Before using the system, some preparation works must be done.

Firstly, all servers are required to be deployed by using Amazon Web Service (AWS) EC2 instances.

Secondly, some files (scripts) should be put on servers. // TODO

Thirdly, all servers deployed should install Java. They can install Java by using the following commands:

- sudo add-apt-repository ppa:webupd8team/java

- sudo apt-get update

- sudo apt-get install oracle-java8-installer

General way of using:

The system supports a Domain Specific Language(DSL), Stitch, which is developed by Shang Wen Cheng, David Garlan, Bradley Schmerl and so on.

To start using the system, users should input some Stitch scripts. Scripts can be written from 2 sides, attacker and defender, to do the simulation and experiment. Each side of Stitch scripts should contain 1 strategies script and 1 tactics script. The attacker’s tactics script should be named as “attackerTactics.s” and the defender’s tactics script should be named as “defenderTactics.s”.

Researchers can play roles of attacker and defender concurrently, which means you can input 2 attacker scripts, run them, and input 2 defender scripts, run them at the same time.

Operations provided:

You can only use operations that we support in Stitch scripts. Attacker can use the operations including “sendPhishingEmail”, “A.downloadLogFile”, “A.decodeLogFile”, “A.loginWeb”, “A.injectShell”, “A.crackPasswd”, “A.storePasswd”, “A.firmware”, “A.transaction”, “A.deleteLogFile”, “A.deleteWebCredential”, “F.deleteLogFile”.

Defender can use the operations including “filterPhisihngEmail”, “W.resetPassword”, “increaseWebThreshold”.

You can also add whatever operations you want, just remember to write statements about those operations in tactic’s actions. After using the system’s Antlr visitors, these Stitch statements will be finally directed to execOperations(str) function in Model file. The “str” here is a parameter of String data type, which corresponds to a kind of nodes “MethodCall” in the generated Abstract Syntax Tree (AST). Therefore, specific operations should be implemented or called in this execOperations(str) function.

Boolean conditions provided:

Conditions of the system are represented by some Boolean values. Accepted Boolean conditions include “hasWebCredential”, “hasLogFile”, “hasCardCredential”, “webPasswdExpired”. Other data types of conditions like “W.time” and “W.threshold” are stored and used to set the Boolean condition. Some files in the system also have values to represent its status. For example, email has a Boolean value “isPhishingEmail” to indicate whether it is a phishing email or not.

We did not write most of our conditions as “define Boolean xxx = exists …” like other Stitch examples because of 2 reasons. First, this system is not as complicated as Rainbow and we do not have so many servers to monitor, which means it is unnecessary to write them all in that complex way. Second, we want to update these conditions in real time instead of setting all of them in the beginning of strategy script.

Also, you can add other conditions if you want. If they are written in the same way in scripts as our system, their related updated functions should be implemented in the Probe or in execHook(str) function in the Model.

Model description:

We have provided an interface of Model and implemented 1 kind of model named “TargetModel”. Our model uses some data structures like HashMap and HashSet to store servers’ IP addresses, tactics visited, hooks, probes, time thresholds, and so on. Some getter() and setter() methods are also implemented in our model. We also implemented hooks and operations in execHook(str) and execOperations(str).

Each system of a specific number of servers should have only 1 related model to store its status and methods. If the system is changed, either category of servers or number of servers has been changed. A new model needs to be implemented. You can add any number of models if you want to change the system into any structure.

Future works:

Although we have implemented our own version of visitors to visit the AST generated from Stitch scripts, our visitor is not very complete due to the limited time. We didn’t implement the TypeChecker part and didn’t do a lot about ErrorHandler, which are both implemented in Rainbow system. We can add those parts if we have more time. If future users want to use our system as a starting point, it will not be too hard to add TypeChecker and ErrorHandler. And possibly they will not influence other parts of the system.

Since we only have 1 server for each category, some logics and advantages of Stitch language cannot be used well. For example, statements like “exists …”, “set …”, “for (…)”, “forall …” seldom occur in our scripts. This part can be improved for a more complicated system or systems with more servers.

The defense in our simulation is still weak and incomplete, more defender’s strategies and tactics can be explored in the future. How to better design defender’s strategies and related trigger conditions; how to monitor more system status from a defender’s side; and how to integrate defense with Game Theory or Machine Learning techniques are all directions that can be focused on.