Strengthening Capsicum Capabilities with Libpreopen

(Changed the title by modifying the file `thesis.tex`)

by

© *Stanley Uche Godfrey* (change this in `thesis.tex`)

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the

requirements for the degree of

Master of *Science* (change this in `thesis.tex`)

Department of *Scientific Computing* (change this in `thesis.tex`)

Memorial University of Newfoundland

December 2017 (change this in `thesis.tex`, too)

St. John's Newfoundland

Abstract

This document provides information on how to write your thesis using the LaTeX document preparation system. You can use these files as a template for your own thesis, just replace the content, as necessary. You should put your real abstract here, of course.

"The purpose of the abstract, which should not exceed 150 words for a Masters' thesis or 350 words for a Doctoral thesis, is to provide sufficient information to allow potential readers to decide on relevance of the thesis. Abstracts listed in Dissertation Abstracts International or Masters' Abstracts International should contain appropriate key words and phrases designed to assist electronic searches."

— MUN School of Graduate Studies

Acknowledgements

Put your acknowledgements here...

"Intellectual and practical assistance, advice, encouragement and sources of monetary support should be acknowledged. It is appropriate to acknowledge the prior publication of any material included in the thesis either in this section or in the introductory chapter of the thesis."

— MUN School of Graduate Studies

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Gargets to be Secured

As more networks and gadgets are connected to the internet, the web becomes indispensable, it hosts productivity software suites for creating documents, spreadsheets and emails.Applications suites for scientific calculations, live television streaming and weather details hosted on the web. Some of the services web applications provide are online banking services, services for storing pictures and documents in the cloud, personal computers and mobile devices. Web applications also provide services that connect home devices such IP cameras to mobile phones for remote monitoring and e-commerce services. Sensitive data like passwords and credit card details are usually required to access these web services. These web services can have vulnerabilities which attackers can exploit, despite network defenses like firewall and intrusion prevention systems [3].

One such vulnerability in an application could be as a result of a Buffer Overflow [4]. A Buffer Overflow is a programming bug usually in a non-memory safe programs that results when a programmer fails to check if an input data is within the bounds of that input buffer. If the input data is more than the buffer can accommodate, the overflowing data will overwrite the contents of adjacent memory which may push instructions to be executed into the stack. An attacker who is able to add more data in a buffer than the buffer can accommodate could change execution path of applications intentionally and may acquire the root user right of the system which will allow the attacker to take total control of the system.

If a web application is vulnerable, attackers can use a technique known as Heap Spraying [1] to send malicious code to the heap memory of the web application in a computer. The Heap Spraying technique is used to duplicate the malicious code in different locations of the running application's heap memory to increase the chances of execution of the malicious code. Heap spraying is created with scripting languages like JavaScript. Different Malware exploited vulnerabilities found in internet explorer 6 and 7 around 2004 when the internet explorer web browser was believed to be the most popular web browser. Some of the vulnerabilities exploited in internet explorer include ANI(CVE2007-0038),VML(CVE-2006-4868) and the Operation Aurora exploit (CVE-2010-0248).

The first known buffer overflow exploitation that gained mainstream media attention was accomplished by Robert Morris, a graduate student of Cornell University. [4] Morris wrote an experimental program that duplicates itself in a computer and disseminates itself to other computers through a computer network. Morris was able to put this program on the internet which was fast replicating,infecting and refecting computer at a fast rate. Morris' program, known as a worm exploited a buffer overflow bug in the UNIX Sendmail program, a program which runs on a computer and waits for connections from other computers which it receives emails from. Morris' program also exploited a buffer overflow in UNIX finger. UNIX finger is a program that prints out the login and other details of a logged in user in a UNIX system.

These sorts of unauthorized access to computer resources by attackers are what capsicum mitigates, and libpreopen will make capsicum easier to use in limiting the damage intrusive malicious code from such cyber attack could cause.

