Using Unikernels to Enhance the Attack-Resistance of Spire, a Network-Attack-Resilient Intrusion-Tolerant SCADA for the Power Grid

Brad Whitehead
Mike Boby

CS3551 – Advanced Topics in Distributed Systems
Class Project Checkpoint #1
March 24, 2020

Revised Project Goals

Convert Spire to self-contained unikernels and demonstrate that:

- They continue to operate correctly and
- They exhibit the increased performance and reduced resource utilization characteristics of unikernel technology
- Spire compromise resistance can be increased by combining polymorphic executables (Multicompiler) with unikernel
- If possible, demonstrate the increased compromise resistance of the unikernel-based Spire (both GCC- and multicompiler-based)

[Red = Changes from original]

Anticipated Project Steps

1) Familiarization with the Spire system (~~obtain and compile the code~~ (Done) and run the supplied benchmarks

2) ~~Research available unikernel libraries and select the most appropriate one~~ (Done)

3) ~~Select an appropriate paper on unikernels and security to present in class~~ (Done)

4) Compile the Spire executables into unikernels

5) Iteratively, make necessary code changes

6) Test and benchmark Spires unikernels using the included benchmark suite

7) Investigate the compromise resistance of the Spire unikernels (this step is dependent on the availability of any existing compromise/penetration tests or test tools)

8) Document the project

9) Prepare and deliver project presentation for class

Accomplishments To-Date

- Met with Dr. Babay to review Spire and recommended configuration
- Set up testbed (OS, accounts, and remote desktop access)
- Created Git repositories in ‘/opt’ for:
  - Spire
  - Multicompiler
  - Hermitux (primary unikernel system candidate)
  - Nanovms (alternative unikernel system candidate)
- Installed known build dependencies
- Configured GNU gold.ld (as opposed to the normal GNU ld) as the default (required by the Multicompiler)
- Installed Docker (for Hermitux and Nanovms build systems)

Challenges and Lessons Learned To-Date

- Secure remote computing is still (too) difficult to configure
  - We live in a cloud computing world, however…
  - Setting up ssh and remote desktop protocol (RDP) is still bizarre and non-intuitive
- Linux build systems for source code are still (too) uncertain
  - Successful outcomes are too dependent on operating system distribution, distribution version, build system version, dependency versions, etc
- Outside events (pandemics) can have unexpected effects on project planning
  - Upside – Boccaccio's Decameron and Chaucer's The Canterbury Tales
    - Common theme – written in self-isolation during plague periods
    - Inspiration to make unikernel, polymorphic Spire a classic best seller ;-)
- Things take longer than expected
  - Definitely not a new lesson :-(

Revised Project Plan

- **Class Week #11 (March 22 – 28)**
  - **Present Status Checkpoint to Class**
  - **Compile Spire system using GCC**
  - **Install KVM and create the VM configuration files for the ~~6~~ 8 VMs we need to test Spire**
  - **Run the Spire benchmark, using recommended configuration**
- **Class Week #12 (March 29 – April 4)**
  - **Compile Hermitux build tools**
  - Link Hermitux and GCC-compiled Spire executables
  - Re-run the benchmark, using these Hermitux unikernel executables
- **Class Week #13 (April 5 – 11)**
  - **Compile Multicompiler**
  - Re-compile Spire using the Multicompiler
  - Re-run the Spire benchmark, using the Multicompiler-compiled executables
- **Class Week #14 (April 12 – 18)**
  - Link Hermitux and Multicompiler-compiled Spire executables into unikernel executables
  - Re-run the benchmark using the Hermitux & Multicompiler unikernels
- **Class Week #15 (April 19 – 23)**
  - Present Finding to Class