attacks...
BBQSQL

doesn't care about your data!
doesn't care about your database!

Images from http://www.freedigitalphotos.net/
BBQSQL

- Exploits Blind SQL Injection
- For those hard to reach spots
- Semi-automatic
- Database agnostic
- Versatile
- Fast
- Fast
- Fast
- Did we mention it is fast?
BBQSQL: Use

- Must provide the usual information
  - URL
  - HTTP Method
  - Headers
  - Cookies
  - Encoding methods
  - Redirect behavior
  - Files
  - HTTP Auth
  - Proxies
  - ...

(NEOHAAPSIS)
BBQSQL: Use

• Provide two additional pieces of info
  ○ Specify where the injection goes
  ○ Specify what syntax we are injecting
BBQSQL: Use

- The injection can go ANYWHERE:
  - url => "http://google.com?vuln='${query}""
  - data => "user=foo&pass=${query}"
  - cookies => {'PHPSESSID':'123123','FOO':'BAR${query}'}

- doesn't understand data
- doesn't care about your annoying:
  - serialization format
  - processes and rules
  - encodings
BBQSQL: Use

- The query specifies how to do binary search:
  - query => 
    "' and ASCII(SUBSTR((SELECT data FROM data LIMIT 1 OFFSET ${row_index:1}), ${char_index:1}, 1))${comparator:>}${char_val:0} #"

- Database agnostic

- Doesn't care about your annoying:
  - SQL syntax
  - Charset limitations
  - IDS/IPS
BBQSQL: Speed

- Concurrent HTTP requests
- Multiple search algorithms
  - Binary search
  - Frequency based search
BBQSQL: Speed

- Concurrent HTTP requests
- Multiple search algorithms
  - Binary search
  - Frequency based search
BBQSQL: grequests

grequests = gevent + requests
BBQSQL: grequests

grequests = gevent + requests
"gevent is a coroutine-based Python networking library that uses greenlet to provide a high-level synchronous API on top of the libevent event loop"

-http://gevent.org
BBQSQL: gevent

- Coroutine ~ function
- You spawn many simultaneous coroutines
- Only one runs at a time
- When a coroutine encounters blocking (network IO) it yields and allows the next coroutine to run while it waits
- This forms an event-loop
- Functionally, it appears to act like threading
BBQSQL: grequests

grequests = gevent + requests
BBQSQL: requests

"HTTP For Humans"
-docs.python-requests.org

- **Awesome** HTTP API built on top of urllib3 in Python
- Written/maintained by Kenneth Reitz
  - API designing badass
BBQSQL: grequests

grequests = gevent + requests
BBQSQL: grequests

Good Evented HTTP for Python
BBQSQL: Speed

- Concurrent HTTP requests
- Multiple search algorithms
  - Binary search
  - Frequency based search
BBQSQL: Binary Search

Average Case: $O(\log(n))$
BBQSQL: Speed

- Concurrent HTTP requests
- Multiple search algorithms
  - Binary search
  - Frequency based search
BBQSQL: Linear Search

Average Case: $O(n/2)$
BBQSQL: Frequency

- Analysed lots of books, source code, CCs, SSNs :
P
- Most common characters are [', 'e', 't', 'o', 'a']
- Most likely characters to follow 'e' are [' ', 'r', 'n']
BBQSQL: Frequency

- Very fast against non-entropic data:
  - English
    - ~10 requests/character
  - Python
    - ~8 requests/character
  - Credit card numbers
    - ~5.5 requests/character

- VS. binary search
  - English
    - ~12 requests/character
BBQSQL: UI

- UI is built using source from Social Engineering Toolkit (SET)
  - Thanks Dave (ReL1K) Kennedy!
- Input validation is performed on each configuration option in real time to prevent snafu
  - You don't have to wait till you type up a huge request on the CLI and find out your 600 char POST data is malformed!
BBQSQL: UI

- Configuration files can be imported and exported through UI or CLI
  - Uses ConfigParser so easy to work with
- Can export attack results as CSV file
Credits

- Wikipedia (math is hard)
- Neohapsis Labs
- Image links are embedded in presentation
- ReL1K - SET [https://www.trustedsec.com/downloads/social-engineer-toolkit/]
Thanks

Ben Toews - @mastahyeti
Scott Behrens - @helloarbit

Neohapsis(.com) << Hiring
<< bonus4us

BBQSQL

github.com/neohapsis/bbqsql