1.2 Background

Capsicum is a system that boosts UNIX security with sandboxed capability mode and capabilities. Capability mode is the ability of capsicum to prohibit application fragments to interact with each other except in a regulated manner using capsicum capabilities rights. Fragments of applications in capability mode are totally isolated and these fragments are not allowed access to global namespaces. The reason for these restrictions is to contain vulnerabilities to a fragment and not allow the corruption to spread to other fragments of the application or the entire system.

Processes in total isolation cannot perform any task, this is where capsicum capabilities are required. capsicum capabilities are used to grant isolated processes in capsicum capability mode limited rights to perform specific actions in the capability token on a shared resource. For instance, a process may inherit file descriptors from a parent process or it may request access to a file from another process that has the right to send the file descriptor of the requested system file through IPC and before each of the processes enters capsicum capability mode. Regardless of how capability rights are acquired, processes in capability mode can only perform actions allowed in the capabilities granted on the file descriptors they acquire. A file descriptor acquired with capability right of cap_read cannot be used for fchmod(2) or have cap_write operation perform on it. [2]

For an application to be compartmentalized by capsicum, the application developer would make some modification to make the application conform to capsicum features. The modifications can be rigorous and time-consuming. These difficult application modifications are what libpreopen relieves application developers who want to make use of capsicum compartmentalization features of.

An example of an application developed without capsicum capability sandboxing in mind is tcpdump. tcpdump is a command line application for printing protocols and packets transmitted or received over a connected network and for printing the communication of another user or computer. In a network through which unencrypted traffic such as telnet or HTTP passes, tcpdump can be used to view login details, url and the content of visited websites by a superuser. Packet filter such BPF can be used to limit the number of packets captured by tcpdump.[2] For tcpdump to be sandboxed with capsicum and has its privileges reduced, it must be modified with the code in listing (1.1) and (1.2) and analysed with procstat tool to ensure that the capabilities exposed are the ones intended by the program author.

Listing 1.1: code to add capability mode to tcpdump

```
if (cap_enter() < 0)
 error("cap_enter: %s", pcap_strerror(errno));
status = pcap_loop(pd, cnt, callback, pcap_userdata);
```

Listing 1.2: code to narrow rights delegation in tcpdump

```
if (lc_limitfd(STDIN_FILENO, CAP_FSTAT) < 0)
 error("lc_limitfd: unable to limit STDIN_FILENO");
if (lc_limitfd(STDOUT_FILENO, CAP_FSTAT | CAP_SEEK | CAP_WRITE) < 0)
error("lc_limitfd: unable to limit STDOUT_FILENO");
 if (lc_limitfd(STDERR_FILENO, CAP_FSTAT | CAP_SEEK | CAP_WRITE) < 0)
 error("lc_limitfd: unable to limit STDERR_FILENO");
```

Chapter 2

Design and Implementation of Libpreopen

2.1 Design

The design of libpreopen was made to strengthen capsicum from these two viewpoints.

(1) libpreopen fortifies capsicum by making it possible for an application running in capsicum capability mode to run some commands that require System calls and access global namespaces without compromising the system security.

(2) libpreopen eradicates tedious application modifications, developers have to make in order to incorporate capsicum compartmentalization and sandboxing capabilities in their application.

2.2 Implementation

Libpreopen makes it possible for applications that requires global file namespace access to be sandboxed in capsicum without any modification by the author of these applications. Libpreopen is able to workaround the global file namespace access request of some applications at the moment being sandboxed with capsicum and have these applciations executed successfully in caspsicum capability mode. Listing (2.1) is the header file of libpreopen.

Listing 2.1: The header file of Libpreopen, libpreopen.h

```
struct po_map;

struct po_relpath {
int dirfd;
const char *relative_path;
};

struct po_map* po_map_create(int capacity);

void po_map_free(struct po_map *);


struct po_map* po_map_get(void);

void po_map_set(struct po_map*);
```

```

struct po_map* po_add(struct po_map *map,
 const char *path, int fd);

int po_preopen(struct po_map *, const char *path);

struct po_relpath po_find(struct po_map *map, const char *path,
cap_rights_t *rights);

const char* po_last_error(void);

int po_pack(struct po_map *map);

int po_map_length(struct po_map *map);

const char* po_map_name(struct po_map *map, int i);

int po_map_fd(struct po_map *map, int i);
```

Listing (2.1) is the header of file of libpreopen, listing the structures and functions which libpreopen was implemented with.

