2019 GSoC OWASP Proposal

Personal Profile

My name is Bocheng Zhang(Mars Zhang) and I am a sophomore student majoring in Computer Science at Hangzhou Dianzi University(HDU), China.

My Coding Skill Sets

Basically, I am well equipped with Python, C, C++ and Machine Learning knowledge due to my study and project experience. And I am also familiar with Windows and Linux system, especially with KALI and Centos.

My Project Experience

I mainly program on windows system and Linux system (KALI and centos). I am fluent in C python as the programming language for research and production code usage.

Since last year, I have been working as a researcher with an Information Security Research team at my university, an innovative group aiming at the security of the industrial control equipment. The project I mainly contribute to is to is a system which has four functional modules: on-line verification and scanning of industrial control systems; non-intrusive verification of industrial control systems; plug-in verification of industrial control systems; customized search and visualization of situational awareness. My role is implementing scan identification, as well as using HMM model and TF-IDF algorithm to our security identification system. This experience not only stimulates my enthusiasm in information security, but also helps develop my ability to update my own vulnerability information and the multi-process crawls the latest vulnerability information of the first-line security website. This link shows a demo of the initial implementations of our system.

I am committed to building and managing databases and successfully implementing scan identification. I also use NLP to implement real-time automatic updates to the database and to eliminate redundancy.

In the meantime, I work for HDU INPAINT Research Group which focuses on the research on Image Restoration Technology. Photographs are often unrecognizable due to late damage or unrelated factors on the spot. The goal of this group is to fix these photos with impurities. I tried a variety of methods introduced in cutting-edge conference papers such as Object Removal by Exemplar-Based Inpainting, combined with deep learning to fix photos.

My Open Source Experience

Open Source

I have always liked the form of open source. I think open source can draw more people's wisdom and inspiration so that the project can be more dynamic and innovative. I have used the nmap package before, and I have read the relevant code that is exposed on GitHub. BLT has attracted me so far, I also cloned it to my computer and tried to run it. These experiences have made me feel that open source is a very good form of the project, both for users and developers.

