Honeynet Project – GSoC Management Report

Last Updated: 17 September, 2009

Table of Contents

Executive Summary

The Google Summer of Code (GSoC) is annual event sponsored every summer by Google. The purpose of the annual event is to promote and develop the field of open source code for students. Google selects approximately one hundred different organizations to sponsor open source projects. Orgaizations selected can have one to many different projects. Each project must have an assigned mentor that has extensive experience in open source development. Students then apply for these projects. Mentors (and the sponsoring organizations) review the student applications and then select one student for each mentor. These students, under the guidance of each mentor, then spend three months during their summer break developing new code and tools. These tools are then released to the public at the end of the summer under an approved open source license. In addition to providing the infrastructure and guidance, Google also funds each student $4,500 for their time and effort. The Honeynet Project was one of the selected organizations for the summer of 2009. It sponsored nine projects for GSoC. In addition, the Honeynet Project sponsored another three projects on its own, to include funding each student $4,500 out of it's own finances. This is referred to as HSoC.

Executive Findings

Overall the program was a success, we definitely want to do it again next year. In addition, the independent HSoC event sponsored and finances by the Honeynet Project worked out better then expected. It is highly recommended we have more of these events throughout the year. Some key points and lessons learned.

a. Each student should be assigned to no more then one project. Communication becomes very difficult when there is two or more projects. Often one of the students project's suffer while the other one does well.

b. Use of IRC channel was a big help, encourage more of it.

c. Use of maillists helped tremendous.

d. Preparation is key.

e. Each student create a YouTube video explaining project with screenshots on use.

f. Blogging very helpful, but need to improve the interface.

GSoC Results

Project 1 – Develop and Improve PhoneyC

PhoneyC is a low-interaction client honeypot designed to allow researcher to quickly and easily identify and analyze malicious websites and their malware. We hope to be adding DOM emulation and automated shellcode detection using LibEmu this summer, amongst other features, to help improve detection and performance.
Student: Zhijie Chen <chenzhijie@icst.pku.edu.cn>
Primary Mentor: Jose Nazario <jose@arbor.net>
http://code.google.com/p/phoneyc/source/browse/phoneyc#phoneyc/branches/phoneyc-honeyjs
https://www.honeynet.org/gsoc/project1
http://joyan.appspot.com/tag/phoneyc

1. What were your project goals for GSoC 2009 and were you able to achieve them?

   a.   The original goal of GSoC 2009 Project 1 is to add shellcode detection and emulation mechanism, and also support collection of downloads. To achieve these, I have finished two modules called honeyjs and hcalert, basically doing the following jobs:

   b.   Shellcode Detection using Libemu: I trace the javascript engine in phoneyc (spidermonkey) in the opcode level and interrupt it at each assignment. If it's a string assignment and the length of the right value is between 32 and 65535 bytes, the honeyjs module will use libemu to check if the string contains shellcodes.

   c.   Heapspray Sledge Detection based on entropy: As there are shellcodes that libemu fails to detect them, the honeyjs module will also calculate the infomation entropy value of strings longer than 65535 bytes to detect heapspray sledges. If the entropy value is lower than 1.0 (or some other threshold, i haven't decided yet), the module will raise an heapspray alert.

   d.   Shellcode Analysis using Libemu: I have extended the libemu's python binding so it supports shellcode analysis APIs and return the shellcode's profile (sequence of the function calls, its arguments and the return values) into python context.

   e.   Collection of Downloads: After a successful analysis of the shellcode, the url in the RLDownloadToFilecall in the shellcode will be retrieved through wget.

2. What do you plan on doing with your project now that you are done?
I will merge the codes into the trunk and discuss the future work needed to do with my mentor (Jose Nazario).

3. Would you do the event again next year?

I'd like to, if there are any good ideas.

4. What is one thing honeynet project can do to improve the program

Improve the blogging system... Some xmlrpc support or at least supporting more html tags, and i still don't know how to _upload_ images...:p

5. What is one thing that we did right and should do next year?

The irc channel and maillist is good and very useful. The call for status update and notifications are good stimulations for me.

