Beyond SQL Injection

Network Attacks to Keep You Up at Night

Kevin Feasel

Who Am I? What Am I Doing Here?

- Database Administrator
  - SQL Server DBA
  - SSIS developer
  - Currently working for Aetna
    - Standard employer disclaimer
  - Catallaxy Services
    - http://www.catallaxyservices.com
- Security Nut
- Cyclist
- Occasional world traveler

Protecting That Which Is Yours

- DBAs are data stewards: we protect the data
- How far should we go to protect it?
- What about the really important stuff?

Protecting The Data: The Basics

- Logins have strong passwords
- Follow least privilege for accounts
- Use roles and groups to create database access control list (ACL)-like substances
- Track login failures (and successes?)
- Encrypt databases which require encryption
- Encrypt and securely store backups
- Protect against SQL injection in code

Moving Beyond The Basics

- Key assumption: your organization already handles the basics fairly well
  - If not, I know of a ~~tax shelter~~ consulting firm which can help…
- More advanced attacks (from Derbycon 2012)
  - Reversing SQL authentication passwords
  - SQL Server man in the middle attack

Reversing SQL Authentication

- Credit: Nicolle Neulist
  - http://www.rogueclown.net
- SQL authentication “encryption” is terrible
- Good encryption: assume your attacker has perfect knowledge of everything but the key; your algorithm should still not be reversable

Reversing SQL Authentication

- SQL authentication:
  - Expand each password byte to two bytes
  - Swap the higher and lower 4 bits of each byte
  - XOR each byte with A5
- There is no key; there is only a process!

Step 0: Select A Character

- Start with the character “p”
- ASCII: p
- Hex: 0x70
- Binary: 01110000

Step 1: Expand to Two Bytes

- Append an empty byte to each byte in the plaintext (in this case, “p”)
- Hex: 0x70 0x00
- Binary: 01110000 00000000

Step 2: Swap Higher and Lower Bits

• Old Hex:	0x70	0x00
• Old Bin:	01110000	00000000
• Hex:	0x07	0x00
• Binary:	00000111	00000000

Step 3: XOR with A5

XOR with A5 (1010 0101)

• Old Hex:	0x07	0x00
• Old Bin:	00000111	00000000
• XOR:	10100101	10100101
• Binary:	10100010	10100101
• Hex:	0xA2	0xA5

Decrypting SQL Authentication

- Drop every even byte
  - Alternatively, drop all 0xA5 bytes—0x00 is never a valid character in a password
- XOR each byte with 0xA5
- Swap the higher and lower bits of each byte

Coding This Solution

- C code: http://www.securiteam.com/tools/6Q00I0UEUM.html
- Python Code: http://rogueclown.net/sqlbreak.py
- Powershell Code: DEMO

Risk Factor And Mitigation Strategy

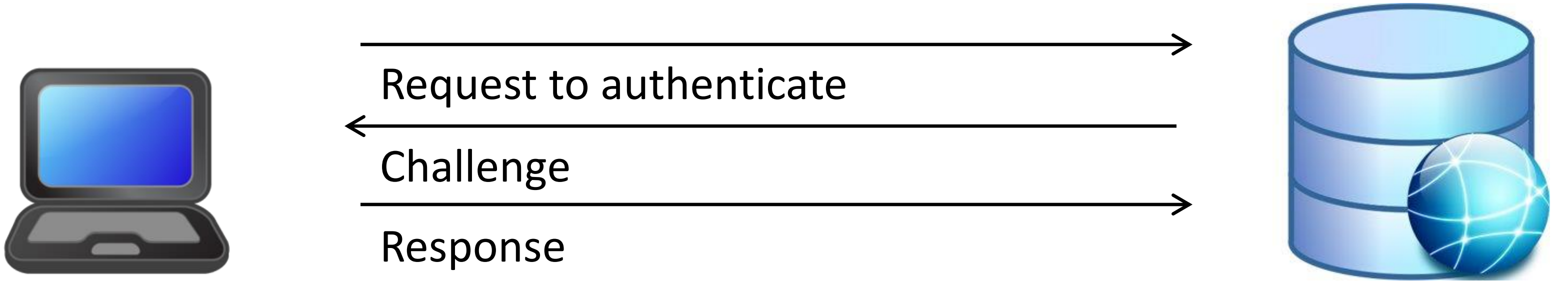
- Risk factor: Low
  - This attack was released to the public by 2004
  - Fixed with SQL Server 2005: SQL authentication credentials sent encrypted using a self-signed certificate
- Mitigation Strategies:
  - Switch to Windows Authentication
    - Windows authentication using Kerberos is solid
  - Limit SQL authentication account privileges
  - Disable sa account
  - Audit logins and correlate accounts to IP addresses
  - Only accept traffic from known good IP addresses