OWASP

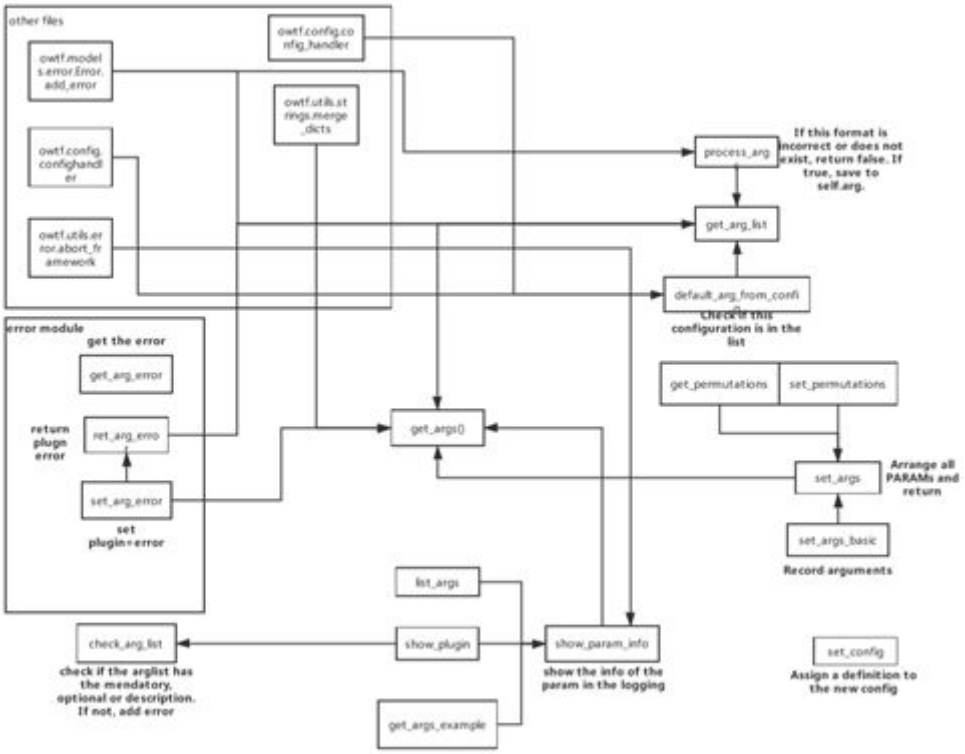
Fix Bug #975

- At the same time, I am also working on bugs for network scanner. Some progress on this bug: there is no get_all function in the target_manager class of owtf.managers.target in the process_plugin_list function, which cause the errors. According to the performance of the process_plugin_list function, the get_all function should be used to obtain the ID information of the target. After examining at the target_manager class, I replaced the get_all function with the get_all_target() function with the help of mohit and it worked fine.
- I also raised my own doubts about this and the inference of possible reasons: In the 'for http_ports in http' loop, the plugin_group value passed to the process_plugins_for_target_list function is "web". The function has one priority is to judge 'if plugin_group == "network" ', so no subsequent operations are performed.
- In the 'for loop' mentioned above, the passed target_list parameters are {"https://{}".format(target.split("//")[1])} and {target}. But the target does not be in the list format, so it may lead to the repetition of the following steps.
- With the help of Viyat, I think the reason why it doesn't work is: In the 'loop of the target', the 'lastwave' is always the value of 'wave[0]' after the second loop, which causes the parameters of the subsequent 'get_tcp_ports' function always is two equal values.

Other Issues

Function Bug

The following is a UML I created for modules related to param.py.



It seems ‘get_arg_list’ function does not work properly. When argument is optional, it also jumps out of the loop directly through return, which causes the error to be output and the loop to be terminated when the function calls and enters 'mandatory=true'. Function issue can be seen as belows.

```
if not self.init:
    self.init = True
    if not self.process_args():  # Process Passed arguments the first time only
        return self.ret_arg_error({}, plugin)  # Abort processing (invalid data)
args = {}
for arg_name in arg_list:
    if arg_name not in self.args:
        config_default_order = [
            "{0}_{1}_{2}".format(plugin["code"], plugin["type"], arg_name),
            "{0}_{1}".format(plugin["code"], arg_name),
            arg_name,
        ]
        default = self.default_arg_from_config(args, arg_name, config_default_order)
        if default or mandatory is False:
            # The Parameter has been defaulted, must skip loop to avoid assignment at the bottom or
            # argument is optional = ok to skip
            continue
```

```
    mandatory = self.get_arg_list(session, full_args_list["Mandatory"], plugin, True)
    optional = self.get_arg_list(session, full_args_list["Optional"], plugin, False)
```

I tried to make the following corrections:

```
'''def get_arg_list(self, session, arg_list, plugin, mandatory=True):
    if not self.init:
        self.init = True
        if not self.process_args():  # Process Passed arguments the first time only
            return self.ret_arg_error({}, plugin)  # Abort processing (invalid data)
    args = {}
    for arg_name in arg_list:
        if arg_name not in self.args:
            config_default_order = [
                "{0}_{1}_{2}".format(plugin["code"], plugin["type"], arg_name),
                "{0}_{1}".format(plugin["code"], arg_name),
                arg_name,
            ]
            default = self.default_arg_from_config(args, arg_name, config_default_order)
            if default or mandatory is False:
                # The Parameter has been defaulted, must skip loop to avoid assignment at the bottom or
                # argument is optional = ok to skip
                continue
                if mandatory:
                    Error.add_error(
                    session,
                    "USER ERROR: {!s} requires argument: '{!s}'".format(self.show_plugin(plugin), arg_name),
                    "user",
                    )
                return self.ret_arg_error({}, plugin)  # Abort processing (invalid data)
            args[arg_name] = self.args[arg_name]
    return args'''
```

Name Repeat Issue

From the code, it looks that the ‘args parameter of list_args’ function and the ‘full_args_list of get_args_example’ function are equal, but their titles are different. The following values and keys seem to have the same issue.

```
def list_args(self, args, mandatory=True):
    """List of available arguments

    :param args: args
    :type args: `dict`
    :param mandatory: True/false if mandatory to set
    :type mandatory: `bool`
    :return: None
    :rtype: None
    """
    logging.info("")  # Newline
    if mandatory:
        logging.info("mandatory parameters:")
    else:
        logging.info("Optional parameters:")
    for arg_name, arg_description in list(args.items()):
        if arg_description is None:
            arg_description = ""
        logging.info("- %s%s%s", arg_name, (30 - len(arg_name)) * "_", arg_description.replace("\n", "\n"))
```

Project Information

Project Name: OWASP DefectDojo_Unittests and Python3 Completion

Project Description

DefectDojo is a tool that helps information security workers manage product security vulnerabilities. This can make the related product vulnerabilities more orderly and clear, thus greatly improving the efficiency of the staff. The workflow of this program is that firstly creating a product type for a product that needs to process. Then adding the method to test and create test environment for the product. Last, using tests and inputting to get specific test information and generate related reports.

