Automated Testing Framework: Experiences

How it works

For our automated testing framework, we are using Python as our scripting language. Our runAllTests.py script reads through all of the files in the testCases directory and adds all the values in each file to a dictionary that is then pushed onto a list of other test-case dictionary (so a list of dictionaries).

Using the file names, runAllTests.py then uses those names to run the tests themselves from the testCaseExecutables directory. The specific dictionary holding data for that test case is sent along to the text case executable on the command line. This is done by serializing the dictionary in runAllTests and decoding the serialization once it reaches the specific test case executable. This way the data sent to the test case can be dynamic based on the test case's needs. Each test writes to a results file that will eventually be used to show the results of all tests in your browser, once all tests are run.

How-To

To run the automated testing framework, the instructions are simple:

1. From the terminal, change to the TestAutomation directory
2. From TestAutomation, type the following command: "python ./scripts/runAllTests.py"
3. You will see several tests pop up on your browser, then you will see a results page for which tests passed and which tests failed.

5 / 25 Test Cases Used

Section	Description
Test Case #	1

Section	Description
Summary	Verify that the base64 encoding works through the EnDe interface
Prerequisites	Web browser can properly load EnDe suite (currently, not Chrome)
Procedure	1. User enters encoding text into the Encoding Text area
	2. User hovers over 'Base-N' option in left-hand pane
	3. From the pop-up menu the User selects the 'base64' option
Test Data	Encoding Text: `Euro`
Oracle	Decoded Text: `RXVybw==` (found using python's base64 algorithm, not EnDe's)

Section	Description
Test Case #	2
Summary	Verify that the hexidecimal conversion works through the EnDe interface
Prerequisites	Web browser can properly load EnDe suite (currently, not Chrome)
Procedure	1. User enters char (string) value into the Encoding Text area
	2. User hovers over 'Numbers' option in left-hand pane

Section	Description
	3. From the pop-up menu the User selects the 'Character to Hex' option
Test Data	Encoding Text: `Hex Test`
Oracle	Decoded Text: `4865782054657374`

Section	Description
Test Case #	3
Summary	Verify that integer to binary conversion works through the EnDe interface
Prerequisites	Web browser can properly load EnDe suite (currently, not Chrome)
Procedure	1. User enters integer into the Encoding Text area
	2. User hovers over 'Numbers' option in left-hand pane
	3. From the pop-up menu the User selects the 'Integer to Binary' option
Test Data	Encoding Text: `42`
Oracle	Decoded Text: `101010`

Section	Description
Test Case #	4
Summary	Verify that the ROT13 encoding works through the EnDe interface
Prerequisites	Web browser can properly load EnDe suite (currently, not Chrome)
Procedure	1. User enters encoding text into the Encoding Text area
	2. User hovers over 'Coding' option in left-hand pane
	3. From the pop-up menu the User selects the 'ROT13' option
Test Data	Encoding Text: `Testing`
Oracle	Decoded Text: `Grfgvat`

Section	Description
Test Case #	5
Summary	Verify EnDe's morse code encoding
Prerequisites	Web browser can properly load EnDe suite (currently, not Chrome)
Procedure	1. User enters encoding text into the Encoding Text area
	2. User hovers over 'Symbols' option in left-hand pane
	3. From the pop-up menu the User selects the 'Morse' option

Section	Description
Test Data	Encoding Text: `sos`
Oracle	Decoded Text: `. . .  ___  . . .`

Some More Testing - Summarized:

Here is our sample text used to test some of the encoding and decoding, as well as encryption and decryption:

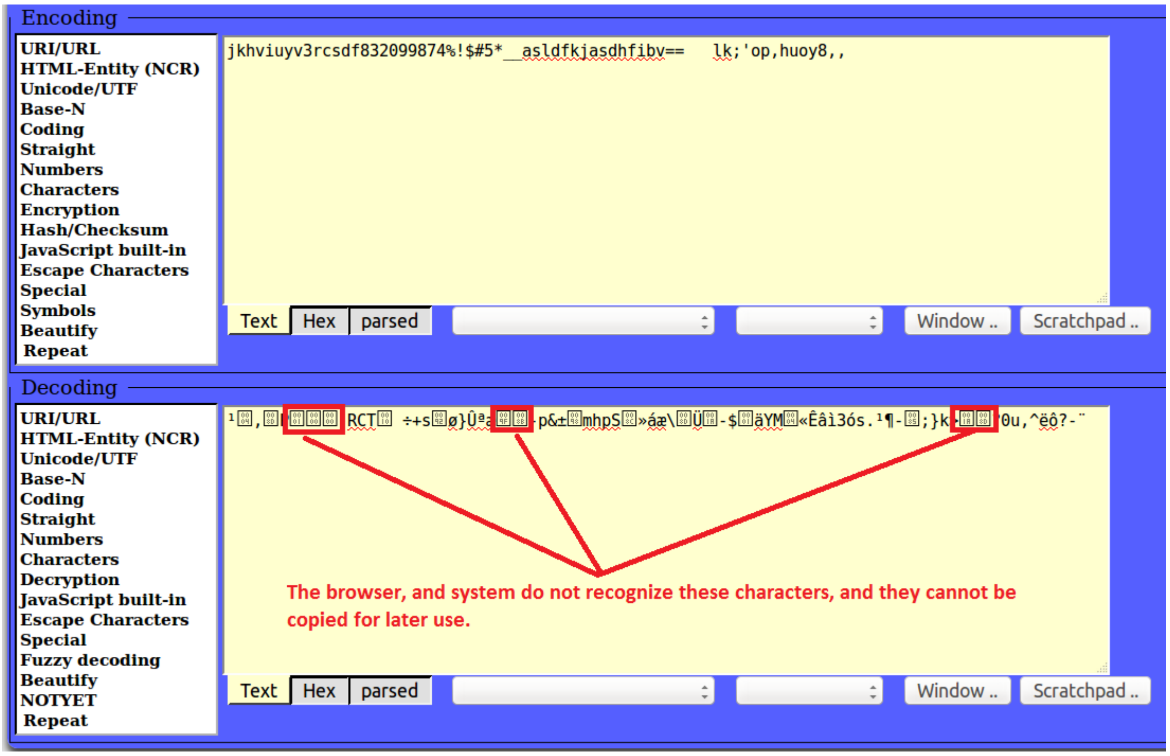
```
 jkhviuyv3rcsdf832099874%!$#5*__asldfkjasdhfibv==   lk;'op,huoy8,,
```

These Base(XX) encoding functions work fine as these are the outputs that return the same when decoded:

```
 Base64:
amtodml1eXYzcmNzZGY4MzIwOTk4NzQlISQjNSpfX2FzbGRma2phc2RoZmlidj09ICAgbGs7J29wLGh1b3k4L
Cw=
 Base85: rQ7pbxCcR@;SCZ1uR@4|9G+rLNB@pQcB@;pQidnWG8D+^uBD7hzuMB0
```

And so on...