Man In The Middle Attacks

- Credit: Laszlo Toth and Ferenc Spala
  - http://soonerorlater.hu
- SQL Server passes data using Tabular Data Stream (TDS)
- tdsproxy allows us to hijack a SQL Server connection using a man in the middle attack
  - http://soonerorlater.hu/download/tdsproxy_v0.1.tar.gz

What Is A Man In The Middle Attack?

- Normal connection:

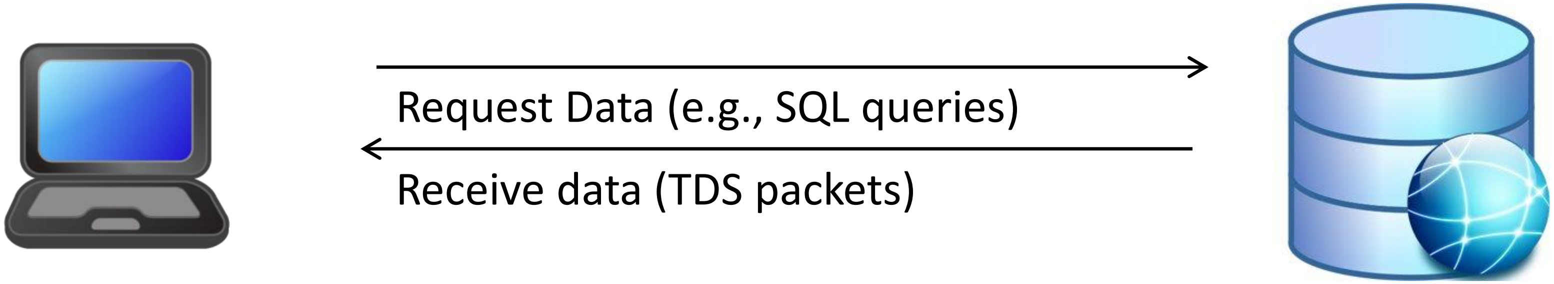


Request to authenticate

Challenge

Response

Simplified authentication model

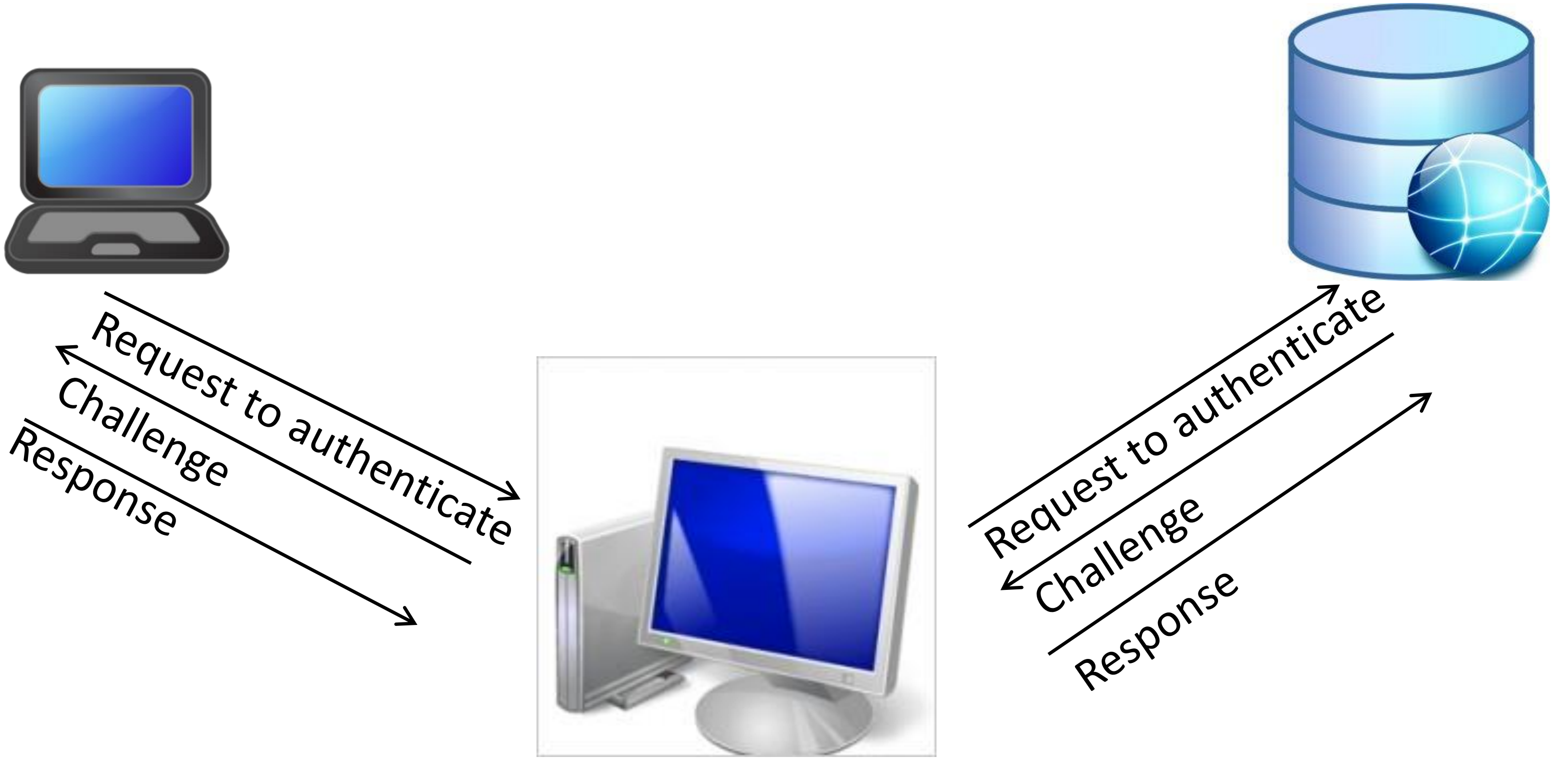


Request Data (e.g., SQL queries)

Receive data (TDS packets)

What Is A Man In The Middle Attack?

- Man In The Middle: attacker interposes its device between victim and resource



Request to authenticate

Challenge

Response

Request to authenticate

Challenge

Response

What Can A MITM Do?

- Passive attack:
  - Watch transmissions between victim and server
    - Collect the same data the victim receives
  - Collect credentials for later use
- Active attacks:
  - Perform additional queries against the server **using the victim's credentials**
  - Disconnect the victim (denial of service)

How To Perform A MITM Attack

- ARP cache poisoning
  - Address Resolution Protocol (ARP): used to connect OSI layer 3 (Network) to OSI layer 2 (Data Link)
    - In other words, link IP addresses to MAC addresses
  - ARP cache: local table connecting IP addresses to MAC addresses
    - arp -a

How To Perform A MITM Attack

- ARP has **no** authentication and **no** verification mechanism