6. Any other feedback you would like to provide to the management about HP or GSoC?

Very responsible organisation. HP should apply next year.

Project 2 – Develop and Improve PhoneyC

PhoneyC is a low-interaction client honeypot designed to allow researcher to quickly and easily identify and analyze malicious websites and their malware. We hope to be adding DOM emulation and automated shellcode detection using LibEmu this summer, amongst other features, to help improve detection and performance..
Student: Geng Wang
Primary Mentor: Jose Nazario

1. What were your project goals for GSoC 2009 and were you able to achieve them?

 The original idea of phoneyc is that we extract the javascript code from a html document, add some DOM simulation (which is also in javascript), then use spidermonkey to execute it, and finally put the output content into analysis. The code we submitted when applying for GSoC is like this, and the original goal is to get a perfect DOM simulation, as spidermonkey often crash because of the inadequate of our DOM simulation. Under the advice of my mentor, we gave up our original code, but use python-spidermonkey, which is also an open source project to handle the javascirpt code. It can be used as a python based javascript interpreter, and allows us to execute a piece of javascript code at a time in a python program. We altared our goal, not to simulate perfectly, but only enough to handle most cases of malicious content. We transfer our DOM simulation part from javascript into python, because python is more powerful than javascript. For an object properly defined in python, We can intercept by methods __getattr__, __setattr__ and __call__ each access, modify or call (with arguments) to its attributes or methods. This made it easier for us to handle some details which are usually used to hide malicious content, such as innerHTML (or we have to assign every innerHTML of DOM objects, that is huge work and may affect the efficiency of our program). It also avoids those errors caused by incomplete simulation of DOM and ActiveX objects, and the intercepted function calls or attribute changes will be checked to find if there is some malicious content in them. This is very useful for unknown ActiveX objects. By using py-sm, a python object can be used in javascript just as a javascript object. So, we can combine those DOM and ActiveX objects closely to the javascript code without causing additional trouble. Besides, we transferred the ActiveX simulation script from javascript to python to accord with the new structure of the program.
By testing on the samples on milworm and something else, the malicious contents can be detected in most cases (but not all cases). I think my GSoC project is partly successful, still with some troubles.

2. What do you plan on doing with your project now that you are done?

In fact, there are still some troubles on my program. It cannot support character sets other than ASCII well, and sometimes it is unsafe when transferred from javascript to python. The most common case is that 'undefined' and 'null' in javascript are both 'None' in python, but 'typeof undefined' != 'typeof null'. And I wonder if there is still more bugs to fix, since the testing job is somehow not enough.

3. Would you do the event again next year?

Surely I will if I still have enough time next year.

4. What is one thing honeynet project can do to improve the program?
There is a group working on phoneyc in Beijing, including an assistant mentor, so I have not met with too much difficulties on the project. But I do expect more advice from honeynet project, not only from my mentor. (There is, however, one who tested the program and gave me feedback, and I'm grateful.)

5. What is one thing that we did right and should do next year?
The mail list and irc are both fine, the blog system, although with some bugs, provides a good way for communication. I wish they could be found more useful next year.

6. Any other feedback you would like to provide to the management about HP or GSoC?
Nothing else. Thanks for giving me the chance to get involved in GSoC. It's a precious experience and i learned a lot from it.

Project 1 & 2 – Develop and Improve PhoneyC (Mentor Feedback)

PhoneyC is a low-interaction client honeypot designed to allow researcher to quickly and easily identify and analyze malicious websites and their malware. We hope to be adding DOM emulation and automated shellcode detection using LibEmu this summer, amongst other features, to help improve detection and performance..
Primary Mentor: Jose Nazario

1. What were your project goals for GSoC 2009 and were you able to achieve them?
o Deobfuscation Mechanisms by combining DOM Simulation and Javascript Engine

Geng led this effort and greatly improved the DOM handling. In the past year or two a number of bugs in PhoneyC became very common triggered and the tool would fail to properly analyze HTML or to find links. For example, pages now routinely alter the "innerHtml" elements to set up the exploits, and PhoneyC needed a more robust DOM engine to satisfy this. Thanks to Geng we have it! He moved the DOM handling out of JS (it had previously been mocked up as a Document object in a JS prepend) and into Python for more manageable code. All of the tests reveal that this works very well, and we now have a more complete DOM engine.

