The Honeynet Project

Google Summer of Code 2016 Project Ideas

Thu, 02/18/2016 - 21:12 — felix.leder

Twitter Facebook LinkedIn

This page contains a list of potential project ideas that we are keen to develop during GSoC 2016. We also have additional project ideas that are currently undergoing internal review. These will be added here, once project deliverables and available mentors have been confirmed.
During the previous years of GSoC, the Honeynet Project's students have created a wide range of very successful open source security projects, many of which have gone on to become the industry standard open source tools in their respective fields. Examples for these include:

- Cuckoo Sandbox (2010+)
- Thug Client Honeypot (2012+)
- MITMProxy (2012+)
- DroidBox Android Sandbox (2011+)
- ConPot ICS/SCADA Honeypot
- Glastopf Web Application Honeypot (2009+)
- Dionaea (2009+)

If you can't find something to immediately interest you, please tave a look at GSoC 2009, GSoC 2010, GSoC 2011, GSoC 2012, GSoC 2013, GSoC 2014 and GSoC 2015 project ideas pages for other inspiration. Or you might like to work on one of our existing tools, rather than working on something new.
We are also always interested in hearing any ideas for additional relevant computer security and honeynet-related R&D projects (although remember that to qualify for receiving GSoC funding from Google your project deliverables need to fit in to GSoC's 3-month project timescales!). If you have a suitable and interesting project, we will always try and find the right resources to mentor it and support you.
Please note - even if you aren't an eligible GSoC student, we are also always looking for general volunteers who are enthusiastic and interested in getting involved in honeynet R&D.
Each sponsored GSoC 2016 project will have one or more mentors available to provide a guaranteed contact point to students, plus one or more technical advisors to help applicants with the technical direction and delivery of the project (often the original author of a tool or its current maintainer, and usually someone recognised as an international expert in their particular field). Our Google Summer of Code organisational administrators will also be available to all sponsored GSoC students for general advice and logistical support. We'll also provide supporting hosted svn/trac/git/redmine/mailman/IRC/etc project infrastructure, if required.
For all questions about the Honeynet Project, the GSoC program or our projects, please contact us on **#gsoc-honeynet** on **irc.freenode.net**, subscribe to our public mailing list for people interested in GSoC at https://public.honeynet.org/mailman/listinfo/gsoc or email us directly at project@honeynet.org.
To learn more about the Google Summer of Code event, see the the GSoC 2016 Website.

Student Application Tips

We are looking for bright minds that have a long-term interest to do research in the security community and especially the Honeynet Project. As we live and breathe open source, we are looking for students with great coding skillz. For some hints and tips for you application and more details about what we are looking for check out: our Application Tips page

R&D Focus Areas

This year our honeynet R&D focus for GSoC will primarily directed into a number of priority areas, which are:

- Topical malware (e.g. attacks against ICS/SCADA systems and mobile platforms such as Android, etc)
- Automated malware analysis (sandboxes)
- Distributed data collection, analysis and visualisation (honeypots and honeypot data)

GSoC 2016 Project Ideas Overview

- Project 1:**CuckooML** - Clustering malware analysis reports to find families by behavior
- Project 2:**DNP3 protocol support** - Add a real life protocol to the ICS/SCADA honeypot ConPot
- Project 3:**AWL/STL/PLC simulator** - Simulate a PCL to make the ICS/SCADA honeypot ConPot look more real
- Project 4:**DroidBOT** - An artificial user to interact with malware in an Android Sandbox
- Project 5:**Online Android Sandbox** - Web interface to simplify use of Android malware analysis
- Project 6:**Mitmproxy kick-ass web UI** - Fully featured UI for the SSL capable MITMProxy
- Project 7:**Mitmproxy awesomeness** - Take the SSL interception MITMProxy to the next level
- Project 8:**Rumāl** - Cooperative analysis UI for web threats
- Project 9:**YAPDNS** - Collect passive DNS data from non-DNS sources
- Project 10:**PCAPOptikon** - Analyze context information in attacks by combining Surricata and Pcaps
- Project 11:**Vulnerability emulation for SNARE** - Inject emulated vulnerabilities into known web sites
- Project 12:**Honeypots as a Virtualized network functions** - Take honeypot deployments to the next level

(more project ideas and mentors to follow, once internal review is complete)

