

Windows Packer/Loader

Mitchell Palmer   
*Cyber Security Engineering*  
*George Mason University*Fairfax, VA, USA  
mpalmer7@gmu.edu

Andrew Chapin  
*Cyber Security Engineering*  
*George Mason University*Fairfax, VA, USA  
achapin2@gmu.eduCarl Bai  
*Cyber Security Engineering*  
*George Mason University*Fairfax, VA, USA  
cbai2@gmu.edu

Andre Herrera  
*Cyber Security Engineering*  
*George Mason University*Fairfax, VA, USA  
aherre12@gmu.eduHunter Rowlette  
*Cyber Security Engineering*  
*George Mason University*Fairfax, VA, USA  
hrowlet2@gmu.edu

*Abstract*— Our project fulfills a request to produce a software toolkit that allows for remote code execution completely in RAM and file transfer via a service running on a remote host. The goal of our stakeholder, Lockheed Martin Corp. (LM), is for our research to identify a unique way to accomplish this task. The following requirements were provided by LM: (1) the toolkit must be comprised of two separate executables – a “packer” and a “loader”; (2) the “packer” runs locally on Linux, compresses, then encrypts with AES via a user-provided password before sending data to remote hosts; (3) the “loader” runs on a Windows remote host as a service, receives incoming packed data, decrypts/decompresses, and executes any PE files entirely in RAM (i.e. without touching disk). Other loader operating systems were desired. We delivered. A four-part concept of operations was established: (1) a user selects a data block (e.g. executable file) and sends it to the packer where it is packed, (2) the now-packed data is sent over the internet to the remote host, (3) the remote host receives the packed data with the running loader service, (4) the loader decrypts the data block and will either run it in RAM or make it available on the Disk. Specifically, a CLI was built for the packer for user interaction, and a heartbeat process for the loader was designed in order to communicate uptime and availability of remote hosts to the user. Our toolkit, written in C++, implements the desired objectives of our stakeholder. We utilized libssh, filesystem, miniz, etc. libraries to accomplish the objectives. Finally, quality assurance was established through integration and unit tests of the toolkit. The software was then handed over to LM for confirmation and testing in their environment. Alterations were made as requested and the final product was shipped. This paper provides an in-depth analysis of our product and our research into similar products and methodologies to our solution.

Keywords— cybersecurity, remote code execution, CLI

# Introduction

There is no best way to covertly and securely deliver a binary file to a remote computer system and have it executed without ever existing on the hard disk. Many ways will work - conditionally. Code that is functional for one operating system (OS) may not work on a different OS. Personal security products (PSPs) may catch one method of code execution, but not others. These factors depend on the security and configuration of the target remote host (RH). Networking factors can play a role, too. Firewall restrictions, IDS/IPS, and network segmentation all effect the ability to deliver code. The method for subvert and secure network delivery must also be chosen wisely and is target specific. Our stakeholder, Lockheed Martin, asked us to tackle this research question and deliver a two-part software product with specific conditions. The below paper explores the various methods for (1) subvert and secure network delivery of a payload and (2) remote code execution completely in memory; and a description of our product and how it achieves this task.

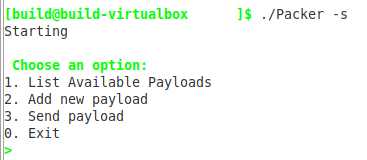


Fig 1. Packer selection menu

# Remote Code Delivery

## SSH Port Forwarding

For delivery of the file we’ve decided upon using SSH port forwarding. We set up an SSH server on the Windows Loader side that the Linux Loader connects to. We decided to use SSH to help manage the key store. We utilized the functions included in libssh to create the SSH session between the Packer and Loader and then created a second channel for the port forwarding. This forward any communication we wanted between the ports we assigned that packer and loader. Doing this allowed us to write the file to the channel and send it securely and discretely between the two.

