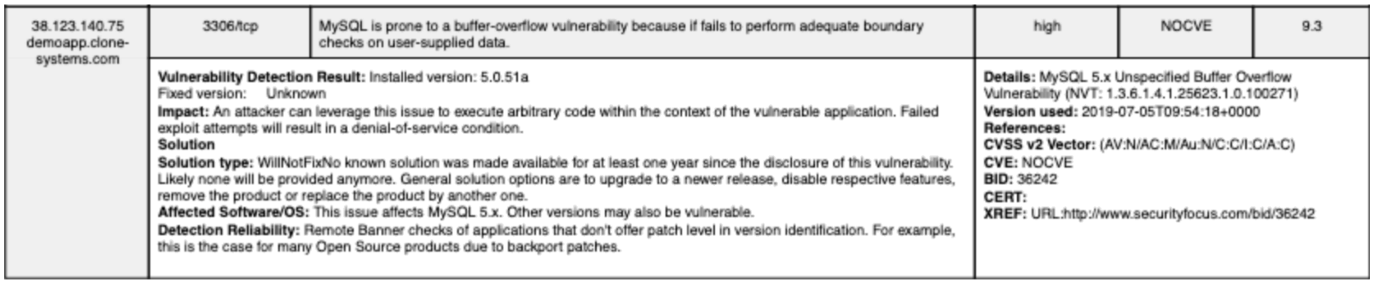
PCAP

- For this assignment I chose the MySQL protocol. This protocol is used as communications between a MySQL database & client software.

