Merlin for Red Teams

lockboxx.blogspot.com/2018/02/merlin-for-red-teams.html

Go checkout Merlin. It's your classic HTTP beaconing, remote access trojan, but with a few twists that we will cover in this post. If you're new to the project and just want to get started with the server and client, the first post is for you. Starting w/ Merlin is very easy, you can build from source (which I highly suggest), or you can simply download and run one of the release binaries. If your using one of the precompiled servers, the only setup you have to do is create a folder structure for the SSL certs. For starters the project is all built around GoLang, which is my current programming obsession for a number of reasons (natively compiled binaries, writes like an interpreted language). This makes the project really fun to develop on and the project creator, Ne0nd0g, has a small community already growing this project. I also really appreciate the vendoring, which allows one to track the revisions of all the packages, and use static versions of those packages. Versioning is something that is pretty critical when your talking about agent integrity and RCE on clients. The following are some of the things that I think really sets Merlin apart from other agents.

One of my favorite things about Merlin is how versatile the agents are, because they are based on GoLang. I like how Russel really highlights the cross-platform nature of GoLang by

showing all the various systems that Merlin can be cross-compiled for. This means we can use a single code base and put agents on most any machine we encounter, having them all call back to whatever platform we decide to run the listening post (vs only being able to put agents on Windows, or only being able to run listening posts on Linux). This cross-platform nature can also be seen in this default make file, which has an easy mode "make all" to build an agent and server for the three major platforms (OSX, Windows, Linux). The wiki is great for build information. There is also tons of information on the wiki for running a server (aka operating the agents). Merlin also includes some advanced logging, which is really useful for capturing details after an operation has concluded.

I like that the transport protocol is HTTP/2 and modern TLS, I think this gives Merlin both a lot of flexibility and security. The HTTP/S protocol is one of the most readily available outbound ports (tcp/443), as most people need to reach web sites during the day (the port that the protocol uses is configurable as well). The TLS gives us end to end encryption, and the PFS ciphers make it even harder for network security monitors in the middle to decrypt our traffic. Further, the HTTP/2 that is negotiated after the TLS handshake adds an extra layer of obfuscation to the C2 coms, making the traffic a non-human readable protocol. I learned all about HTTP/2 here, as this project was my first introduction to the protocol. Russel also wrote a SANS whitepaper on the HTTP/2 protocol that covers inspecting the protocol in use and detecting its use on the network.

The JavaScript agent is an awesome twist, as it lets the attacker control unforeseen platforms or browsers from the same common listening post or server. One primary end goal of a cross platform implant is a unified management server. A JavaScript agent extends this to any platform that can run JavaScript, albeit more limited than a native implant. Another bonus with the JavaScript agent is you can do things like perform CSRF attacks on other pages that the user may be logged into. You can read more about the JavaScript agents here, and keep in mind these are currently only in the dev branch.

Currently the master branch is a little light in terms of features but the dev branch is already teeming with new features. The project is really well laid out so it’s pretty easy to add new features and if you want to see the things we are adding checkout the dev branch. There are still a bunch of basic features we need to add so feel free to dig into the code and add something! Ne0nd0g is putting extensive planning and foresight into the structure of this project. This will soon surface in the shape of modules, which I'm super excited about! I think this will really open Merlin up to more specialized OS-specific development and cool post exploitation modules. I also enjoy working w/ Russel, he's energetic, appreciates the other developers, and he adheres to some great programming standards. If your looking to learn GoLang, work on a great project, or even learn from some awesome developers, I highly encourage you to mess with the Merlin source. Ultimately, should you decide add to the project, def read the developers guide, as it has many of those great programming standards listed in there.