Libpreopen creates an extendable storage, pre-opens directories of file system that may be required by an untrusted application and stores the directory descriptors and the paths to the directories in the extendable storage.

Libpreopen has its implementation of lib c functions open(2) , access(2) and stat(2) at the moment. The functions are not allowed in capsicum capability mode because access to global file namespace is required. Therefore libpreopen implements these lib c functions using the fstatat(2) ,openat(2) and faccessat(2) variants which are capsicum capability friendly.When libpreopen is loaded with runtime linker ld_preload as environment variable, libpreopen is loaded before any other library and call to any of libc functions open(2) , access(2) and stat(2) made in the environment will execute libpreopen's version of the function because the search for the function will first be made in libpreopen.

2.3 Capsh

Capsh is a shell program for compartmentalization and running of untrusted application in capsicum capability sandbox. A program to be run in capsh is given as commands to capsh in one line. The first command is the application name and the others that follow are arguments needed to run the application. When a command is given to capsh to execute a program, capsh puts the name of the program to be executed and the program's arguments into a storage. capsh forks itself into a new process, enters capsicum sandbox capability mode and set file descriptors of shared libraries' directories as environment variables using ld_library_path_fds . Setting file descriptors of paths to library directories as enviroment variables conform to capsicum compartmentalization and sandboxing rule. After setting the environment variables, capsh starts the execution of the program with libc function fexecv.

Without libpreopen, capsh cannot do much exciting things. Only applications that do not require access to global file namespace can be executed on capsh for example, echo, a command that prints strings on the terminals of UNIX shell programs. If an attempt to execute a command that requires access to global file system namespaces is made the "action not permitted in capability mode" exception of capsicum is thrown and the program execution terminated.

Capsh with libpreopen at the moment can execute some UNIX commands that require access to global file namespace for execution. Figure 2.3.1 shows the how incorporation of libpreopen into capsh can make safe execution of an application that requires global file namespace in capsicum possible.

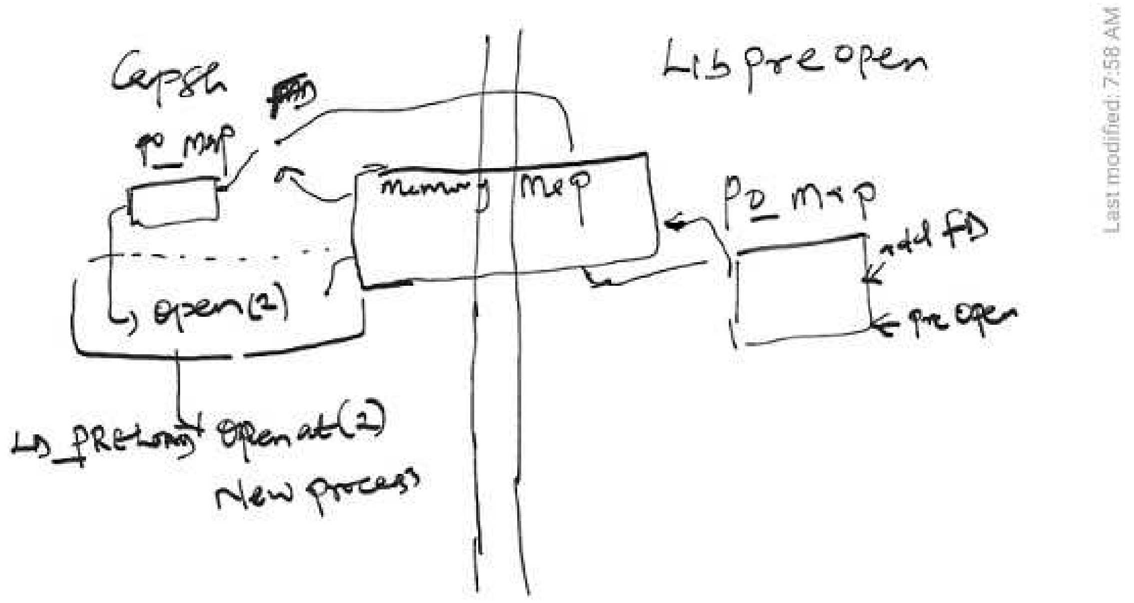

Figure 2.1: Shared memory mapping between Libpreopen and a process started by Capsh