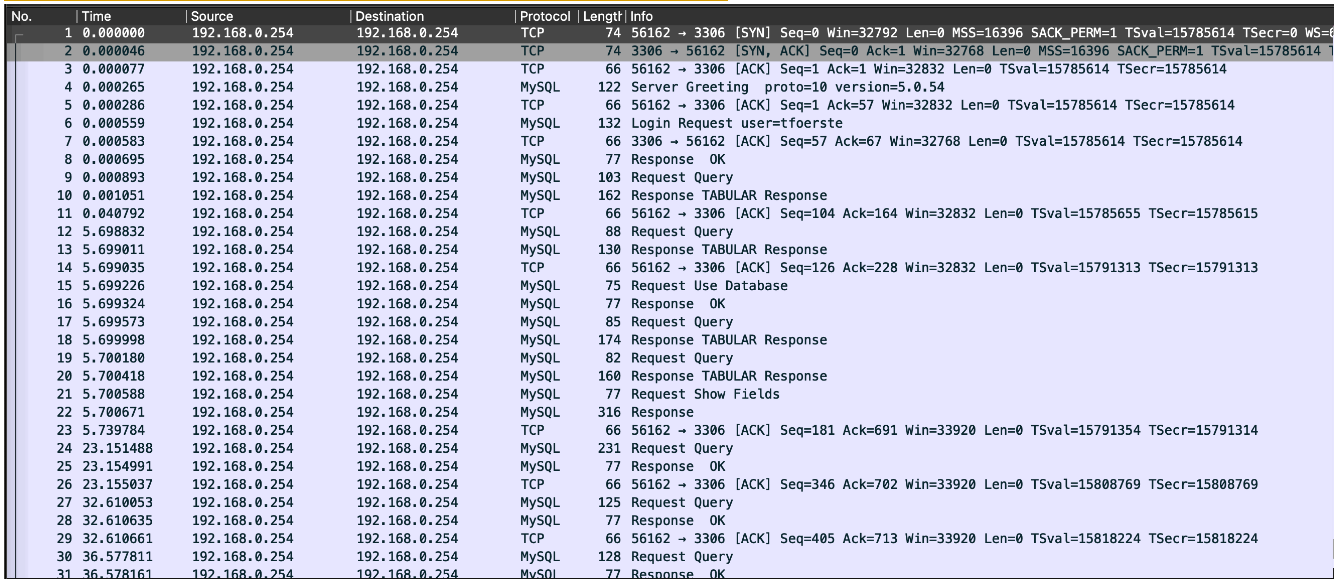


38.123.140.75 demoapp.clone-systems.com	3306/tcp	MySQL is prone to a buffer-overflow vulnerability because if fails to perform adequate boundary checks on user-supplied data.	high	NOCVE	9.3
	Vulnerability Detection Result: Installed version: 5.0.51a Fixed version: Unknown **Impact:** An attacker can leverage this issue to execute arbitrary code within the context of the vulnerable application. Failed exploit attempts will result in a denial-of-service condition. **Solution** **Solution type:** WillNotFixNo known solution was made available for at least one year since the disclosure of this vulnerability. Likely none will be provided anymore. General solution options are to upgrade to a newer release, disable respective features, remove the product or replace the product by another one. **Affected Software/OS:** This issue affects MySQL 5.x. Other versions may also be vulnerable. **Detection Reliability:** Remote Banner checks of applications that don't offer patch level in version identification. For example, this is the case for many Open Source products due to backport patches.		**Details:** MySQL 5.x Unspecified Buffer Overflow Vulnerability (NVT: 1.3.6.1.4.1.25623.1.0.100271) **Version used:** 2019-07-05T09:54:18+0000 **References:** **CVSS v2 Vector:** (AV:N/AC:M/Au:N/C:C/I:C/A:C) **CVE:** NOCVE **BID:** 36242 **CERT:** **XREF:** URL:http://www.securityfocus.com/bid/36242		

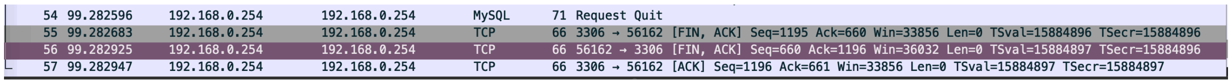
1. Vulnerability Assessment
   - Look for vulnerabilities specific to one port (of your choosing) exposing risk to the systems, to report and recommend a remediation.
     - In this example, the `MySQL` protocol is prone to a buffer-overflow vulnerability because it does not check user-supplied data to ensure that it is within the specific bounds/limitations allocated to the database.
     - More on buffer-overflows:
       - A buffer-overflow occurs when ***software writing data to a buffer overflows the buffer's capacity***, meaning however much memory was allocated to the buffer. A frequent type of buffer-overflow is the stack-overflow, in which the stack acts as a buffer that data is written to, and that data overflows the capacity of the stack. Many may know stack-overflow from the forum, but it actually got it's name from this notorious vulnerability.
   - Notice the URL and IP of the scan target, that is the system that was scanned for vulnerabilities.
     - URL
       - `demoapp.clone-systems.com`
     - IP address
       - `38.123.140.75`
   - What is the port # and Protocol (TCP or UDP) found vulnerable?
     - Port
       - `3306`
     - Protocol
       - `TCP`
   - Why are they finding a vulnerability, what is the risk to the client system?
     - The vulnerability is that the `MySQL` protocol does not have a limitation to how much data a user may enter. In most regular work environments, this won't often be a problem. However, attackers can use this to their advantage. By overflowing the buffer, they can potentially overwrite memory in the buffer, potentially being able to execute malicious code. This is

done through overwriting memory in areas known to contain executable code. The attacker can then run their own malicious code which can drastically affect a program or database.

- The severity and score follow the CVE scoring system to help drive remediation priority
  - Severity
    - `High`
  - CVE
    - `NOCVE`
  - CVSS score
    - `9.3`
  - Potential remediation
    - The best remediation for this is to upgrade to a newer release, disable respective features, remove the product, or replace it with a newer one.