GSoC 2016 Project Ideas

Automated Malware Analysis

Project Name: Project 1 - CuckooML
Mentor: Hugo Gascon (ES)
Backup mentor: Jurriaan Bremer (NL)
Skills required: Python, familiarity with Scikit-Learn and machine learning concepts, knowledge of Theano and Keras is also welcome
Project type: New technology in existing tool.
Project goal: Implement a new machine learning module in Cuckoo to perform clustering, anomaly detection and classification of existing and new behavioral analyses.
Description:
Cuckoo Sandbox (developed during GSoC 2010-2015 with The Honeynet Project [1]) has evolved to become the de-facto open-source standard for malware analysis systems. It contains capabilities for analyzing in malware in various Windows, Android[2] and Apple [3] environments, has a clean architecture and easy-to UI. It is used by many open source and commercial sandboxing efforts, including Google's own VirusTotal infrastructure.

The goal of this project is to develop a module for machine learning in Cuckoo using Scikit-Learn [4] that hould be able to cluster all reports according to similar behaviors. Given a class, the module will be able to find the most representative element (prototype) of each class. Once that a clustering exists and a new sample is analyzed, the new report can be assigned to one of the clusters and compared with similar samples. The module should also be able to perform anomaly detection, so alternatively, if no similar behavior is observed, a new cluster should be created. It should be possible to choose among several methods to do this. For example, the distance to the clusters could be measured or an SVM could be trained on existing data using the cluster labels. After the functionality based on stored analysis data from Cuckoo Sandbox is implemented, the module will be integrated into Cuckoo for command line and web-based interaction.
IRC Chat: #cuckooml @ freenode
[1] http://www.cuckoosandbox.org/
[2] https://github.com/pjlantz/droidbox/
[3] https://honeynet.org/gsoc2015/slot1
[4] http://scikit-learn.org/stable/

Conpot Projects

Conpot is a low interactive server side Industrial Control Systems honeypot designed to be easy to deploy, modify and extend. By providing a range of common industrial control protocols we created the basics to build your own system, capable to emulate complex infrastructures to convince an adversary that he just found a huge industrial complex. To improve the deceptive capabilities, we also provided the possibility to server a custom human machine interface to increase the honeypots attack surface. The response times of the services can be artificially delayed to mimic the behaviour of a system under constant load. Because we are providing complete stacks of the protocols, Conpot can be accessed with productive HMI's or extended with real hardware.
Project page: http://conpot.org
Code repository: https://github.com/glastopf/conpot
We are on #conpot@chat.freenode.net

Project Name: Project 2 - Conpot #1: Improve DNP3 protocol support used for communication between SCADA master stations and RTUs and IEDs.
Mentor: Lukas Rist (DE)
Backup mentor: Johnny Vestergaard (DK)
Skills required: Python, basic C++, dissecting network traffic
Project type: Improve existing tool
Project goal: Improve Conpot's current very minimal support for the DNP3 protocol. Goal is to provide a server capable of basic DNP3 communication.
Description:
Conpot provides a variety of common protocols: Modbus, S7Comm, SNMP, HTTP and Kamstrup. We are always working on getting additional protocols supported. This is a rather complicated task as many protocols don't have an open source implementation, documentation is rather complex or simply not available. One of the protocols we are interested in is DNP3 (Distributed Network Protocol) which is similar to IEC 60870-5 and often used for communication between control centers, RTUs (Remote Terminal Units) and IEDs (Intelligent Electronic Devices). Conpot has a feature which we call the Proxy Module. This allows us to proxy incoming requests through Conpot to a service and back to the client. When we implement a new protocol in Conpot, we set up an instance with this proxy module and tunnel all requests from the client to e.g. a real device or a service with that protocol running on another host. Then, piece by piece, we are decoding the message in Conpot while it passes through so we get insight into the intention of the request. Right now we have a very basic decoder for the DNP3 protocol which we would like to extend.
A student would get insights into information security, honeypot development, industrial protocols, dissection of network traffic and working in an open source project.

Project Name: Project 3 - Conpot #2: Add support for a AWL/STL/PLC simulator
Mentor: Lukas Rist (DE)
Backup mentor: Johnny Vestergaard (DK)
Skills required: Python, programming PLCs (AWL, STL), network programming
Project type: Improve existing tool
Project goal: Add support for a AWL/STL/PLC simulator. Goals is to run AWL/STL programs in Conpot and allow communication with the PLC emulator.
Description:
Conpot supports the protocols a common PLC is providing but not the functionality of a PLC. This means besides some randomized values and linear incrementing values like uptime the data in the honeypot is static. In order to appear more realistic and handle input values properly, we would like to support a PLC simulator. A good candidate is Awlsim (http://bues.ch/cms/hacking/awlsim.html): Awlsim is a free Step 7 compatible AWL/STL Soft-PLC written in Python. Awlsim provides an interface for virtual hardware connection modules (currently available are PROFIBUS-DP and LINUX-CNC). This interface could be used to connect Awlsim to Conpot.

