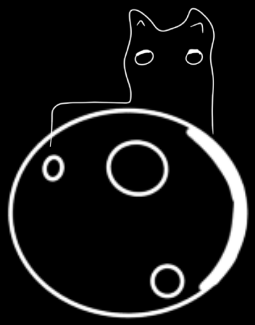


SkyPenguinLabs

Samples

Proper Offensive Tooling & Development Introduction

SPL-OPRGC1

Powered by SkyPenguinLabs

Introduction

> Proper Offensive Tooling and Development Introduction

In today's world of reverse engineering and cybersecurity, the need for hackers to understand what they are breaking becomes more and more important as time goes on. One of my favorite examples of this is checking out the shift between web2.0 and web3.0 and how threat adversaries have taken advantage of this shift to obscure malicious activity on blockchains, which requires a large fundamental knowledge base of today's landscape.

I have mentioned in previous lessons related to cybersecurity or reverse engineering as well, that one of the parts people often miss is how the development shifts change how we operate.

This also includes the tools we employ day-to-day.

So, its important to stay ahead of the game. One of the best ways to keep things fresh is by learning some form of handy development. And what better way to mash offensive security knowledge to develop them?

If you could not tell, in this lesson, we will be exploring how offensive security researchers, reverse engineers, and really anybody can take knowledge applied to the security space to design and create proper offensive frameworks or utilities to:

- Aid them in personal situations
- Expand their understanding of a flaw or exploit
- Expand their thinking skills and perspectives, getting a new angle on how logic gets thought out during the reverse process
- And in most cases, progress any further in a system.

When we say we are 'developing offensive tools,' we are referring to engineering frameworks by applying general security knowledge to create tools that attackers can use to advance their objectives. While not developing a tool today, we are going to be introducing you to what exactly this area is and how it touches the normal spectrum of cyber + where it can take you!

The next page describes the lessons table of contents.

Powered by SkyPenguinLabs

Table of Contents (ToC)

> Lessons outline and details

This lesson contains sections that make up theoretical content on engineering, thinking out, and building offensive tools for security applications.

- **Section 0x00 |** Prerequisites - *(What you may need to know before reading)*
- **Section 0x01** | What/How/Why are they - (A section which covers what we mean when we categorize 'offensive tool', how they work, and what it means to be an offensive tool developer? Is it ethical? What are the grey areas?)
- **Section 0x02** | Marketing & Social Media - (A section which introduces you to the shitshow that marketing and social media have created for any offensive tool developed to serve freelance testers or the public through FOSS. It also covers common misconceptions people have about this, calling it useless, and whatnot. Which gets explored in further thought.
- **Section 0x03** | Differences -The most MINOR differences when it comes to traditional software engineering & engineering frameworks for offensive operations)
- **Section 0x04** | Concluding & Resources - (Concludes all of what you learned and sends you off with some extra resources)

Section 0x00

> Prerequisites

Before we begin this lesson, it's important to note that some engineering experience is expected. The core idea here is to take concepts and techniques from offensive security, such as exploit development, vulnerability scanning, reconnaissance, and more, and apply them within a software engineering context.

The ultimate goal is to design a tool or program that automates or assists with tasks typically encountered in these security scenarios. The experience required for understanding can be easily understood in this context.

Section 0x01

> What/How/Why are they

I feel like every single time I say ‘offensive tool development,’ it can be very easily misinterpreted. So I'd rather first explain what I mean when I say ‘*offensive tool*’.

Directly speaking, offensive tools are types of software apps specifically designed for scenarios that can be deployed by offensive operators, such as penetration testers, to attack or gain information on applications. Think of basic tools like your everyday exploit-suggester. Tools like this are specifically designed to be used in offensive security contexts.

This is all that it takes to define what an offensive security tool is. However, when many people hear offensive, malware may be the first thing that comes to mind. While malware is definitely in the category of ‘offensive utilities’ used by threats, there are other forms of software we are talking about. One of the most commonly talked about and known categories of ‘offensive tools’ often talked about by people who engage in ‘red team’ activities is known as..

Post Exploitation Frameworks

Post-exploitation frameworks are used by security teams, both private and government, all around the world. As the name implies, these types of frameworks are used when attackers already have a good point of leverage on the system, and have gotten past specific layers of the system and gained full command execution.