From figure 2.1, it can be observed that when libpreopen is used with capsh, capsh delegates pre-opening of directories an application to be executed in capsh may need access to their contents. The file descriptors and paths to these directories are put in an extendable storage, mapped into shared memory segment and the file descriptor of the shared memory segment returned to capsh by libpreopen. Capsh sets the returned shared memory segment's file descriptor as environment variable, pre-load libpreopen with ld_preload in the environment for the application capsh is about to start executing and fexecves to start the execution of the new application. During the execution of the application, if call to any of lib c functions open(2) , access(2) and stat(2) is encountered, libpreopen's version of these functions which are capsicum sandbox capability friendly will be called instead.

Chapter 3

Evalution

LaTeX can produce cryptic error messages at times. However, with some experience, it is usually not too difficult to determine what the problem is and how to fix it.

As mentioned earlier, appropriate search terms in Google may help you fix these error messages.

Chapter 4

Lorem Ipsum

Now, for your reading pleasure, some *Lorem ipsum*, courtesy of:

`<http://www.lipsum.com/>`

This gives a good view of the margins — note that the left margin is a bit wider than the right margin to accommodate binding.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam odio elit, viverra eu tempor non, pulvinar ac nisi. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Sed adipiscing, dui quis viverra facilisis, quam libero adipiscing justo, vitae dictum libero mauris ac magna. Aenean sem ligula, vulputate at vestibulum eu, pellentesque in justo. Sed et eros mauris, sed placerat nulla. Maecenas nulla velit, facilisis et rutrum nec, volutpat id lorem. Duis vestibulum odio velit, id elementum tortor. Sed pellentesque leo ac nibh iaculis at fermentum orci lobortis. Suspendisse arcu magna, porta nec pretium non, feugiat vitae orci. Vivamus at enim arcu, at sagittis nisl. Vestibulum at mi enim, vel malesuada justo. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.

Nullam sed nunc at enim posuere sagittis. Vivamus augue turpis, mattis a blandit non, sollicitudin non nisl. Integer vestibulum, est vitae cursus adipiscing, elit libero pretium leo, in scelerisque augue felis volutpat nisl. Donec commodo posuere arcu, eget feugiat dui ornare nec. Nullam eros mi, condimentum ac ultricies ac, euismod lobortis nibh. Cras ac ligula pharetra risus elementum pharetra vel in quam. Fusce ac augue vulputate nibh imperdiet convallis sit amet et quam. Integer porttitor dictum fermentum.

Nullam id ante arcu. Nulla facilisi. Vestibulum sodales, mi sodales ultricies pulvinar, orci leo dictum diam, quis imperdiet turpis lacus ut sem. Nulla rutrum odio sit amet elit aliquam blandit gravida nunc placerat. Aenean et neque ut leo condimentum vehicula. Fusce quis orci vitae enim dapibus tincidunt in vel ipsum. Phasellus auctor neque ac eros egestas sit amet ultricies erat vestibulum. Ut erat ligula, pharetra vel hendrerit vitae, mattis ac turpis. Ut malesuada diam vitae lacus vestibulum a tempus nisl posuere. Ut nisi sem, dictum eu laoreet sed, commodo eget enim. Morbi vel lacus neque, tempus fringilla tellus. Nunc id egestas felis. Nullam eu mollis neque. Ut non mauris malesuada eros sagittis congue. Cras vitae felis ut nisl mollis semper ut quis risus. Sed eu arcu urna, et commodo sapien. Donec vestibulum, libero sit amet ultrices blandit, erat lorem volutpat lectus, sed feugiat leo elit in orci. Aliquam vitae leo tellus, placerat pulvinar massa. Nulla at sapien hendrerit diam varius vehicula.