o Client-side vulnerability analysis, signature generation and detection

In addition to the libEmu-based shellcode detection and analysis, we've begun making generic ActiveX modules to facilitate future exploits without writing specific modules each time.

o Client-side exploit analysis to support collection of downloads

Zhijie led this effort to integrate libEmu as a shellcode detection and analysis engine in PhoneyC, adding heap spray detection along the way, basic heuristics on shellcode entropy as well as detailed shellcode analysis to handle alerts. This also required some additions to libEmu's Python bindings. From this we can gather next stage URLs to gather more malware via a download.

2. What do you plan on doing with your project now that you are done?
Merge it into trunk, merge in some of my output changes, and prepare a 1.0 release.

3. Would you do the event again next year?
Maybe, I think we got some great students this year and they were supported by the GSOC effort. I need to devote more time to it, but luckily I had two motivated students.

4. What is one thing honeynet project can do to improve the program?
Have our stuff - forums, code repositories, etc - well before set up before things begin.

5. What is one thing that we did right and should do next year?

We clearly made a good call for students: we outlined projects, goals, requirements; we screened candidates, and we allocated slots reasonably.

6. Any other feedback you would like to provide to the management about HP or GSoC?

Too much chatter on the official GSoC mailing list; Melange is more fragile than I would expect for a Google effort; the "pencils down" date enforcement was a bit sketchy.

Project 3 – Qebek: QEMU Based Sebek

Advanced new data capture technique for virtualized environments, looking to extend our existing work on Sebek for high interaction honeypot I/O capture to the hypervizor layer, for increased stealth and performance.
Student: Chengyu Song <chengyu.song@gmail.com>
Primary Mentor: Brian Hay <brian.hay@alaska.edu>
https://projects.honeynet.org/sebek/wiki/Qebek

1. What were your project goals for GSoC 2009 and were you able to achieve them?
Qebek has three milestones:

MS1: A generate hooking framework can could be used to hook any kind of function calls, and supports accurate pre & post call interception. This is mainly for creating an environment where I or anybody who is interested in adding new hookings could conveniently use the old experience from writing OS based hooking. The framework should also include some basic supports for reading raw data of the virtual machine. This milestone is almost but not completely achieved, it lacks accurate post call interception between thread switch within a process, e.g. thread A calls functionA, but before this function successfully returns, thread A is suspended and OS switch thread B from the same process to execute, if thread B calls functionA too, then, when functionA returns, the hooking framework may call the postcall callback function with wrong user_data.

MS2: Port all the functionalities from old Sebek Win32 version into this new framework. This milestone can be further divided into two stage. The first one is to port ConsoleSpy, which is used to monitor keystrokes inside the virtual machine. The second one is to port NetworkSpy, the component used to capture network activities of processes. This information is then used to correlate network flows and OS processes. The current Qebek is able to feed almost the same data to sebekd as existing version does. But unfortunately, this milestone is also not completely achieved either. There are two main flaws. One is it lacks a full function Unicode to ANSI converter. The other is NetworkSpy cannot capture sys_accept of port 139,445. The sys_accept event is reported normally for processes other than svchost. So it seems like svchost uses a different way to create a listening socket other than NtDeviceIoControlFile.

MS3: Improve the data correlation of ConsoleSpy and NetworkSpy. This is mainly to solve the problem that current Sebek cannot accurately tell which network flow is binded to which console. This milestone is not full achieved. The data for correlating (socket/std handle) is ready, but I am not sure if I could use the fd field in Sebek header to store this information. So this information is not reported now.

2. What do you plan on doing with your project now that you are done?
Fix flaws mentioned above. Optimize the performance. Add new functionality like kernel object observation. Port to new TCG framework. Port Linux version.

3. Would you do the event again next year?

If there is any interesting ideas. Maybe a new bootkit based Sebek if I couldn't managed to do this before next summer.