Whether the tool is used for malicious or non-malicious purposes depends on the client. Whether or not it's ethical, is- well, up for determination depending on each different tool you come across. If a firm comes across a client who is intending to build frameworks for purely malicious purposes, such as a post-exploitation framework, the typical route is to ignore this client and avoid business as engaging in the development of programs or platforms purely with the intent of being used for malicious scenarios (*depending on the contractor, level of work, etc.*) it is illegal in most places of the world that are civilized, including where these courses are primarily sold (*The United States of America*).

The purpose behind the tool is pretty important to pay attention to. This usually tells you how the tool would have been developed. For example, if you notice that someone is spamming Telegram links, to which you use a sock puppet + a nice VPN to join the group, and notice *'oh fuck, bro is over here releasing Windows C2 frameworks'*

You could probably see that the tool may or may not use the Windows API, and may most likely have 1 or 2 bypasses that make it a little bit more persistent. Either way, the point is that paying attention to the meaning behind the tool is a little bit more important than a lot of people make it out to be.

If you decide to spin up an offensive framework yourself, make sure that it is used only in authorized scenarios. For example: your home lab, not your friends' RPI, hooked up to the local WiFi network. And I also feel that it is important to know when you are a skid trying to build DDoS frameworks or a person who genuinely wants to write and engineer an exploitation framework for a security testing lab, a legitimate offensive tool.

This is the kind of stuff we would have to be considering. I am sure a LOT of people probably read this or picked it up expecting me to do just that, but I do not want to promote the development of those types of frameworks at all. These are the exact lines and grey areas where you step into which can flip to you crossing the line FAST. So its best to just avoid it.

- **Note**: Stress testers, often known as DDoS tools, ping flood tools, etc, in SOME environments, have a legitimate use case, but they are often used to check the existence of rate limiting, resource protection, resource consumption, and more. NOT to just knock people's websites offline. While they are used like this, stress testing tools can also be used to find or uncover bugs in software that processes large streams of data. Consider an application that implements its network protocol that builds on top of TCP. A custom stress-testing tool would be useful to see how much the network can take being slammed at it at once, what the resource load is on a system, and various other things, as we mentioned above. This saves developers a large amount of time, and saves security researchers more when a system is quick to fail for them.

I mentioned if you were not a skid, you were somebody trying to build something useful, such as an exploitation framework for a security testing lab.

Many companies starting up, security researchers with hardly any funding for enterprise tools, or people wanting to do their compliance testing, may engineer or develop their own security testing tools and frameworks around their needs.

Compliance testing is a huge example with companies like BishopFox. Either they spend the money on other tools, or instead, because of how advanced they are, and with how much they have released, such as sliver, they most likely would build a framework to make sure that their compliance testing can be done in an automated fashion, without third party tooling or intervention, and without having to worry about codebases or services they can't control.

For many organizations, depending on the tasks at hand) Developing frameworks, scripts, or services is important for security teams to save time during engagements.

On top of engagements, some security teams may notice a pattern amongst clients and build entirely separate pipelines around their processes to work with a specific type of client differently. However, depending on the company, this approach may vary from place to place.

If you are wondering why I did not focus on many specifics or tools here that are used for offensive security operations, the above is why. It's very hard to cover the specifics when there is a lot more diversity in the realm of possible reasons tools can be employed or developed, which ultimately changes the way they are designed, engineered, developed, and more. Generally speaking, the idea of finding people who can do this involves finding an engineer who has a good security background. That is it. Any good engineer with a security background can easily cross scenarios.

- **A note on investing in them** - Some people find that investing in these types of tools, such as sniper, is worth it. If you find this same conclusion as well, I want to say that it may be worth it, it may definitely be, depending on your scenario. This is why some organizations never ONCE will develop frameworks or tools internally, while others

```
run half and half on third parties and their internal
software. Every company and field is different in this.
```

Now that we have explored the surface details and were able to define what an offensive tool was / offensive tool developer was, we can move on to the next section.

In the next section, you will learn a little bit about the social media marketing behind offensive tools, the types of markets that exist for them, and how offensive frameworks can be repurposed and sold to people who only want to use them for harm. No, this is not some scary, spooky dark web link, so don't start going like 'ohhhhh haxxer clickbaitin'.