2. MySQP Protocol sample capture from Wireshark wiki.
3. 

```
No.   Time       Source          Destination     Protocol Length Info
  1 0.000000    192.168.0.254   192.168.0.254   TCP       74  56162 → 3306 [SYN] Seq=0 Win=32792 Len=0 MSS=16396 SACK_PERM=1 TSval=15785614 TSecr=0 WS=
  2 0.000046    192.168.0.254   192.168.0.254   TCP       74  3306 → 56162 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0 MSS=16396 SACK_PERM=1 TSval=15785614 T
  3 0.000077    192.168.0.254   192.168.0.254   TCP       66  56162 → 3306 [ACK] Seq=1 Ack=1 Win=32832 Len=0 TSval=15785614 TSecr=15785614
  4 0.000265    192.168.0.254   192.168.0.254   MySQL    122  Server Greeting  proto=10 version=5.0.54
  5 0.000286    192.168.0.254   192.168.0.254   TCP       66  56162 → 3306 [ACK] Seq=1 Ack=57 Win=32832 Len=0 TSval=15785614 TSecr=15785614
  6 0.000559    192.168.0.254   192.168.0.254   MySQL    132  Login Request user=tfoerste
  7 0.000583    192.168.0.254   192.168.0.254   TCP       66  3306 → 56162 [ACK] Seq=57 Ack=67 Win=32768 Len=0 TSval=15785614 TSecr=15785614
  8 0.000695    192.168.0.254   192.168.0.254   MySQL     77  Response  OK
  9 0.000893    192.168.0.254   192.168.0.254   MySQL    103  Request Query
 10 0.001051    192.168.0.254   192.168.0.254   MySQL    162  Response TABULAR Response
 11 0.040792    192.168.0.254   192.168.0.254   TCP       66  56162 → 3306 [ACK] Seq=104 Ack=164 Win=32832 Len=0 TSval=15785655 TSecr=15785615
 12 5.698832    192.168.0.254   192.168.0.254   MySQL     88  Request Query
 13 5.699011    192.168.0.254   192.168.0.254   MySQL    130  Response TABULAR Response
 14 5.699035    192.168.0.254   192.168.0.254   TCP       66  56162 → 3306 [ACK] Seq=126 Ack=228 Win=32832 Len=0 TSval=15791313 TSecr=15791313
 15 5.699226    192.168.0.254   192.168.0.254   MySQL     75  Request Use Database
 16 5.699324    192.168.0.254   192.168.0.254   MySQL     77  Response  OK
 17 5.699573    192.168.0.254   192.168.0.254   MySQL     85  Request Query
 18 5.699998    192.168.0.254   192.168.0.254   MySQL    174  Response TABULAR Response
 19 5.700180    192.168.0.254   192.168.0.254   MySQL     82  Request Query
 20 5.700418    192.168.0.254   192.168.0.254   MySQL    160  Response TABULAR Response
 21 5.700588    192.168.0.254   192.168.0.254   MySQL     77  Request Show Fields
 22 5.700671    192.168.0.254   192.168.0.254   MySQL    316  Response
 23 5.739784    192.168.0.254   192.168.0.254   TCP       66  56162 → 3306 [ACK] Seq=181 Ack=691 Win=33920 Len=0 TSval=15791354 TSecr=15791314
 24 23.151488   192.168.0.254   192.168.0.254   MySQL    231  Request Query
 25 23.154991   192.168.0.254   192.168.0.254   MySQL     77  Response  OK
 26 23.155037   192.168.0.254   192.168.0.254   TCP       66  56162 → 3306 [ACK] Seq=346 Ack=702 Win=33920 Len=0 TSval=15808769 TSecr=15808769
 27 32.610053   192.168.0.254   192.168.0.254   MySQL    125  Request Query
 28 32.610635   192.168.0.254   192.168.0.254   MySQL     77  Response  OK
 29 32.610661   192.168.0.254   192.168.0.254   TCP       66  56162 → 3306 [ACK] Seq=405 Ack=713 Win=33920 Len=0 TSval=15818224 TSecr=15818224
 30 36.577811   192.168.0.254   192.168.0.254   MySQL    128  Request Query
 31 36.578161   192.168.0.254   192.168.0.254   MySQL     77  Response  OK
```

- Important Metadata
  - Notably, the only protocols used are `MySQL` / `3306` & `TCP`.
  - On `line 7` the user requests to login to the application, using username `tfoerste`.
    - On `line 4`, we can see that the user is on version `5.0.54` and what I believe is HandshakeV2 Protocol 10.
    - The user requests multiple queries, to which they receive a `TABULAR` response, and then the action is completed in `TCP`.
    - We can also view that the user makes several requests to show fields.
    - 

```
 54 99.282596   192.168.0.254   192.168.0.254   MySQL    71  Request Quit
 55 99.282683   192.168.0.254   192.168.0.254   TCP      66  3306 → 56162 [FIN, ACK] Seq=1195 Ack=660 Win=33856 Len=0 TSval=15884896 TSecr=15884896
 56 99.282925   192.168.0.254   192.168.0.254   TCP      66  56162 → 3306 [FIN, ACK] Seq=660 Ack=1196 Win=36032 Len=0 TSval=15884897 TSecr=15884896
 57 99.282947   192.168.0.254   192.168.0.254   TCP      66  3306 → 56162 [ACK] Seq=1196 Ack=661 Win=33856 Len=0 TSval=15884897 TSecr=15884897
```

    - On `line 54` it is seen that the user quit after 99.28 seconds.