Extra Credit: The Morris Worm

Description:

Let's change it up this week! Rather than a barrage of new lecture videos and lab challenges, let's take a stroll back through software exploitation history and see if you can apply what you've learned to write exploit(s) for a **real** piece of software from the past. :)

In November of 1988 the internet saw it's first self-propagating worm, The Morris Worm! Written by Robert Morris, this clever little bit of code took advantage of a few different bugs and misconfigurations to spread to several thousand computers. One of those bugs was a buffer overflow vulnerability in the BSD Fingerd service. It never received a CVE (those wouldn't exist for another decade), but to the trained eye it's not terribly difficult to spot in the source code. Here's a copy of that program, can you spot the bug? :)

http://www.retro11.de/ouxr/43bsd/usr/src/etc/fingerd.c.html

On our Labs CTF site you will find three new challenges under the "Extra Credit" category. Each is built from a (very slightly) modified version of this source code. Each contains the same vulnerability. The only difference is their compile options:

- **Morris Worm (easy)** is built much like the original program in 1988 would have been; simple, with no exploit mitigations at all.
- **Morris Worm (medium)** adds NX protection which did not exist at the time. This will make writing an exploit more difficult, but not impossible.
- **Morris Worm (hard)** Is compiled 64-bit, and includes NX, but is missing a few key components you might have wanted to use in your ROP chain. Can you find creative alternative solutions?

Put yourself in the shoes of Robert Morris and try to write exploits capable of running code on the remote machine for each. I'll give 3 points extra credit for each solve, and 1 more bonus point if you solve all three, meaning that you can earn up to 3+3+3+1 = 10 points total here, the equivalent of a full assignment!

Remember to please document your solutions. "Documentation" may be as simple as code comments throughout your script, explaining what each part does, or as fancy as a separate write-up with screenshots, diagrams, and full explanation. Up to you, I'd just like to see your thought process.

Don't forget to submit the flags!

Deliverables:

- For each of the labs listed above, please turn in to the dropbox…
  - Your well documented script which solves the challenge
  - A screenshot of it running, showing that it works.