## Another subsection

More stuff

# Code Execution in Memory

Stuff

## Benefits of Execution in Memory

Code execution that never touches the Disk has a number of advantages. It is much harder to trace back and analyze a program that has only ever existed in volatile memory which is wiped after the system is powered off.

## More subsections

More stuff

# Product Description

Stuff

## Future Development

Although our tool functions as both the downloader and loader for the malware, traditionally a packer/loader tool works in junction with a downloader. The malware is packed into a portable executable as an encrypted/compressed blob with a prepended loader stub designed to unpack it. This packed binary is then sent to the remote target and staged by a downloader or stager (the initial payload executed on the remote system). Once on the remote system, the downloader executes the loader stub on the packed binary to unencrypt/decompress the binary and then pass execution back to the original malware. Both the networking and packing of malware are complicated tasks that must consider changing security technologies and the state of the target operating system architecture. Developing a universal solution to perform all tasks perfectly is practically impossible and instead development should focus on small pieces of each problem. An ideal future design for an effective packer/loader solution would revolve around modular toolkits that perform small tasks but very effectively, that can then be combined and interchanged when needed to provide an extensible and adaptable form of deployment for malware.

1. Table Type Styles

| Table Head | Table Column Head | | |
| --- | --- | --- | --- |
| Table column subhead | Subhead | Subhead |
| copy | More table copya |  |  |

1. Sample of a Table footnote. (*Table footnote*)
2. Example of a figure caption. (*figure caption*)

##### Acknowledgment *(Heading 5)*

We would like to provide special thanks to our stakeholder, Lockheed Martin (LM), and our primary point of contact at LM: Allison Elfring. Ms. Elfring was incredibly knowledgably, professional, and helpful as the subject matter expert on the project. We would also like to thank Rock Sabetto for being everything we could have possibly wanted in a faculty mentor and George Mason University as our degree program.

##### References

The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use “Ref. [3]” or “reference [3]” except at the beginning of a sentence: “Reference [3] was the first ...”

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

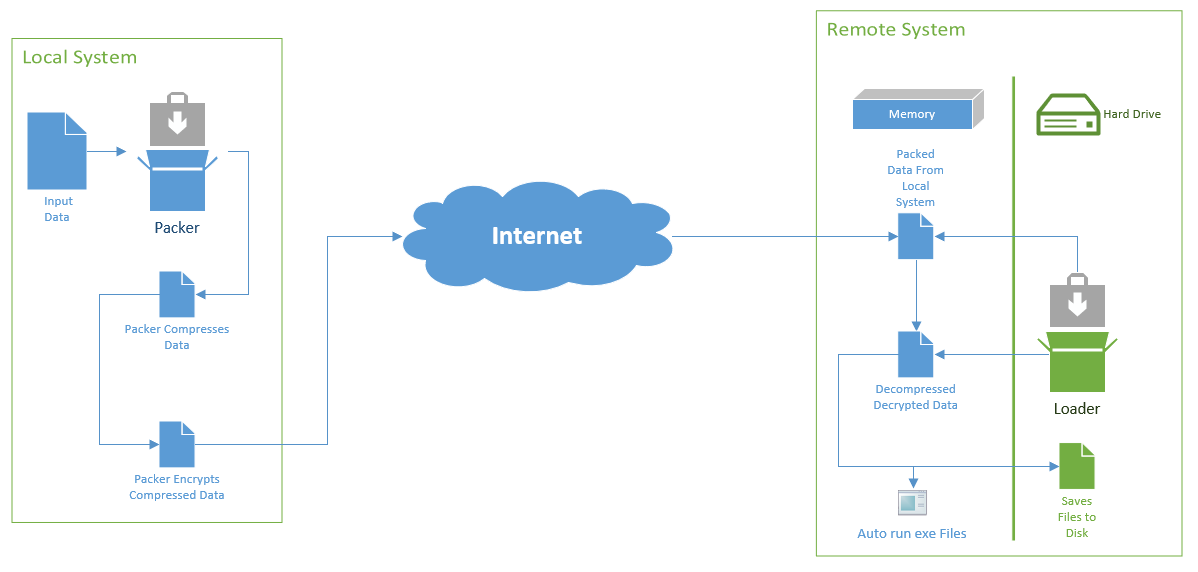
Unless there are six authors or more give all authors’ names; do not use “et al.”. Papers that have not been published, even if they have been submitted for publication, should be cited as “unpublished” [4]. Papers that have been accepted for publication should be cited as “in press” [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

