Laboratório - Atacando um banco de dados MySQL

Objetivos

Neste laboratório, você visualizará um arquivo PCAP de um ataque anterior contra um banco de dados SQL.

Parte 1: Abra o Wireshark e carregue o arquivo PCAP.

Parte 2: Veja o Ataque de Injeção SQL.

Parte 3: O Ataque de Injeção SQL continua...

Parte 4: O Ataque de Injeção SQL fornece informações do sistema.

Parte 5: O Ataque de Injeção SQL e Informações da Tabela

Parte 6: O Ataque de Injeção SQL Conclui.

Histórico/Cenário

Os ataques de injeção SQL permitem que hackers mal-intencionados digitam instruções SQL em um site e recebam uma resposta do banco de dados. Isso permite que atacantes adulterem dados atuais no banco de dados, identidades falsas e malícia diversa.

Um arquivo PCAP foi criado para você exibir um ataque anterior contra um banco de dados SQL. Neste laboratório, você visualizará os ataques de banco de dados SQL e responderá às perguntas.

Recursos necessários

- Máquina Virtual CyberOps Workstation

Instruções

Você usará o Wireshark, um analisador de pacotes de rede comum, para analisar o tráfego de rede. Depois de iniciar o Wireshark, você abrirá uma captura de rede salva anteriormente e visualizará um ataque de injeção SQL passo a passo contra um banco de dados SQL.

Parte 1: Abra o Wireshark e carregue o arquivo PCAP.

O aplicativo Wireshark pode ser aberto usando uma variedade de métodos em uma estação de trabalho Linux.

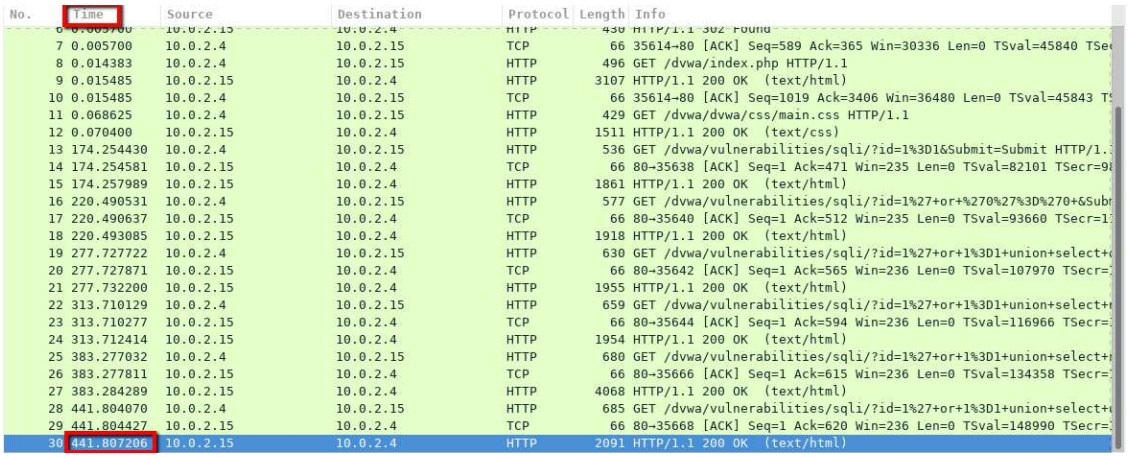
a. Inicie o CyberOps Workstation VM.

b. Clique em **Applications > CyberOps > Wireshark** na área de trabalho e navegue até o aplicativo Wireshark.

c. No aplicativo Wireshark, clique em **Open** no meio do aplicativo em Arquivos.

d. Navegue pelo diretório **/home/analyst/** e procure **lab.support.files**. No diretório **lab.support.files** e abra o arquivo **SQL_Lab.pcap**.

e. O arquivo PCAP é aberto no Wireshark e exibe o tráfego de rede capturado. Esse arquivo de captura se estende por um período de 8 minutos (441 segundos), a duração desse ataque de injeção SQL.