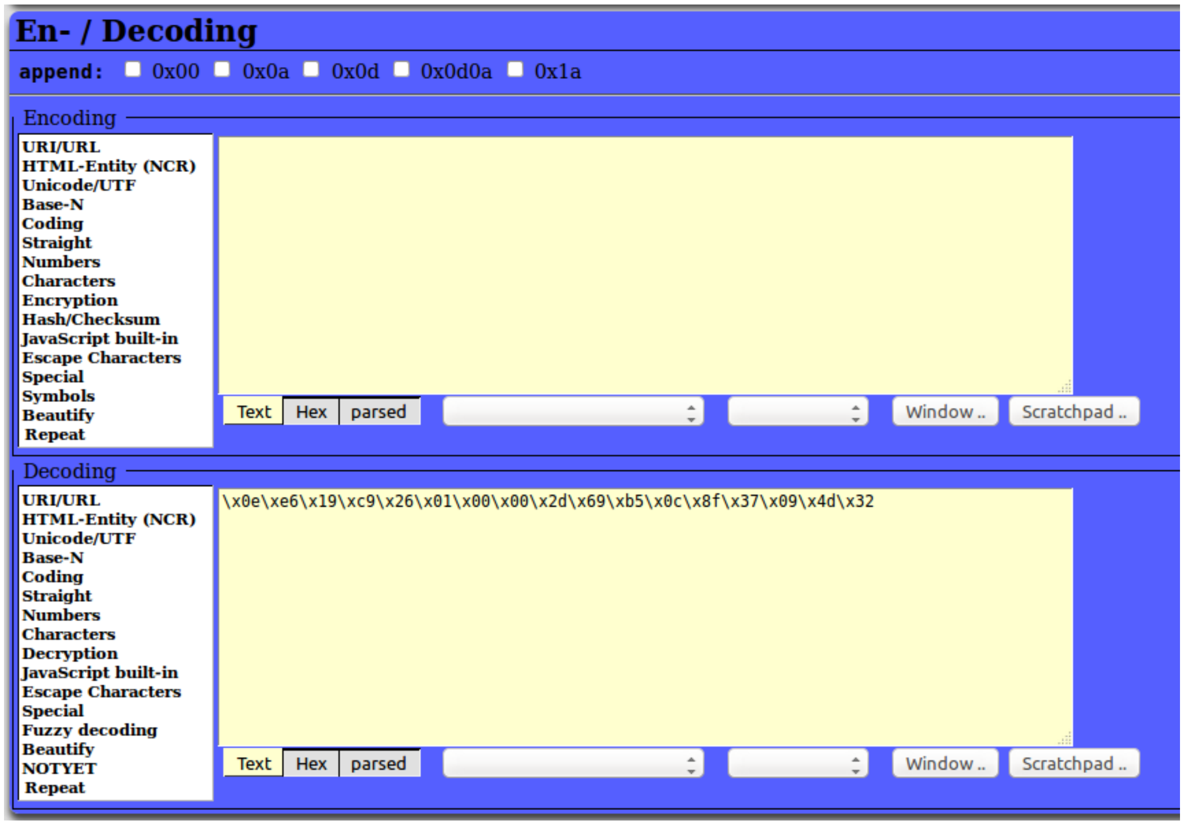
However, encryption and decryption tests are somewhat, difficult. The encryption functions sometimes returns characters not identified by either the browser, or the system running the tests, and cannot be placed back into the function correctly. The system records "¹,□P" as the encoded text when "copied", and outputs content in a similar fashion, as seen below:



```
jkhviuyv3rcsdf832099874%!$#5*__asldfkjasdhfibv==   lk;'op,huoy8,,
```

```
¹□,□□□□ RCT□ ÷+s□ø}Ûªa□□-p&±□mhpS□»áæ\□Ü□-$□äYM□«Êâì3ós.¹¶-□;}k-□□?0u,^ëô?-¨
```

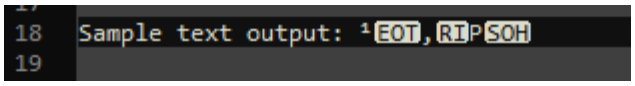
The browser, and system do not recognize these characters, and they cannot be copied for later use.



```
\x0e\xe6\x19\xc9\x26\x01\x00\x00\x2d\x69\xb5\x0c\x8f\x37\x09\x4d\x32
```

Errors like this occur for all except for BLOWFISH and BLOCK (TEA) ESCAPED encryption.

ENDE provides a ENDEtest.js, and a ENDEtest.txt file, however, manually trying the encryption yeilds no results. Either this is an issue with the browser and javascript IDE (netbeans), or there may be something wrong with the encryption methods used (ie: javascript/python/C/C# encryption methods) - It would be safe to assume the first would be the issue, that the characters requested for the text are simply not found on the host's system.



```
Sample text output: ¹EOT,RIPSOH
```

The following text provides no results when decrypting AES text using the same method:

```
_title      Encryption
#----------+--------------------------------------------------------
aes128      \xa3\x98\x17\xc9\x26\x01\x00\x00\x2d\x7c\x4d\x3b\xfe\x1d\xc2\x01\x07
aes192      \xd5\x53\x18\xc9\x26\x01\x00\x00\x2d\xd0\x9a\x62\x0f\xf2\x75\x90\xc0
aes256      \xda\xf1\x18\xc9\x26\x01\x00\x00\x2d\x7d\x6f\x93\xb7\x01\xde\xed\x7a
aes128r     \x14\x41\x19\xc9\x26\x01\x00\x00\x2d\x90\x64\x70\x6c\xfa\x19\xed\x4f
aes192r     \x23\x8a\x19\xc9\x26\x01\x00\x00\x2d\x65\xbe\x34\x94\xda\x41\x4e\x9c
aes256r     \x0e\xe6\x19\xc9\x26\x01\x00\x00\x2d\x69\xb5\x0c\x8f\x37\x09\x4d\x32
teaesc      !1!!227!!7!!130!!159!!218!!26!!240!
teacor      \x01\xe3\x07\x82\x9f\xda\x1a\xf0
tearaw      \x01\xe3\x07\x82\x9f\xda\x1a\xf0
```