4. What is one thing honeynet project can do to improve the program

The blog system. And maybe we could provide a test set.

5. What is one thing that we did right and should do next year?

The communication platform. And the HSoC.

6. Any other feedback you would like to provide to the management about HP or GSoC?

Thank you very much for this summer! And we should apply next year.

Project 4 – Sebek Data Visualization

Develop and improve the ability to read and visualize Sebek data from high interaction honeypots.
Student: Kevin Galloway
Primary Mentor: Kara Nance
https://www.honeynet.org/blog/197

1. What were your project goals for GSoC 2009 and were you able to achieve them?
My goals for GSoC 2009 were to create a visualization for Sebek data, that would help users analyze
trends in the data collected from multiple instances of Sebek. This was done through two main parts:
a parser that would load the data into the visualizer, and then the visualizer itself. The parser is comprised of a
python script that separates the data received from Sebek into several lines that the visualizer can read in. The
two important pieces of data are a numerical timestamp, so that the visualizer can order the events properly,
and the event themselves, so that the visualizer can group them, according to the user's preferences. The
visualizer has three separate visualizations, so that a user has multiple tools to be able to identify trends. The
first is just a simple separation based on color, and a similar one is based on height. Both of these show how
each event follows the previous, to show patterns. The third shows how much of each event occured in a five-
second interval and allows the user to step through them to also identify trends. All the visualizations work for a
single instance of Sebek, along with multiple instances. Both of these goals were realized, although there are
some improvements to be made.

2. What do you plan on doing with your project now that you are done?
My plan for the project is to implement several improvements in it, mainly focusing on the User Interface, as well
as making it a little more versatile in both the data in can handle, and in regards to user customizability.
Optimizing the code, along with making it a little more stable is something else that I would like to work on, as
well as trying to solicit advice from my mentor and other visualization people to incorporate into the project.

3. Would you do the event again next year?
If I had a good idea, and I'm still a student, then sure.

4. What is one thing honeynet project can do to improve the program
The big thing for me would be encouraging more communication between students and members of the
Honeynet Project in general. I would have liked for the various developers to comment on each others'
projects and the like.

5. What is one thing that we did right and should do next year?
The blogs were especially helpful for helping me keep to a schedule, as well as giving me a place to
place updates that others can see.

6. Any other feedback you would like to provide to the management about HP or GSoC?

The management was very good, and the variety of projects were another plus, and I definitely think the HP should apply next year.

Project 5 – Improve High Interaction Honeypots

Improve our Capture-HPC and Capture-BAT high interaction client honeypots, including bug fixing, improved performance, new functionality and simplified use.

Student: Van Lam Le

Primary Mentor: Peter Komisarczuk

1. What were your project goals for GSoC 2009 and were you able to achieve them?

The main goal for GSoC 2009 is improving Capture-HPC, a hight interaction client honeypot. The project has three parts: adding database functionalities to capture-server, testing and adding network monitor functionalities to capture-client, and creating framework to develop WindowsAI for Capture-HPC. However, I have just finished the first two parts because I had spent a lot of time for testing and developing network monitor functionalities.

2. what do you plan on doing with your project now that you are done?

 I intend to add a friendly user interface for Capture-HPC. It can make easier for end users to use it. In my opinion, web interface could be a good choice in this case.

3. would you do the event again next year?

  I'm going to work on Capture-HPC but not sure for next event.

4. What is one thing honeynet project can do to improve the program

It is great if we have a communication channel between students who are working in honeynet projects. Setting up a place where students can put their codes and let communities give feedbacks.

5. What is one thing that we did right and should do next year?

Friendly updates and reminders are so great!

6. Any other feedback you would like to provide to the management about HP or GSoC?

Project 6 – Develop Hybrid Honeypot Architecture

Combine the functionalities and advantages of low and high interaction honeypots into a single solution by intelligently switching the recipients of known and unknown attacks at the network flow level.
Student: Robin Berthier
Primary Mentor: Georg Wicherski
http://honeybrid.sourceforge.net

1. What were your project goals for GSoC 2009 and were you able to achieve them?
My goal was to turn Honeybrid from a prototype into a robust application to help deploy honeypot-based experiments. This goal involved the following tasks:

a. cleaning up the code,
b. adding a parser for the configuration file,
c. re-writing the module-based decision engine,
d. adding a module to interface Honeybrid with Snort.