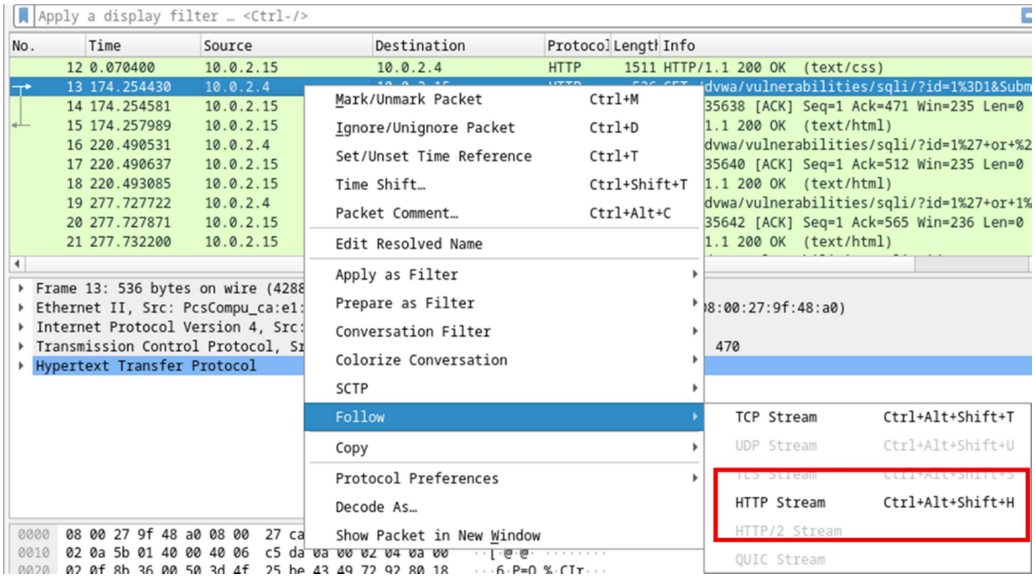
No.	Time	Source	Destination	Protocol	Length	Info
7	0.005700	10.0.2.4	10.0.2.15	TCP	66	35614→80 [ACK] Seq=589 Ack=365 Win=30336 Len=0 TSval=45840 TSe
8	0.014383	10.0.2.4	10.0.2.15	HTTP	496	GET /dvwa/index.php HTTP/1.1
9	0.015485	10.0.2.15	10.0.2.4	HTTP	3107	HTTP/1.1 200 OK (text/html)
10	0.015485	10.0.2.4	10.0.2.15	TCP	66	35614→80 [ACK] Seq=1019 Ack=3406 Win=36480 Len=0 TSval=45843 T
11	0.068625	10.0.2.4	10.0.2.15	HTTP	429	GET /dvwa/dvwa/css/main.css HTTP/1.1
12	0.070400	10.0.2.15	10.0.2.4	HTTP	1511	HTTP/1.1 200 OK (text/css)
13	174.254430	10.0.2.4	10.0.2.15	HTTP	536	GET /dvwa/vulnerabilities/sqli/?id=1%3D1&Submit=Submit HTTP/1.
14	174.254581	10.0.2.15	10.0.2.4	TCP	66	80→35638 [ACK] Seq=1 Ack=471 Win=235 Len=0 TSval=82101 TSecr=9
15	174.257989	10.0.2.15	10.0.2.4	HTTP	1861	HTTP/1.1 200 OK (text/html)
16	220.490531	10.0.2.4	10.0.2.15	HTTP	577	GET /dvwa/vulnerabilities/sqli/?id=1%27+or+%270%27%3D%270+&Subr
17	220.490637	10.0.2.15	10.0.2.4	TCP	66	80→35640 [ACK] Seq=1 Ack=512 Win=235 Len=0 TSval=93660 TSecr=1
18	220.493085	10.0.2.15	10.0.2.4	HTTP	1918	HTTP/1.1 200 OK (text/html)
19	277.727722	10.0.2.4	10.0.2.15	HTTP	630	GET /dvwa/vulnerabilities/sqli/?id=1%27+or+1%3D1+union+select+
20	277.727871	10.0.2.15	10.0.2.4	TCP	66	80→35642 [ACK] Seq=1 Ack=565 Win=236 Len=0 TSval=107970 TSecr=
21	277.732200	10.0.2.15	10.0.2.4	HTTP	1955	HTTP/1.1 200 OK (text/html)
22	313.710129	10.0.2.4	10.0.2.15	HTTP	659	GET /dvwa/vulnerabilities/sqli/?id=1%27+or+1%3D1+union+select+
23	313.710277	10.0.2.15	10.0.2.4	TCP	66	80→35644 [ACK] Seq=1 Ack=594 Win=236 Len=0 TSval=116966 TSecr=
24	313.712414	10.0.2.15	10.0.2.4	HTTP	1954	HTTP/1.1 200 OK (text/html)
25	383.277032	10.0.2.4	10.0.2.15	HTTP	680	GET /dvwa/vulnerabilities/sqli/?id=1%27+or+1%3D1+union+select+
26	383.277811	10.0.2.15	10.0.2.4	TCP	66	80→35666 [ACK] Seq=1 Ack=615 Win=236 Len=0 TSval=134358 TSecr=
27	383.284289	10.0.2.15	10.0.2.4	HTTP	4068	HTTP/1.1 200 OK (text/html)
28	441.804070	10.0.2.4	10.0.2.15	HTTP	685	GET /dvwa/vulnerabilities/sqli/?id=1%27+or+1%3D1+union+select+
29	441.804427	10.0.2.15	10.0.2.4	TCP	66	80→35668 [ACK] Seq=1 Ack=620 Win=236 Len=0 TSval=148990 TSecr=
30	441.807206	10.0.2.15	10.0.2.4	HTTP	2091	HTTP/1.1 200 OK (text/html)