Why I am interested in OWASP DefectDojo

The project I want to apply for is DefectDojo because it is very comprehensive and it can be used as a very efficient and practical information security vulnerability management tool.

I did some work related to security protection with another application before, so when I first used DefectDojo, I was impressed by DefectDojo's functions. For example, DefeciDojo offers specialized procedures to help security workers manage and collect security information with consistent and clear standards, so users can save time to effectively exploit vulnerabilities, investigate vulnerability handling, and so on.

What's more, I have experience of using python3 and python2 to write code, so I am more confident that I can change code into the form of python3 based on project requirement.

After I studied the relevant code about plugin owing to the DefectDojo, I found some issues that I want to improve. For example, the existing test file to be very scarce. Most of the files in the tools folder and the dojo folder do not have their own reliable test files. So I want to design their unittest files for these files to make them more operational. At the same time, there are quite a few files that still have Python2 written forms such as URLparse and so on. Therefore, I want to unify the format of these codes into a python3 form. I hope to help improve these issues.

Deliverables

Complete the Unittest Files

- Write a test file that can run successfully for the following files

- Tools: the parser.py file in appspider, arachni, bandit, burp, checkmarx, contrast, generic, gosec, nessus, nexpose, nikto, nmap, openvas_csv, qualys, qualyswebapp, retirejs, skf, snyk, sslabs, trufflehog, trustwave, veracode, zap folders. Among them, I wrote the unittest for parser.py in the nsp folder as an example. The links are as follows: test_nsp_parser.py

- Dojo: the view.py file in banchmark, cred, development_e, engagement, finding, home, jira_l, metrics, notifications, object, product, produxt_t, reports, rules, scan, search, system_s, test, test_t, tool_c, tool_p, tool_t folders.

Python3 Completion

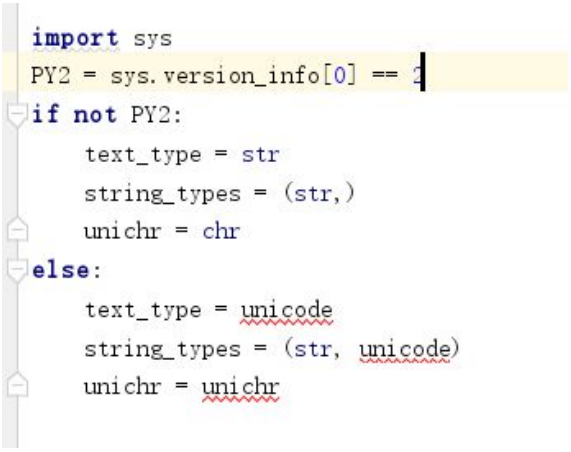
- Change the code inside the existing Python2 form to the available Python3 form of code.
  - As shown, the print in the code is obviously still in the form of python2, which is changed to the form of print().

```
print issue.fields.issuelinks[0]

print "Jira Issue: " + str(issue)
print "Resolution: " + str(issue.fields.resolution)

if issue.fields.resolution is not None \
    from lxml import etree
except ImportError:
    print "Missing lxml library. Please install using PIP.
https://pypi.python.org/pypi/lxml/3.4.2"
    exit()

try:
    import html2text
except ImportError:
    print "Missing html2text library. Please install using PIP.
https://pypi.python.org/pypi/html2text/2015.2.18"
```

  - The string type of python3 has also changed. The Unicode string in Python 2 is a normal string in Python 3, because in Python 3 the string is always in Unicode form.
  -
  - As shown below, the key returned by the dictionary in Python3 is not directly in the form of a list, and list() should be added to convert it into a list.

```
for idx, widget in enumerate(widgets):
    if widget.keys()[0] == 'page-break':
        selected_widgets[widget.keys()[0] + '-' + str(idx)] = PageBreak()
    if widget.keys()[0] == 'endpoint-list':
```

  - Design some specialized modules to initially test whether the target code is in python3 format, as follows:



```
import sys
PY2 = sys.version_info[0] == 2
if not PY2:
    text_type = str
    string_types = (str,)
    unichr = chr
else:
    text_type = unicode
    string_types = (str, unicode)
    unichr = unichr
```

Update My Work

- Keep my work updated to my GitHub so that subsequent contributors can clearly understand my workflow.
- Communitate with mentors and group to get feedback.