I was able to achieve tasks a, b, and c

2. What do you plan on doing with your project now that you are done?
I plan on:
*   continuing the development to complete task 4) and to add support for autoconf/automake,
*   having Honeybrid tested and evaluated by other people,
*   integrating Honeybrid with other honeypot tools (for example: to be
*   included as part of Honeywall, or to be able to output data to Nebula)

3. Would you do the event again next year?
I won't be a student next year, but I would definitely like to be involved in the next GSoC with the Honeynet Project.

4. What is one thing honeynet project can do to improve the program
I think it would help if students could receive more feedback from the community.

5. What is one thing that we did right and should do next year?
Communication from the organizers was great, it was really motivating.

6. Any other feedback you would like to provide to the management about HP or GSoC?
I think the Honeynet Project fits very well with the mission of the GSoC, and it showed that it has the right skill-set to mentor a great variety of projects, so HP should apply next year.

Project 7 – Improving PicVIZ

PicViz is a dynamic tool used to visualize log and honeypot data via parallel coordinate charts.
Student: Victor Amaducci <victor@las.ic.unicamp.br>
Primary Mentor: Sebastien Tricaud <sebastien.tricaud@gmail.com>
http://www.las.ic.unicamp.br

1. What were your project goals for GSoC 2009 and were you able to achieve them?
Picviz 0.5 release have been a poor GUI and need improve its feature. This proposal was aimed to define some features and changes on Picviz to improve the analizis of date ploted by Picviz. Before to implement the proposal goals, we need learn "how Picviz work?" and the first observation was "The Picviz have been implemented on procedural paradigm" and we decided change it to "object-oriented paradigm" . After change the picviz to Obect Oriented programming, we start the implement of our proposal goals.

PROPOSAL GOALS:

a.   Addition controls to change relations among the axises improving the preview of graphic; This was one of first goals to be implemented because we believe that it was a most important feature. this feature is ready on Picviz 0.6 release.

b.   Addition features to apply zoom/focus effect in some aspect from graphic;   This feature was implemented and is perfect function, but don't is available on Picviz 0.6 release. Will be available on Picviz 0.7 release.

c.   Addition features to apply scroll on graphic (like google maps) so that to improve the preview of graphic from big datasets; This feature is not implemented, because we reached the conclusion that it was not necessary because the Picviz already has a scroll bar and it is sufficient to navegate on graphic ploted.

d.   Addition features to apply brush effect to provide method of plot marks manually; This feature was implemented and is perfect function, but don't is full available on Picviz 0.6 release. Will be full available on Picviz 0.7 release.

e.   Addition controls to choice a file log and a parser log, using the frontend, so that become easy the build of graphic from any service log;   To perfect function this feature, will be need some changes on Picviz-lib, and it is not on scope this proposal, but this feature was implemented and is function, but don't is full available on Picviz 0.6 release. Will be full available on Picviz 0.7 release.

f.   Automatic parser creation through user aided selection of sections text in the log. This feature is not implemented, because we reached the conclusion that it is not on scope the PicViz tool.

g. Dynamic graphic ploting;   This feature neeed much changes on picviz-lib and turn it is impossible at moment, but on future works on Picviz this feature will be the main Goal to be implemented.

h. Undo and Redo operations;   After most of the goals to be implemented, we reached conclusion that we need to implement a mechanism to Undo an Redo operations. So this feature was implemented and available on Picviz 0.7 release. The Picviz 0.7 release is a GSOC Final Picviz release that will be available on next week.

2. What do you plan on doing with your project now that you are done?
I wanna write a paper about the Improve Picviz and yours features. This paper will describe how picviz (and yours feature that I contribuited to implement) help someone when this person need to analize the file logs

3. Would you do the event again next year?
Yes I would. But I gonna dedicated me to show my experience on GSOC for people with Seminars and Lecturs because this program is a great program and new persons need participate its at least once.