Quais são os dois endereços IP envolvidos neste ataque de injeção SQL com base nas informações exibidas?

Parte 2: Exibir o Ataque de Injeção SQL.

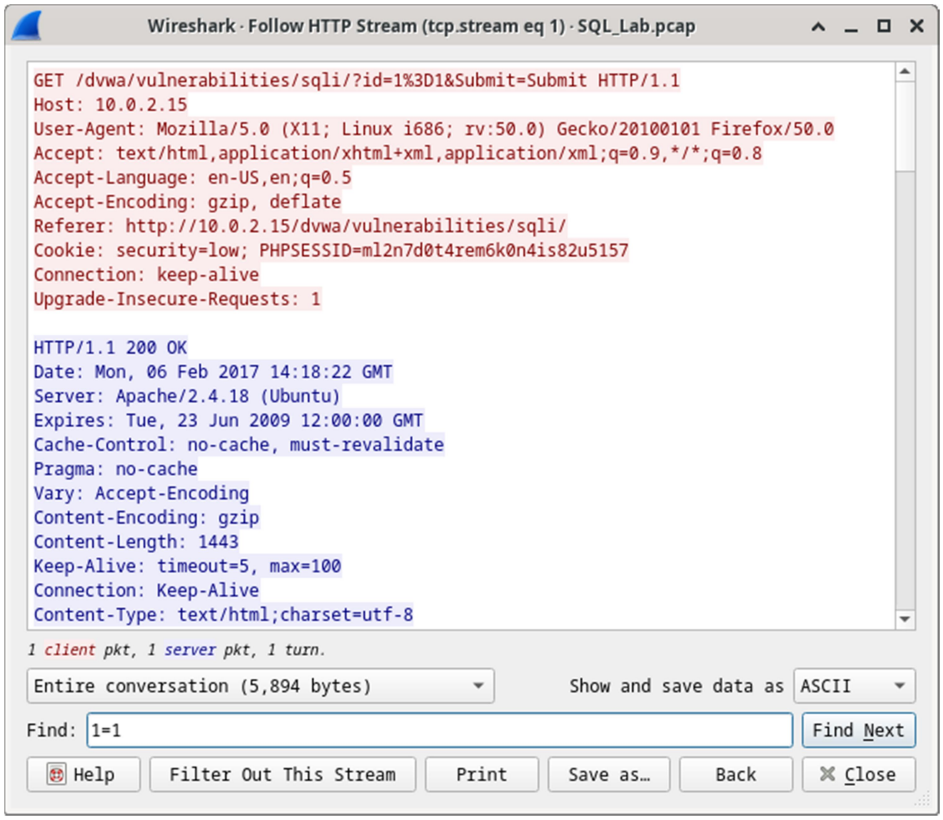
Nesta etapa, você estará visualizando o início de um ataque.