  - Many devices accept ARP replies before sending out requests!
- Broadcast “here is the SQL Server” messages out from the attacker’s MAC address and IP combination
  - ARP entries expire after a certain time, so even if a victim has an entry, keep at it
  - NOTE: must be done on a local network!

tdsproxy

- tdsproxy acts as a proxy server for SQL Server Tabular Data Stream (TDS) traffic
  - Remember: all SQL Server data transmits as TDS packets
- The attacker can turn his tdsproxy-hosting machine into a SQL Server repeater of sorts
  - All traffic going to tdsproxy can be sent along to the SQL Server instance
- Client versions need to be the same as what the victim is using, however

Risk Factor And Mitigation Strategy

- Risk factor: medium
  - Active and exploitable
  - Not a trivial exercise, although much of the code is in Metasploit
  - Not a flaw within SQL Server itself!
- Mitigation strategies:
  - arpwatch: monitor ARP traffic
  - Have external clients connect via VPN
  - Not much we can do from within SQL Server

Conclusions

- Bad news:
  - Even with a fully secure SQL Server instance, there are still ways in
    - Example not shown: brute force attack against logins
  - We need to talk to our network guys more
- Good news: SQL Server is a very secure product
  - Fewer vulnerabilities than, e.g., Oracle
  - Most published vulnerabilities are low-impact (SQL authentication being broken) or external to SQL Server as such (tdsproxy)