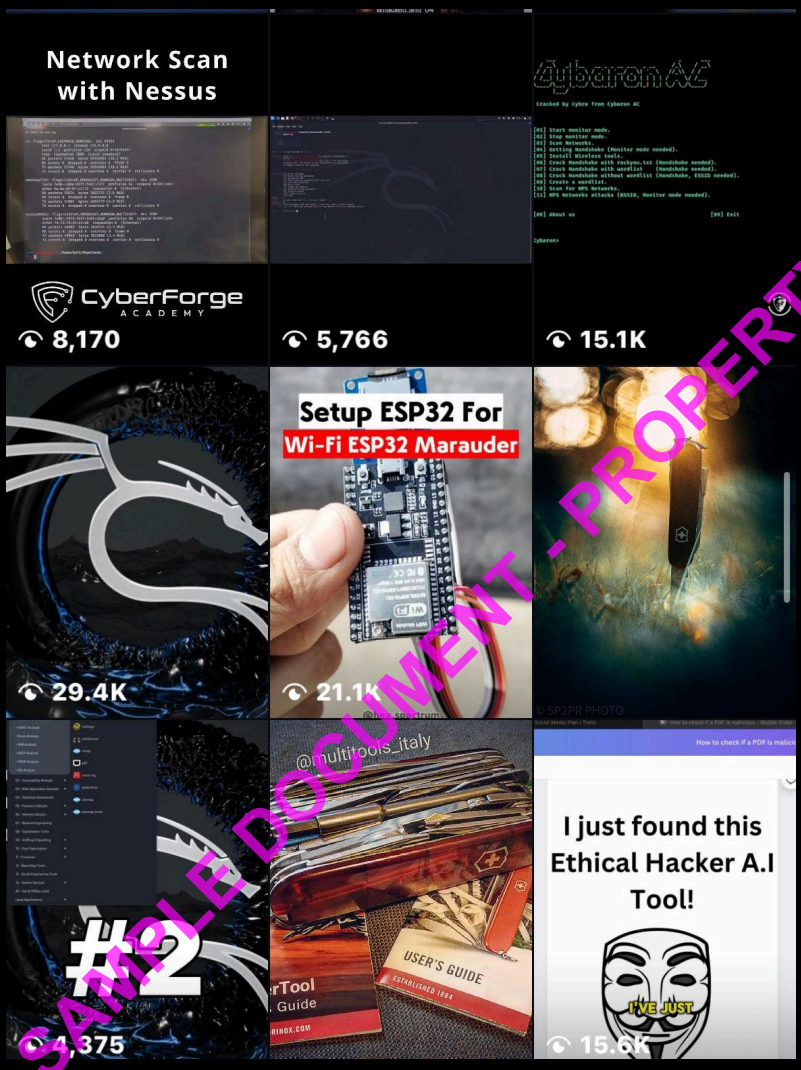
I also wanted to use the second section to explain some misconceptions behind the use of these tools.

Section 0x02

> Marketing and Social Media Trashcan

I know anybody who has been in this field for more than 6 months is already used to the cybersecurity snake oil that exists in this area and we all know this floods the social media as well. Nothing new.

But, I wanted to mention something that I noticed blinds people's perception of what offensive tools are. Here is one good example.



In this image, as you can see, there are multiple videos that have thumbnails that are not really favorable. While some of these tools, such as Nessus, are used in offensive security contexts, Nessus isn't a Red Team tool meant for discovering new exploits. It is designed to look for known vulnerabilities that can be exploited by a bad actor. Nessus is huge for compliance as well.

However, the marketing on this page SCREAMS hacking tools. And the people it's geared towards are your generic script kiddies or beginners who want to touch on the WiFi hacking tools, and the very 'leet hacker' marketed kind of devices.

By the way, in no way am I saying getting these devices or engaging with the software that's shared is bad; however, it is the way it gets marketed.

Unfortunately, many people relate images and marketing themes like that with offensive tools when they are not broken down the concept correctly. Additionally, when they do not understand how to apply the concept.

One of the most common forms of people that can carry a bad reputation with this is freelance developers on fiver or upwork. Most people are legit, however, anybody trying to offer cheap work or development for very niche things like wallet drainers, malware, recon frameworks, or something almost AI-generic, then it is not something you want to engage with.

Again, falling back to the firm scenario. Unfortunately, this puts a bad reputation on the people who want to do this for good and do this with a purpose.

So, professionally speaking, how is this applied when the context is good? And contractors are trusted?

In professional business contexts, a company does not really hire people to do this in the way that most people think of 'hiring'. And that is why we say 'applied when the context is good' because it's only applied now and then, which is why it is not a full-time job in the same light that reverse engineering is not always a full-time position, as it comes with other skills in the offensive security skill tree.