a. Na captura Wireshark, clique com o botão direito do mouse na linha 13 e selecione **Follow** > **HTTP Stream**. A linha 13 foi escolhida porque é uma solicitação HTTP GET. Isso será muito útil em seguir o fluxo de dados como as camadas de aplicativo vê-lo e leva até o teste de consulta para a injeção SQL.

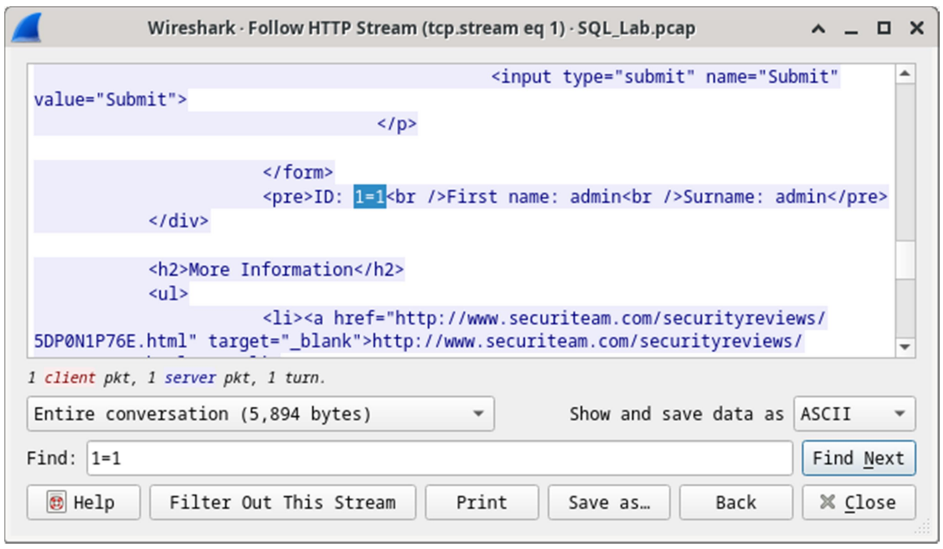


O tráfego de origem é mostrado em vermelho. A origem enviou uma solicitação GET para o host 10.0.2.15. Em azul, o dispositivo de destino está respondendo de volta à origem.

b. No campo **Localizar**, insira **1=1**. Clique em **Localizar próxima**.

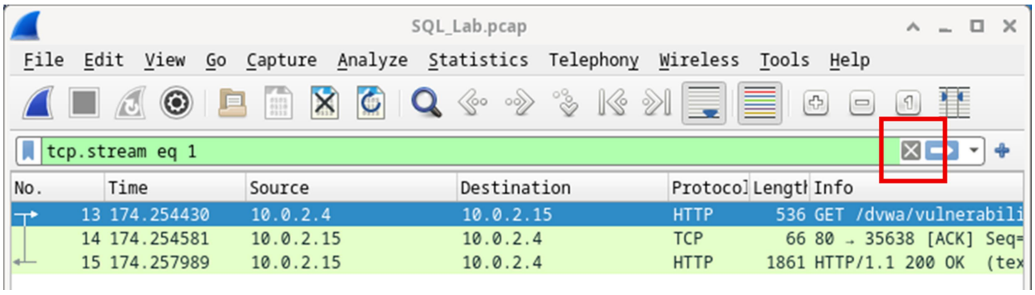


c. O invasor inseriu uma consulta (1=1) em uma caixa de pesquisa UserID no destino 10.0.2.15 para ver se o aplicativo está vulnerável à injeção de SQL. Em vez de o aplicativo responder com uma mensagem de falha de logon, ele respondeu com um registro de um banco de dados. O invasor verificou que pode inserir um comando SQL e o banco de dados responderá. A string de pesquisa 1=1 cria uma instrução SQL que será sempre verdadeira. No exemplo, não importa o que é inserido no campo, sempre será verdade.



d. Feche a janela Follow HTTP Stream

e. Clique em **Limpar filtro** de exibição para exibir toda a conversa Wireshark.