A student would get insights into information security, honeypot deployment, PLC programming, network communication, PLC simulation and working in an open source project.

Droidbox Projects

Droidbox [1] is _the_ open source sandbox for Android app analysis. It has been developed during the previous Google Summer of Code 2012 by Patrik Lantz and has continued to evolved ever since. Several academic, open source, and even commercial projects are based on it [2-4]. Since Android is evolving, we also want to evolve Droidbox to keep up with new technologies (such as ART), make easier to use, and to provide even better integration with other sandboxing frameworks (such as the leading open source Cuckoo sandbox system, which as also developed during previous GSoCs).
[1] Droidbox https://github.com/pjlantz/droidbox
[2] Sandroid http://sanddroid.xjtu.edu.cn/
[3] Mobile Sandbox http://mobile-sandbox.com
[4] Andrubis http://anubis.iseclab.org/
This summer we have a number of Android/Droidbox projects we would like to see students work on, including:
Project Name: Project 4 - DroidBOT
Mentor: Hugo González (MX)
Backup mentor: Hanno Lemoine (DE)
Skills required: Python, Android Development
Project type: Improve existing tool
Project goal: Improve DroidBOT from last GSoC in order to get a better testing coverage of dynamic Android analyse systems.
Description:
A lot of Android malware relies on social engineering in order to infect devices. Since user interaction is required for installation, a large amount of Android malware verifies that a real user is present before starting its malicious actions (e.g. clicking a button). Similarly, some malware requires specific stimuli to verify it is running on a real phone (e.g. changing GPS coordinates). Other malware will check if it is running in an analysis environment by checking if there are at least 15 contacts on the phone.

Last year DroidBOT was build in GSoC [1,2]. It already has a lot of functionality and tests, but there are also some limitations and mission features.
The goal of this project is to provide the most realistically looking environment for malware in order to trigger all of the malicious actions.

One subgoal is to populate existing images in a dynamic way such that each analysis looks like a different phone (e.g. different contacts in address book). In addition, certain stimuli should be created such that they trigger required actions in the malicious app. Last but not least, the project includes to add a fake user that behaves as human as possible.

[1] http://lynnlyc.github.io/droidbot/
[2] https://github.com/lynnlyc/droidbot
[3] Morpheus: Automatically Generating Heuristics to Detect Android Emulators
  http://yimingjing.com/papers/acsac2014_morpheus_slides.pdf
[4] Evading Android Runtime Analysis via Sandbox Detection
  http://users.ece.cmu.edu/~tvidas/papers/ASIACCS14.pdf
[5] http://www.nemesys-project.eu/nemesys/files/document/resources/Infrastructure_for_detecting_Android_malware.pdf
[6] http://www.cc.gatech.edu/~naik/dynodroid.html

Project Name: Project 5 - Online Android Sandbox
Mentor: Hanno Lemoine (DE)
Backup mentor: Hugo González (MX)
Skills required: Python, Android, Cuckoosandbox, virtualization technologies, web tech (HTML, JavaScript) and SQL.
Project type: Improve existing tool
Project goal: Integrate DroidBOT into the Android part of cuckoosandbox (2.0) and setup a online service.
Description:
Droidbox [1] is used in a range of other projects. Even though these projects collect a lot of data, only few of them are available and the data is not always shared. We want to base the online sandbox on the existing code of the cuckoosandbox [2] like you can see on malw.com [3]. The Android Support in cuckoosandbox is brand new in Version 2.0 and docu and features are waiting to being written. One big feature would be to integrate DroidBOT [4] from the last GSoC in order to improve the code coverage of the dynamic analysis.

The online Android sandbox, which will can be hosted in the Honeynet Project's or any other cloud, could be improved as an extensible platform to allow for the inclusion of other tools and techniques, like Androguard [5] for static analysis and Droidbox for alternative dynamic analysis[1].