4. What is one thing honeynet project can do to improve the program?
Rise the projects. The honeynet has much projects and can available more projects to program.

5. What is one thing that we did right and should do next year?
The idea of each students maintain a blog posting news about yours project ias a good Idea and should do next year.

6. Any other feedback you would like to provide to the management about HP or GSoC?
 Was a pleasure work with you on this summer, and the Sebastian Tricaud is a great mentor, I hope the next year you have more students and more projects on this program.

Project 8 – Web Application Honeypot

Many of today's most advanced attacks now happen at the web application layer. This solution is designed to capture information on the latest web application attacks using scalable and easy to deploy low interaction server honeypots.
Student: Lukas Rist <glaslos@googlemail.com>
Primary Mentor: Thorsten Holz <thorsten.holz@gmail.com>
http://trac.glastopf.org/trac
http://glasblog.glastopf.org (mostly German articles)
https://www.honeynet.org/blog/201

1. What were your project goals for GSoC 2009 and were you able to achieve them?
My goal was to push forward the development on a Honeypot for an attack vector in web security which is really underestimated in current discussions. The main objectives could be merged into one intention: Increasing our attractiveness and answering every request as close as possible to a real world system. This got achieved with the new PHP file parser and the dynamic Google dork list which we provide for the Google crawler.

2. What do you plan on doing with your project now that you are done?
I have collected tons of data during the last months. My next goal would be a deeper analysis of this data. After that I plan to continue the work on the PHP parser to get better results for attack replies. During the GSoC time I've met many peoples which are interested in my project so I also try to intensify those contacts.

3. Would you do the event again next year?
If I can fit in any project, sure!

4. what is one thing honeynet project can do to improve the program
I have missed a little bit some sort of reviews, comments or suggestions from the HP developers on my project and progress.

5. what is one thing that we did right and should do next year?
The reports for the HP blog has been a good reason to reflect my own progress and goals. I think this is one thing that should be retained next year.

6. Any other feedback you would like to provide to the management about HP or GSoC?
Actually my project isn't integrated into an HP chapter but I think the HP is a very good environment for it. Are there any plans how we proceed?

Project 9 – Managing Honeypot Client

Adapting to some of the challenges presented by attempting to operate larger, longer running deployments of client honeypots (both high and low interaction) by simplifying and improving our ability to deploy and manage large client honeypot farms.

Student: Thibaut Gadiolet

Primary Mentor: Ian Welch

http://ecs.victoria.ac.nz/Groups/DSRG/HoneyClientManager

1. What were your project goals for GSoC 2009 and were you able to achieve them?

Our goal was to develop a modular web application that would allow users to request honeyclients belonging to multiple organisations to analyse supplied URLs. The analysis would be done asynchronously and the user could login in later to see the results. The results would be stored in a database and some analysis would be possible. The modular architecture would allow the presentation, logic and analysis layers to be updated independently allowing better maintainability. Possible types of analysis were:

* Persisting results data
* Summarizing trends and presenting the results to multiple users
* An intelligent graphical representation for aggregated data
* A history of web sites to check their evolution
* Some features like: filtering, search and alert.

The architecture has been implemented using SVN, Mysql/Hibernate, Spring, Maven, Tapestry and Tomcat. The system allows users to submit requests to honeyclients. However, we are using a dummy version of the honeyclients (essentially a proxy class) because implementing a web service for Capture-HPC (our initial target honeyclient) was going to exhaust our time so we focused on building a sound web application with a well-defined interface to honeyclients instead. Not all of the analysis has been implemented. We could add more trend analysis, history view and email alerting.

2. What do you plan on doing with your project now that you are done?

   a.   Make the code available via the honeypot project svn (done).
   b.   We would like to make it available for testing. Some of this depends upon finding an a suitable home. It is possible for us to host it within our security lab but this makes external collaboration difficult as our University IT people are not keen on providing external access to non-staff or students.
   c.   We would like to build the bridge or proxy to allow invocation of Capture-HPC. As part of MSc project a prototype was built but it relies upon Grid technologies and this maybe overkill for our purposes. One plan is to offer development of a proxy and integration into the web application as fourth year engineering project.

d. Our current security architecture focuses on authenticating users. We need to extend this to allow federated authentication of participating users, organisations and honeyclients.