Timeline

Community Bonding Period (May 06-27, 2019)

- Week 1 (May 06-13,2019): Further discussion about proposal with community.
- Week 2 (May 13-20,2019): Design unittest for the parser.py file in appspider, arachni, bandit, burp, checkmarx, contrast, generic, gosec folders.
- Week 3 (May 20- 27,2019): Change the code of the parser.py file in appspider, arachni, bandit, burp, checkmarx, contrast, generic, gosec folders into Python3 form.

First Evaluation Period Deliverables (May 27 -Jun 29,2019)

- Week 4 (May 27-June 03,2019): Design unittest for the parser.py file in nessus, nexpose, nikto, nmap, openvas_csv, qualys, qualyswebapp, retirejs, skf folders.
- Week 5 (June 03-June 10,2019): Change the code of the parser.py file in nessus, nexpose, nikto, nmap, openvas_csv, qualys, qualyswebapp, retirejs, skf folders into Python3 form.
- Week 6 (June 10-June 17,2019): Design unittest for the parser.py file in snyk, sslabs, trufflehog, trustwave, veracode, zap folders and the view.py file in banchmark, cred, development_e folders.
- Week 7 (June 17-June 24,2019): Change the code of the parser.py file in snyk, sslabs, trufflehog, trustwave, veracode, zap folders and the view.py file in banchmark, cred, development_e folders into Python3 form.
- First Evaluation time: Research and communicate with mentors About previous progress(June 24 -29,2019)

Second Evaluation Period Deliverables (Jul 01- 26,2019)

- Week 8 (July 01-July 08,2019): Design unittest for the view.py file in engagement, finding, home, jira_l, metrics, notifications, object, product folders.
- Week 9 (July 08-July 15,2019): Change the code of the view.py file in engagement, finding, home, jira_l, metrics, notifications, object, product folders into Python3 form.
- Week 10 (July 15-July 22,2019): Design unittest for the view.py file in produxt_t, reports, rules, scan, search, system_s folders.
- Second Evaluation time(July 22 -26,2019)**:** Research and communicate with mentors About previous progress

Final Evaluation Period Deliverables (Jul 26 -Aug27,2019)

- Week 11 (July 26-Aug 2,2019): Change the code of the view.py file in produxt_t, reports, rules, scan, search, system_s folders into Python3 form.
- Week 12 (Aug 2-Aug 9,2019): Design unittest for the view.py file in test, test_t, tool_c, tool_p, tool_t folders.
- Week 13 (Aug 9-Aug 16,2019): Change the code of the view.py file in test, test_t, tool_c, tool_p, tool_t folders into Python3 form.
- Week 12 (Aug 16-Aug 27,2019): Submit

Why you should choose me?

My Ability to Contribute

I have full background development experience and Python experience in college. With the experience of implementing a complete and effective background database automatic collection and update function and related APIs for my research team, I have great confidence in the implementation of DefectDojo's high-performance new unittests. After supplementing with more in-depth learning, I can make some better fixes for DefectDojo's Python3 environment.

Self-learner with High Motivition

I am good at self-learning and seeking out solutions whatever the case is. For example, since I started my GSoC journey, I have been reading organization documents, watching tutorial videos on Youtube and talking with community. Now I am very familiar with the codebase I am working on.

Teamwork With Community

When I was working on my proposal, I enjoy communication and coordination with community and mentors, which helps me get used to community rules and teamwork.I

appreciate everyone's help and I hope I will successfully finish my project to as return to our community.

My Summer Plans

For this summer, I do not have any other plans except GSoC so far. If I am accepted, I will be able to work on GSoC project for 30-40 hours per week. I will always keep in touch with the DefectDojo organization and mentor and actively complete my work.

Reference

- Making Vulnerability Management Less Painful
- Unit Testing in Python
- Python 2 to 3