If successful, this GSoC project will guarantee constant testing and feedback for the cuckoodroid, DroidBOT and Androguard codebases, which will sure lead to further and faster improvements. Moreover, it will provide a constant stream of suspicious samples and a platform to test experimental techniques developed within the Honeynet Project.

[1] https://github.com/pjlantz/droidbox/
[2] https://cuckoosandbox.org/
[3] https://malwr.com/
[4] http://lynnlyc.github.io/droidbot/
[5] https://github.com/androguard/androguard/

MITMPROXY Projects

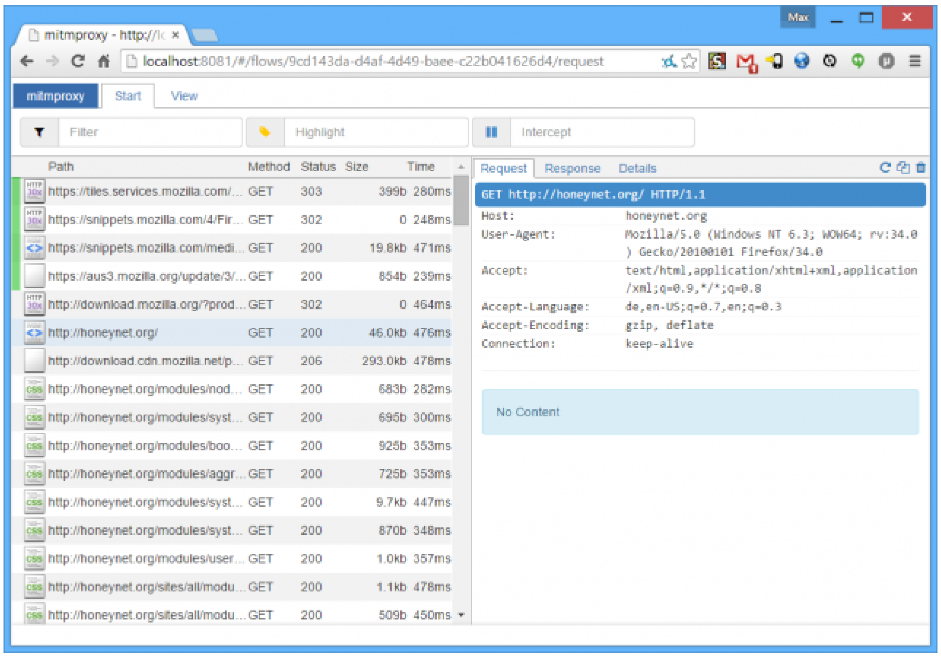
mitmproxy is a man-in-the-middle HTTPS proxy. It is an interactive console program written in Python that allows HTTP network traffic flows to be inspected and edited on the fly. mitmproxy has >100.000 downloads/year, >4000 stars on GitHub and more than 100 contributors - you're going to work on a project with a large community. :-)

Getting started for GSoC: https://github.com/mitmproxy/mitmproxy/issues/934
Project page: https://mitmproxy.org/
Code repository: https://github.com/mitmproxy/mitmproxy
Slack Chat: #gsoc @ http://slack.mitmproxy.org/

Project Name: Project 6 - Create a kick-ass web user-interface for mitmproxy!
Mentor: Maximilian Hils (DE)
Backup mentor: Aldo Cortesi (NZ)
Skills required:

- HTML5/JS (strong, React.js would be a plus)
- Python (familiar)
- HTTP (familiar)

Project type: Improve existing tool
Project goal: Spend the summer creating a modern web interface for mitmproxy!
Description:



Path	Method	Status	Size	Time
https://tiles.services.mozilla.com/...	GET	303	399b	280ms
https://snippets.mozilla.com/4/Fir...	GET	302	0	248ms
https://snippets.mozilla.com/medi...	GET	200	19.8kb	471ms
https://aus3.mozilla.org/update/3/...	GET	200	854b	239ms
http://download.mozilla.org/?prod...	GET	302	0	464ms
http://honeynet.org/	GET	200	46.0kb	476ms
http://download.cdn.mozilla.net/p...	GET	206	293.0kb	478ms
http://honeynet.org/modules/nod...	GET	200	683b	282ms
http://honeynet.org/modules/syst...	GET	200	695b	300ms
http://honeynet.org/modules/boo...	GET	200	925b	353ms
http://honeynet.org/modules/aggr...	GET	200	725b	353ms
http://honeynet.org/modules/syst...	GET	200	9.7kb	447ms
http://honeynet.org/modules/syst...	GET	200	870b	348ms
http://honeynet.org/modules/user...	GET	200	1.0kb	357ms
http://honeynet.org/sites/all/modu...	GET	200	1.1kb	478ms
http://honeynet.org/sites/all/modu...	GET	200	509b	450ms