1. “libssh 0.9.3,” libssh. [Online]. Available: http://api.libssh.org/stable/. [Accessed: 16-Mar-2020].

[2] R. Geldreich, “richgel999/miniz,” *Github*, 09-Mar-2020. [Online]. Available: https://github.com/richgel999/miniz. [Accessed: 16-Mar-2020].

[3] “Filesystem library,” *cppreference.com*, 15-Jun-2018. [Online]. Available: https://en.cppreference.com/w/cpp/experimental/fs. [Accessed: 16-Mar-2020].



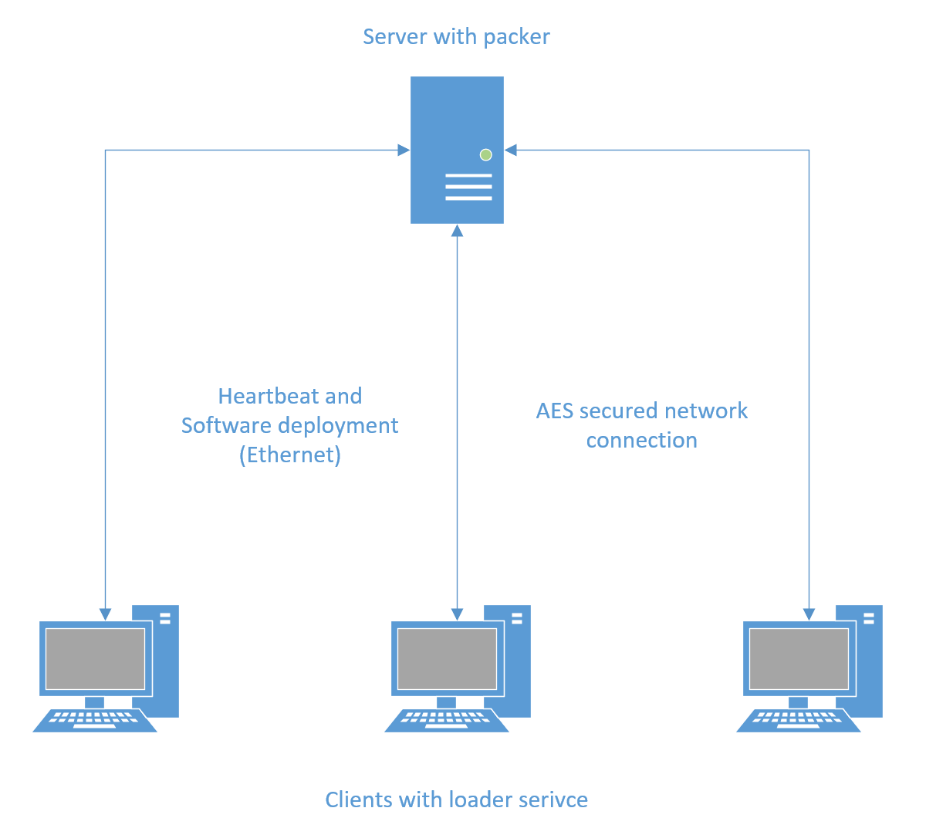


Figure - Architecture Diagram

We suggest that you use a text box to insert a graphic (which is ideally a 300 dpi TIFF or EPS file, with all fonts embedded) because, in an MSW document, this method is somewhat more stable than directly inserting a picture.

To have non-visible rules on your frame, use the MSWord “Format” pull-down menu, select Text Box > Colors and Lines to choose No Fill and No Line.