3. Would you do the event again next year?

Yes definitely. I would like to develop this project further.

4. What is one thing honeynet project can do to improve the program

Ask mentors/mentees to develop demo videos that could be posted somewhere (say YouTube). We developed a screencast of the honeypot manager and intend to post this to increase visibility of the project.

5. What is one thing that we did right and should do next year?

The mailing lists were very useful! Asking mentors/mentees to provide regular updates of the honeyblog.

HSoC Results

Project 10 – Improving the effectiveness of low interaction honeypots.

Nepenthes is one of the leading low-interaction honeypots designed to automate the capture and analysis of malware, and this project will be a next generation development of low interaction server honeypots able to automatically and scalably detect known and unknown malware.
Student: Markus Koetter & Mark Schloesser <ms@mwcollect.org>
Primary Mentor: Paul Baecher <pbaecher@gmail.com>
http://dionaea.carnivore.it
websvn: http://svn.carnivore.it/browser/dionaea/trunk/
irc: freenode /#nepenthes for `personal` contact

1. What were your project goals for GSoC 2009 and were you able to achieve them?
Project goal was to write a new low interaction honeypot which is meant to be a successor to the nepenthes low interaction honeypot. Improvements to nepenthes:
* support ipv6
* support ssl/tls
* allow binding services to multiple single ips instead of only one (or any)
* threading to detect attacks, allows making use of multi core plattforms
* embed the python scripting language
* write smb emulation in python to detect attacks for *all* known bugs on the well known ports (Mark)
* emulate cmd.exe in python, parse the input
* download files via tftp, http and ftp
* store successfully downloaded files
* embed libemu to detect shellcodes using emulation
* embed libemu to profile shellcodes using emulation

All project goals were archived.

2. What do you plan on doing with your project now that you are done?
* embed libemu to emulate shellcodes using emulation
* detect unicode shellcode
* improve documentation
  * how it works
  * on the embedded python interpreter, as it is really powerfull, but nobody knows about it
* we still need a logo
* automake needs a curl version check

3. Would you do the event again next year?

Maybe with a 'slightly' smaller project.

4. What is one thing honeynet project can do to improve the program

   a.   Improve David Watson's immune system, I missed him from time to time.

   b.   More inter-student communication.

   c.   Do not allow external resources to post progress reports, central is preferred.

   d.   Ask participants send links to their reports to the ml.

   e.   File the AAR - For the Community report within the first 2 weeks of the program, it was pretty impossible to get links to code/documentation for all projects during soc.

5. What is one thing that we did right and should do next year?

Take part in gsoc, fund development.

Usually you can't get things done that cheap in such short period, h/gsoc makes it possible.

6. Any other feedback you would like to provide to the management about HP or GSoC?

*Soc was an amazing flood of great progress in very different research projects.

Project 11 – Develop and Improve Nebula

Nebula is a distributed system, fed by honeypots, that automates the generation of IDS signatures to detect and identify attacks. Goals are to improve nebula so that the quality of generated signatures gets better. Also, more and different types of sensors should be able to contribute to a distributed signature generation setup.
Student: Tillmann Werner <werner@informatik.uni-bonn.de>
Primary Mentor: Felix Leder <leder@cs.uni-bonn.de>
http://nebula.carnivore.it

1. What were your project goals for GSoC 2009 and were you able to achieve them?
The project was split into the following three work packages:

WP1: Nebula's clustering algorithm is one of the two critical components (the other important one is the signature generation engine). In current implementation, clusters are built following a relatively easy strategy. In particular, cluster density is not considered. The classification results can be improved by replacing the clustering algorithm with a density aware alternative. Another improvement would be to save classification states to files so that the nebula daemon can be interrupted in its execution (expectedly or unexpectedly) and resume from the latest state after a restart.