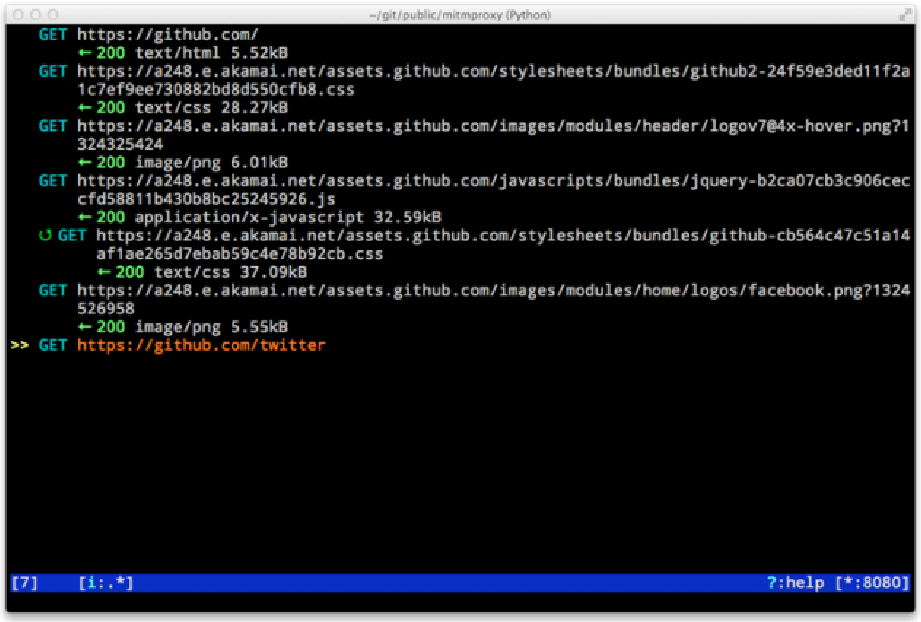
```
GET http://honeynet.org/ HTTP/1.1
Host:               honeynet.org
User-Agent:         Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0
                    ) Gecko/20100101 Firefox/34.0
Accept:             text/html,application/xhtml+xml,application
                    /xml;q=0.9,*/*;q=0.8
Accept-Language:    de,en-US;q=0.7,en;q=0.3
Accept-Encoding:    gzip, deflate
Connection:         keep-alive
```

We started working on our web front-end "mitmweb", which will finally bring a graphical user interface and Windows support to mitmproxy. Our long-term goal is to achieve feature-parity between the web-interface and the console application on most parts. The goal of this project is to add new features to the web interface so that we can ship it at the end of the summer. For example, one part of your project would be to implement and editing component for HTTP requests so that users can modify messages on-the-fly before they reach the server. While we have a good idea of some other features that need to be implemented, the first task for your application is to try out mitmweb and make a rough list of features you think are missing. We'll then mix that with our ideas and expectations and create a great project plan for the summer!
We're using a modern web app technology stack (React.js (Flux), Bootstrap, Gulp, ...), so you can work with the latest technologies and focus on good code rather than IE support. :-)

Project Name: Project 7 - Make mitmproxy's core great again!
Mentor: Thomas Kriechbaumer (DE)
Backup mentor: Maximilian Hils (DE)
Skills required:

- Python (strong)
- HTTP & TCP/IP (familiar)

Project type: Improve existing tool
Project goal: Spend the summer adding highly-requested features to mitmproxy's core!
Description:



```
    GET https://github.com/
        ← 200 text/html 5.52kB
    GET https://a248.e.akamai.net/assets.github.com/stylesheets/bundles/github2-24f59e3ded11f2a
        1c7ef9ee730882bd8d550cfb8.css
        ← 200 text/css 28.27kB
    GET https://a248.e.akamai.net/assets.github.com/images/modules/header/logov7@4x-hover.png?1
        324325424
        ← 200 image/png 6.01kB
    GET https://a248.e.akamai.net/assets.github.com/javascripts/bundles/jquery-b2ca07cb3c906cec
        cfd58811b430b8bc25245926.js
        ← 200 application/x-javascript 32.59kB
    ↻ GET https://a248.e.akamai.net/assets.github.com/stylesheets/bundles/github-cb564c47c51a14
          af1ae265d7ebab59c4e78b92cb.css
          ← 200 text/css 37.09kB
    GET https://a248.e.akamai.net/assets.github.com/images/modules/home/logos/facebook.png?1324
        526958
        ← 200 image/png 5.55kB
>>  GET https://github.com/twitter

[7]    [i:.*]                                                              ?:help [*:8080]
```