Curabitur et orci nulla. Phasellus euismod, massa non hendrerit dictum, dolor enim imperdiet sapien, vitae commodo lorem tellus eu quam. Duis egestas felis velit. Sed in orci nec nulla rutrum posuere. Suspendisse potenti. Nunc vel quam nisi. In at molestie libero. Aenean hendrerit vestibulum orci, ut hendrerit nulla volutpat lacinia. Vestibulum sit amet sapien vitae lectus gravida vehicula. Suspendisse ac purus sit amet est congue auctor.

Morbi pellentesque, quam vel mattis molestie, augue purus vestibulum lorem, nec consequat enim eros eu augue. In odio dolor, scelerisque a lobortis porttitor, commodo ut lacus. Maecenas sit amet diam nec tellus accumsan bibendum. Praesent in turpis velit, malesuada commodo sapien. Nunc ornare urna enim. Sed at diam non metus porttitor suscipit. Aliquam erat volutpat. Duis aliquet magna in mauris semper placerat. Ut eget quam orci. Ut egestas, dolor at dapibus accumsan, leo nibh egestas urna, ac consectetur dui odio quis eros. Nam libero dolor, lacinia eget imperdiet non, malesuada vehicula diam. Etiam id ipsum eget turpis consectetur tristique id at ante. Vivamus blandit nunc eu nisl varius sed accumsan odio molestie.

Chapter 5

Handling Citations

BibTEX can be used to handle all your bibliographic needs. Simply add references to the file `ref.bib` and BibTEX will take care of the rest. An example of a BibTEX book, conference paper and journal article are given in the sample `ref.bib` file. Many online journals have links to BibTEX citations that you can download and incorporate into the `ref.bib` file.

The order of the fields is unimportant. BibTEX will display them in the correct order when constructing your bibliography. Also note that you can specify information about a reference that may not even be included in the actual bibliography. For example, the ISBN field is not required by the bibliography, but you can, if you want, put the ISBN to the BibTEX entry.

We can cite a journal article [**?**] and a conference paper [**?**] in the same way as a book citation. More information can be found in [**?**].

Chapter 6

Conclusions

If a vulnerability in an application is exploited and malicious data injected into the system, Capsicum mitigates the spread of the malicious data by con

fining them to the affected process, since an application running in Capsicum sand- boxed mode is compartmentalized into processes and each process sandboxed.

However, an application running in Capsicum sandboxed capability mode is forbidden to access global OS namespaces such as File system, Process IDs, IPC namespaces. The application also has a restricted access to system calls while access to system calls that involves global namespace access is forbidden. In other words, for Capsicum to contain the damage exploit of a vulnerability in an application can cause, the application has to give up its right to perform certain operations.

Applications running in Capsicum capability mode can acquire Capsicum capability rights, and Libpreopen makes it possible for such applications to request system call operations which the applications have the Capsicum capability rights for and have

Libpreopen perform this system call operations with Libpreopen's version of LibC functions.

6.1 Evaluation

Bibliography

[1] A. Ansari. Heap spraying. `https://www.exploit-db.com/docs/31019.pdf`. Accessed December 16, 2017.

[2] R. N. M. J. A. B. L. K. Kennaway. Capsicum:practical capabilities for unix. *Proceedings of the 19th USENIX Security Symposium*, (3):1–17, 2011.

[3] P. Lonescu. The 10 most common application attacks in action. `https://securityintelligence.com/the-10-most-common-application-attacks-in-action/`. Accessed December 15, 2017.

[4] Veracode. What is a buffer overflow learn about buffer overrun vulnerabilities exploits and attacks. `http://https://www.veracode.com/security/buffer-overflow/`. Accessed December 15, 2017.

Appendix A

Appendix title

This is Appendix A.

You can have additional appendices too (*e.g.*, `apdxb.tex`, `apdxc.tex`, *etc.*). If you don't need any appendices, delete the appendix related lines from `thesis.tex` and the file names from `Makefile`.