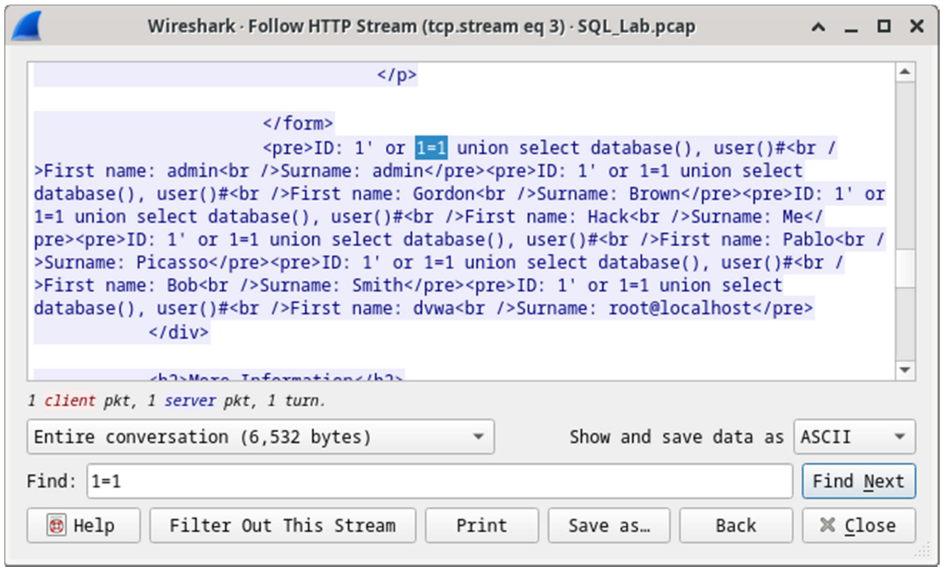
Parte 3: O Ataque de Injeção SQL continua...

Nesta etapa, você estará visualizando a continuação de um ataque.

a. Na captura Wireshark, clique com o botão direito do mouse na linha 19 e clique em **Follow** > **HTTP Stream**.

b. No campo **Find** , entre **1=1**. Clique em **Find next**.

c. O invasor inseriu uma consulta (1' ou 1=1 união select database (), user () #) em uma caixa de pesquisa UserID no destino 10.0.2.15. Em vez de o aplicativo responder com uma mensagem de falha de logon, ele respondeu com as seguintes informações:



O nome do banco de dados é **dvwa** e o usuário do banco de dados é **root @localhost**. Há também várias contas de usuário sendo exibidas.

d. Feche a janela Follow HTTP Stream.

e. Clique em **Clear display filter** para exibir toda a conversa do Wireshark.

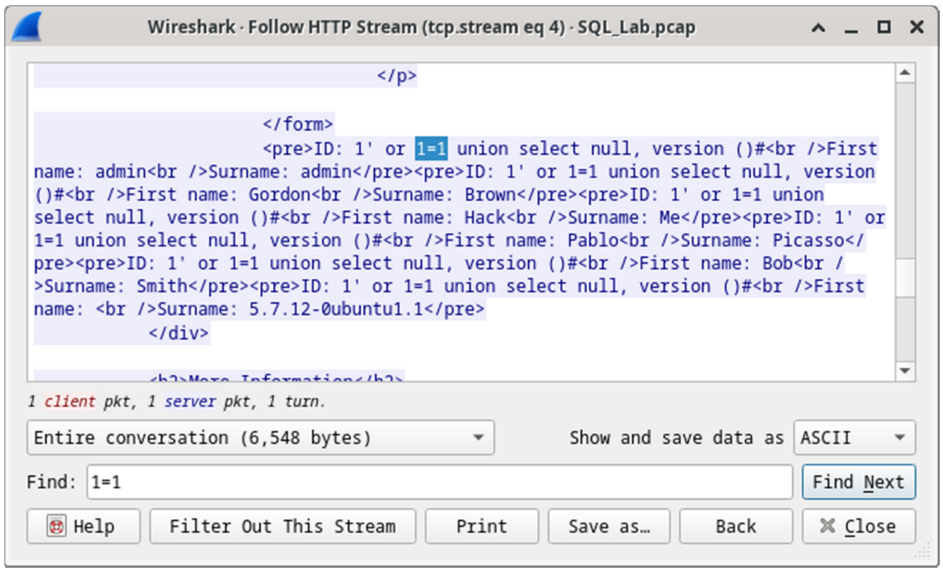
Parte 4: O SQL Injection Attack fornece informações do sistema.