We have a couple of feature requests for mitmproxy that would make really great additions to mitmproxy, but haven't been tackled yet. This project would consist of multiple "mini-projects" spanning from a few days to a few weeks, allowing you to work on isolated tasks at different parts of the code base.

We want to bring mitmproxy, and all related tools to using the latest technology - this includes porting our project to Python 3! We already finished porting "netlib", our internal networking library, so pathod and mitmproxy need to be tackled next. With ~90% test coverage on the project, you can port code without being afraid to break everything.
Currently mitmproxy records all flows in custom file format. While this works very nicely for our major use-cases, we'd like to share mitmproxy dumps with others! Long story short, the idea is to re-implement our flow storage format as an SQLite database.
Electron is a very nifty way to package a JS-based web-app into its own application without any browser. We would like to wrap "mitmweb" in Electron, so that we can distribute mitmproxy with a nice frontend for the user.
mitmproxy does not only support HTTP, but also TCP, if you enable the `--raw-tcp` flag. Why is that feature so hidden? Because TCP traffic is not displayed in our UI yet. Just like our HTTP flows, we'd love to have TCP flows that do just that.
With the freshly landed HTTP/2 support in mitmproxy, we now support the full feature set of RFC 7540. Currently mitmproxy does not expose many of these features, and indicates only few things related to HTTP/2 in the UI. It would be great to display a PUSH_PROMISE indicator on requests, show certain things about the connection state and properly mark reset streams.

NB: We actually think that mitmproxy's core is already in a very good shape - we just couldn't resist with the title. :-)

Other Great Projects

Project Name: Project 8 - Rumāl
Mentor: Pietro Delsante (IT)
Backup mentor: Andrea De Pasquale (IT)
Skills required: Python, Django + TastyPie, HTML/JavaScript, MongoDB
Project type: Improve existing tool
Project goal: Provide a web GUI for Thug, designed as a sort of social network where data can be enriched with metadata coming from various sources, and where users can share results, settings, analyses and whatever else.
Description:
Thug [1] is a client honeypot developed during previous GSoC years that is used to analyse potentially malicious websites. Now that Thug is pretty stable and in general use, this project aims to be Thug's dress - providing a convenient web GUI - but also its weapon, as it should provide a set of tools that should enrich Thug's output with new metadata and allow for correlation of results.

Rumāl is composed of a front-end and a back-end, each running a set of daemons that provide the main functionality, some RESTful APIs and the web GUI. Rumāl is written in pure Python, using Django for the web server and Django-Tastypie for the APIs; the HTML/JavaScript part is made with standard libraries like Material.js, jQuery, jQuery UI and so on.

While it is perfectly possible to use it as a simple web GUI for Thug on your own computer, with you as the only user, we want to take Rumāl to a powerful multi-user environment with you. During GSoC 2016, we want to add all the social elements that are required to make it a strong, cooperative platform. This includes elements like user profiles, data sharing, correlated searches and so on.

[1] https://github.com/buffer/thug

Project Name: Project 9 - YAPDNS
Mentor: Andrea De Pasquale (IT)
Backup mentor: Pietro Delsante (IT)
Skills required: Python, Django, HTML/JavaScript, PostgreSQL/MySQL
Project type: New tool

Project goal: Collect Passive DNS data from various sources; display, correlate and analyze them.
Description:
There are a couple of tools out there to collect Passive DNS data (e.g. passivedns by gamelinux and pdnsd), but they only work by sniffing authoritative DNS answers inside network traffic and by storing them. There is a huge amount of other sources that could be used to collect Passive DNS data: for example, almost every organization has a web proxy or gateway, and its logs almost always contain a domain name, an IP address and a timestamp. The same data set can be extracted from other textual logs from DNS servers (Bind, Microsoft DNS, etc), web servers, IDS/IPS, and even sandboxes (Cuckoo) and honeypots (Thug) or other Passive DNS databases (VirusTotal, DNSDB, etc). YAPDSN should provide an interface (e.g. a Syslog-NG local destination or a Logstash module) to collect basic associations between an IP address and a domain name, along with the first and last time the association was seen. Other data can be added for specific log sources (e.g. DNS logs also contain TTL, record type, etc), or gathered from external repositories (e.g. association with malware in VirusTotal's database, etc).