WP2: Sources that want to submit data to a nebula daemon must implement a proprietary protocol which was designed especially for secure and efficient attack submissions. To date, a submission module exists for honeytrap only. A command line client is available for manual submission of files. In this work packages, a libnebula C library will be developed for easy integration of nebula client functionality into other applications (e.g., nepenthes, snort).

WP3: Intrusion detection systems such as snort can also be used to submit detected attacks to nebula. Although this might at first seem pointless, it is not: Classical intrusion detection signatures often trigger on very simple features. Nebula could be used to generate more complex signatures for the corresponding attacks - not to be used for detection, but for a better understanding of an attack's properties. Signatures computed by nebula extract structural commonalities and are thus very helpful for understanding an attack class's anatomy. In work package 3, a snort plugin will be written based on the library developed in WP2. Snort is the best application around for UDP and TCP stream reassembly which is necessary to reconstruct complete attack strings prior to submission to nebula. As it might perform better to unlink intrusion detection from stream reassembly, a different application will be implemented based on libnids and libnebula. It might even be possible to let this application interact with snort.

While working on WP1, it turned out that it is not beneficial to include density information in the classification decision. The reasons are a bit technical: nebula maps inputs in a metric space, but not in a vector space. Thus, maintaining density information about a cluster (e.g., its diameter) requires testing all present elements

whenever a new element comes in, which is basically too expensive and would slow down the overall algorithm too much. One thing that was not initially planned is an improvement to the signature segment routine. The original greedy strategy was adopted to select more segments, resulting in more accurate signatures. Details are available in my blog post at <http://honeynet.org/node/440>.

The other part of WP1, the state saving and loading capability, has been implemented, but again it turned out that one month was not sufficient. It just took some time to evaluate alternatives for an appropriate storage technology. I thought about inventing my own proprietary and put it in a flat file, which has the advantage to be independent from any third-party libraries. The idea was discarded, because that would also come with the requirement for self-implemented and well-performing query mechanisms, which is a fairly complex thing, Then I thought of using an XML library to save state information in a serialized format, but I did not find any library that seemed mature enough. Further, I am not really a big fan of storing each and every piece of information in a human-readable but bloated format. Finally, I chose sqlite, a lightweight sql database implementation, that stores everything in a single file. The sql it supports is sufficient, and the interface is easy to understand and use. The whole signature generator state engine was written in only about 700 lines of code.

WP1 turned out to require a little more work and time than initially thought. Luckily, this was getting clear very soon at the beginning, and I could approach tasks from the other work packages a bit earlier and work in parallel to stay on time. WP2 was an easy one. The libnebula C library was implemented without any major problems. Existing clients (a honeytrap module and the command line client that comes with nebula) were ported to use the new library. In WP3, two new nebula clients have been implemented on the basis of two intrusion detection systems, namely snort and libnids. The intention was to create passive nebula sensors that can be placed in front of high-interaction honeypots, perform stream reassembly, and submit reconstructed sessions to a nebula daemon. Snort's stream reassembly API is not really documented, and it took me some time experimenting with it to find out how to use it properly. But its performance is quite good, and target-based reassembly policy support (aka stream5) is a cool feature.

Conclusion: Most goals were achieved. The density-based clustering could not have been implemented. Instead, the signature segment selection has been improved.

2. What do you plan on doing with your project now that you are done?
There are lots of ideas how to improve and extend nebula further: libnebula needs support for non-blocking I/O. There are still some things that can be done to improve performance during signature generation. And the website requires some work, it is still incomplete.

3. Would you do the event again next year?
Depending on ideas I have, yes, maybe. Would not make any sense without proper content.

4. What is one thing honeynet project can do to improve the program?

Membership involvement can be improved. Many of the members are very experienced and could help in discussions and with testing. This year it was my impression that only few members followed GSoC activities, apart from the mentors.

5. What is one thing that we did right and should do next year?
The communication platforms were really helpful. The mailing list and the IRC channel were an easy way to stay informed and ask questions.

6. Any other feedback you would like to provide to the management about HP or GSoC?
Cool action, nice outcomes. HP should apply again next year.