O invasor continua e começa a segmentar informações mais específicas.

a. Na captura Wireshark, clique com o botão direito do mouse na linha 22 e selecione **Seguir** > **Fluxo HTTP**. Em vermelho, o tráfego de origem é mostrado e está enviando a solicitação GET para o host 10.0.2.15. Em azul, o dispositivo de destino está respondendo de volta à fonte.

b. No campo **Find** , entre **1=1**. Clique em **Find next**.

c. O invasor inseriu uma consulta (1' ou 1=1 união select null, version () #) em uma caixa de pesquisa UserID no destino 10.0.2.15 para localizar o identificador de versão. Observe como o identificador de versão está no final da saída logo antes do </pre>. fechando código HTML.



Qual é a versão?

d. Feche a janela Follow HTTP Stream.

e. Clique em **Clear display filter** para exibir toda a conversa do Wireshark.

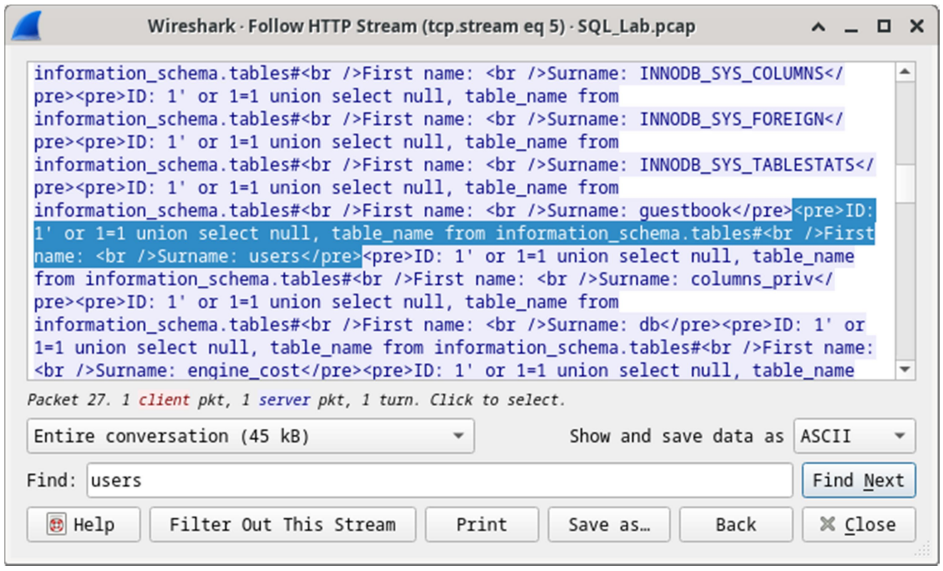
Parte 5: O Ataque de Injeção SQL e Informações da Tabela.

O invasor sabe que há um grande número de tabelas SQL que estão cheias de informações. O atacante tenta encontrá-los.

a. Na captura Wireshark, clique com o botão direito do mouse na linha 25 e selecione **Follow** > **HTTP stream**. A fonte é mostrada em vermelho. Ele enviou uma solicitação GET para o host 10.0.2.15. Em azul, o dispositivo de destino está respondendo de volta à fonte.

b. No campo **Find**, insira **users**. Clique em **Find next**.

c. O invasor inseriu uma consulta (1'ou 1=1 union select null, table_name from information_schema.tables #)em uma caixa de pesquisa de ID de usuário no destino 10.0.2.15 para visualizar todas as tabelas no banco de dados. Isso fornece uma enorme saída de muitas tabelas, como o invasor especificou “nulo” sem quaisquer especificações adicionais.