YAPDNS should also provide an interface with a search engine, a set of dashboards and some correlation rules (e.g. track by ASN, geolocation, fast-flux behaviour, etc). The tool should also provide some REST-like APIs to facilitate integration with other tools.
YAPDNS should also use HPFriends to facilitate data sharing between various trusted entities. The backend database may either be a relational database (e.g. PostgreSQL) or a non-relational one (e.g. MongoDB or ElasticSearch).

Communication with other projects and software may use the Common Output Format proposed by this draft on IETF.

You can find more info on YAPDNS in this blog post: http://www.sysenter-honeynet.org/?p=594

Project Name: Project 10 - PCAPOptikon
Mentor: Andrea De Pasquale (IT)
Backup mentor: Pietro Delsante (IT)
Skills required: Python, HTML/JavaScript
Project type: Improve existing tool
Project goal: Provide a web GUI and a set of RESTful APIs to interact with a Suricata IDS instance and analyze arbitrary PCAP files
Description:
Description: PCAPOptikon is a Django project aimed at providing a simple and easy way to run arbitrary PCAP files through Suricata IDS. It provides a web GUI and a set of RESTful APIs to submit new tasks and get the results.
At the moment, PCAPOptikon's results only contain a list of alerts triggered by Suricata, along with a small blob of binary data captured by the signature. It would be useful to also include a full list of conversations found in the PCAP file, that might provide a context for the alert: for example, the system should provide a list of HTTP requests and responses and associate the IDS alerts to the corresponding request/response pair. This could be done by either integrating Bro IDS [1], that provides an excellent set of protocol dissectors, or by using Suricata's own protocol loggers such as HTTP Logging [2].
The web GUI will also need some restyling to be able to display the information mentioned above.
Current PCAPOptikon code here: https://github.com/pdelsante/pcapoptikon

[1] https://www.bro.org/sphinx/intro/index.html
[2] http://suricata-ids.org/features/all-features/

Project Name: Project 11 - Vulnerability emulation for SNARE and TANNER
Mentor: Lukas Rist (DE)
Backup mentor: Andrea De Pasquale (IT)
Skills required: Python3 and Go
Project type: Improve existing tool
Project goal: Add web vulnerability type emulation for SNARE
Description:
SNARE is a web application honeypot sensor attracting all sort of maliciousness from the Internet. The web page is generated by cloning a real web application and injecting known vulnerabilities. SNARE connects to TANNER, a remote data analysis and classification service, to evaluate HTTP requests and composing the response then served by SNARE. Right now TANNER supports limited vulnerability emulation capabilities which serve more as demonstration.
A student working on this project will learn about HTTP attacks, vulnerability emulation using sandboxing, virtual file systems, system emulation and exposing databases. We will develop a deployment strategy, testing and updating.

Repository:
https://github.com/mushorg/snare
https://github.com/mushorg/tanner

Project Name: Project 12 - Honeypots as a Virtualized network functions
Mentor: Hugo Gonzalez (MX)
Backup mentor: Aniket Panse (IN)
Skills required: Python, C, NFV fundamentals
Project type: New tool (combine several tools)
Project goal: The goal of the project is to create a wrap up around existing honeypots to facilitate integration of them with NFV architectures / products.
Description:
First a brief definition: Network function virtualization (NFV) is a network architecture concept that uses the technologies of IT virtualization to virtualize entire classes of network node functions into building blocks that may connect, or chain together, to create communication services. Firewall or routers could be examples of these services.
One of the main component of NFV framework is "Virtualized network functions" (VNFs) wich are software implementations of network functions that can be deployed on a network function virtualization infrastructure (NFVI). Offering honeypots as VNFs should improve adoption of honeypots in Industry.

The task consist to code a general wrap up to offer existent honeypots (glastopf, conpot, honeyd, mysqladmin-pot , etc) as a VNFs function, exposing them to the overall NFV framework.
[1] https://en.wikipedia.org/wiki/Network_function_virtualization
[2] Network-Function Virtualization (NFV) Proofs of Concept;Framework, GS NFV-PER 002 v1.1.1 (2013-10)

Twitter Facebook LinkedIn