0:00

Hello, and welcome to this course and what we're talking about using Python for reconnaissance as part of the pre-att&ck matrix and the miter att&ck framework. Before we get into the demos of how to apply Python to cybersecurity use cases, we're going to start out with an introduction to scapy. Scapy is one of the python libraries designed for working with network traffic and so it's very useful to us in this particular course because we're going to be using it to build custom packets to send out over the network in a later video. But before we do that, we're going to cover some of the core capabilities of scapy so you have some familiarity with them. For this demonstration, we're just going to open up Python and the first step is bringing scapy into our Python environment. What I'm going to do is do from scapy.all import star, and this allows us to have full access to scapy's functionality and phrasing it in this way lets us specify a particular functions or definitions in scapy without having to say scapy dot everything. To start out, let's take a look at scapy's capabilities from a packet reading and understanding perspective. I'm going to use the read pcap or rdpcap function to read in a packet capture file called http dot cap. This packet capture is available on the Wireshark website under sample captures and is designed to just present a simple HTTP connection. Once I've read this inside packet, I have a collection of different packets, and our summary here says that we have 41 TCP packets, two UDP, no ICMP and no others. To start out, let's pull out the very first packet in this capture, and then I want to take a look at its contents using the show function built into scapy. This is really useful because it shows the various layers within this packet and the values associated with them. From the top down, we see that this has an ethernet layer with the MAC addresses specified and it specifies that ethernet layer is wrapping an IPv4 packet. Next layer down is IP we see version 4 right there, and all of the various header information within an IP packet, including the source and destination IP addresses. Inside this IP packet is a TCP packet. We see our source and destination, port numbers and everything else. In this particular packet, we don't have any packet data. This is designed as a syn packet to set up a TCP connection, so it would be responded to with the synack and then respond with an ack to complete the TCP handshake. If we actually want to see something carrying some data, we need to look at the third or the fourth packet in this capture at index three due to zero indexing. If I do p.show right now, we see that what we have here is again, at the top level, an ethernet packet wrapping IP packet, wrapping a TCP packet, and all of the header information, or at least most of it is identical for this packet as the previous, because it's that same connection. However, now we have a raw section hiring the payload or the actual data contained within the TCP packet. In this case, if we take a look at this, we see pretty quickly that this is carrying an HTTP packet. Scapy doesn't automatically unpack HTTP, however, it's an easily readable protocols we see here, we can understand what's going on. It's a GET request for download.html using HTTP version 1.1, going to ethereal.com, which is what Wireshark used to be, user agent data, et cetera. Taking a look at this and showing the show function because very useful if you want to perform traffic analysis in Python. Wireshark has a lot of functionality, very useful tool. However, sometimes you might want to do programmatic viewing or editing a packet and scapy is a great tool for that. To demonstrate what we can do with scapy, for example, I've got my packet P here and say that I wanted to send that packet somewhere else. I could do, p[tcp].dport and let's send it to port 8080 instead of port 80, hit enter and then show the contents of the packet again. We see that we've switched from the main HTTP to the alternative HTTP port. Not all ports have a special definition like this, if I do like 8045 and then show that scapy will show the actual port number. It just has this replacement function to make it easier to deal with common ports. This is the reading and viewing packets over the passive side of using scapy for traffic analysis. However, what we're going to be doing in the next video is actually building packets, and we can do this in scapy as well. Let me define a new packet P here, and as we see here, we need to build a packet up from layers. If we have an IP packet wrapping a TCP packet, and we can do that very easily. The syntax here would be IP/TCP to make a bare bones packet that we have no control over its contents. If I hit enter here and p.show, we see that scapy has automatically generated and the packet and populated it with default fields or the fields that it chooses. In this case we have a source and destination, address of the local hosts, our source port is the FTP data port, and our destination port is HTTP. I could go through here and manually change these fields based off of what I want after I've generated a packet, like change these two, I want to destination port of 35, take a look we see that updates, et cetera, or we can do this in the packet creation stage. Here, I defined a basic one. Say I want to send it to the destination IP address of 8.8.8, and I want to set my destination port to 53. However, I don't really care about what else is contained within this particular packet. I hit enter now and we see that I have invalid syntax because I forgot to specify this as a string rather than an int, but otherwise it parses properly, and p.show, we now see that it's going to domain. I could continue to build on top of this, say I want to make this a UDP packet containing a DNS packet. This is how I do so, hit enter and p.show, and it's created a default DNS packet for us. This video is just intended as a very quick introduction to scapy. There's plenty of reference material out there on how to use scapy, the commands that are available, et cetera. Just wanted to cover some of the fundamentals of scapy use before we dive into using it in earnest in the next video, when we talk about network standing. Thank you.