O que o comando modificado de (**1' OU 1=1 UNION SELECT null, column_name FROM INFORMAtTION_schema.columns WHERE table_name='users'**) faria para o invasor?

d. Feche a janela Follow HTTP Stream.

e. Clique em **Clear display filter** para exibir toda a conversa do Wireshark.

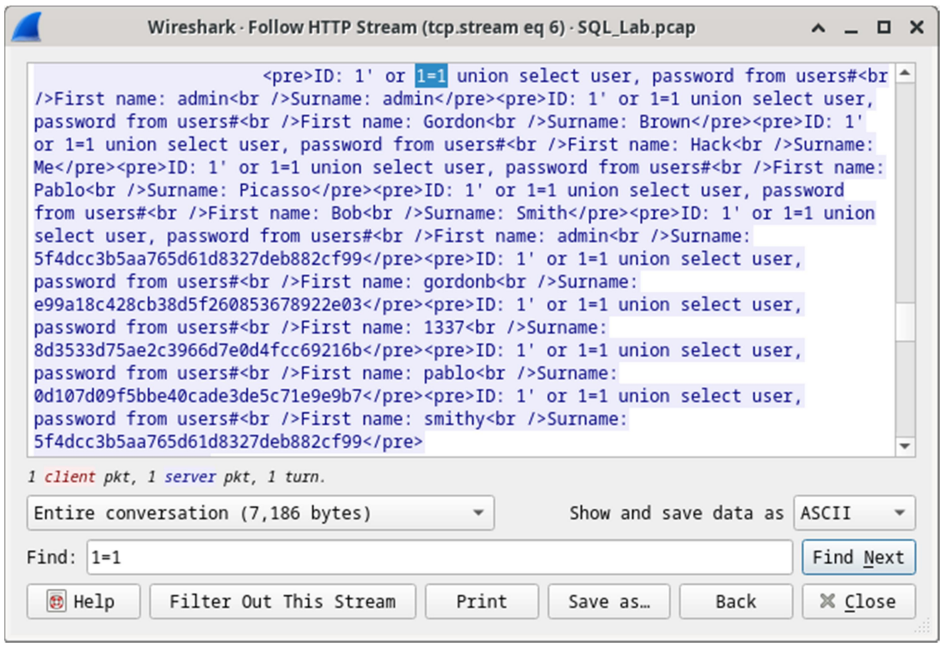
Parte 6: O ataque de injeção de SQL é concluído.

O ataque termina com o melhor prêmio de todos; hashes de senha.

a. Na captura Wireshark, clique com o botão direito do mouse na linha 28 e selecione **Follow** > **HTTP Stream**. A fonte é mostrada em vermelho. Ele enviou uma solicitação GET para o host 10.0.2.15. Em azul, o dispositivo de destino está respondendo de volta à fonte.

b. Clique em **Find** e digite **1=1**. Procure por esta entrada. Quando o texto estiver localizado, clique em **Cancel** na caixa de pesquisa Localizar texto.

O invasor inseriu uma consulta (1'ou 1=1 union select user, password from users#) em uma caixa de pesquisa UserID no destino 10.0.2.15 para obter nomes de usuário e hashes de senha!



Qual usuário tem o hash de senha de 8d3533d75ae2c3966d7e0d4fcc69216b?

c. Usando um site como https://crackstation.net/, copie o hash de senha no cracker de hash de senha e comece a crackear.

Qual é a senha de texto simples?

d. Feche a janela Follow HTTP Stream. Feche todas as janelas abertas.

Perguntas para reflexão

1. Qual é o risco de as plataformas usarem o langauge SQL?

2. Navegue na internet e faça uma pesquisa em “prevenir ataques de injeção SQL”. Quais são os dois métodos ou etapas que podem ser tomadas para evitar ataques de injeção SQL?