The University of North Carolina at Greensboro
CSC 495/680: Trusted Computing
Prof. Stephen R. Tate

Handout 2
September 20, 2010

CSC 495/680 Assignment 1

Due Monday, September 27

The purpose of this assignment is to get you to think about various TPM capabilities, and to become familiar with the lab machines and how the TPM is made available in a typical Linux system. Since we haven't covered programming interfaces to the TPM yet, the hands-on exercises for this assignment are necessarily limited. Due to the light nature of this assignment, it will be graded out of 50 points, and will carry half the weight of the later assignments.

Lab Exercises: Before starting with the lab exercises, review the sheet "Systems for Trusted Computing Assignments" that assigns you specific systems to work on, and describes how to work so as to avoid conflict with other users. I assume you know the basics of working on a Unix/Linux system, but I don't assume any in-depth experience.

Activity 1 (Become familiar with the system): One of the nice things about a Linux system is that it provides access to many system resources and parameters through the filesystem — these look like files, but are actually live data provided by the system. So after logging in to your primary (or backup) system, the first step is to become familiar with the system by exploring some of these system resources. What kind of processor is on this system? You can tell by looking at the file `/proc/cpuinfo` (you can use the "`more` command to page through this information a screen at a time). Make a note of the "model name", and how many processors show up (e.g., a dual core processor will show two processors, and a quad core processor will show four). Next, how much memory is on this system? For this, look at `/proc/meminfo` — it's really just the "MemTotal" figure here that's important.

For this activity, write down the basic CPU and memory information that you discover.

Activity 2 (Finding the TPM device files): Access to different devices is provided via the `/sys` files, so your first task is to find the TPM in this file structure. Each device (such as the TPM) has a directory in the `/sys` subtree, and different files in that directory access different resources of the device. Here's a trick to find the TPM: it is the only device that has PCRs, represented by a file named "`pcrs`", so search for that file name. In particular, try this command and see what the results are:

```
find /sys -name pcrs
```

This will show you the full path to the "pcrs" file, so if you remove the last `/pcrs` part then you have the directory for the TPM device. Unfortunately, this is not the same path on every

system. Next, go into the TPM directory using the `cd` command and see what files are there. Use `cat` or `more` to type out the `caps` file. Can you make any sense out of the "Manufacturer" value? Next, type out the `pcrs` file — this shows the current value of every PCR in the TPM — copy this output (cut and paste) from the terminal window since you will be reporting this as one of the results of this activity.

Note that while the `find` command was used to find the (variable) path to the "real" TPM device, the `/sys` filesystem also provides links between files and directories to make finding the device easier. Explore under `/sys/class/misc` and see if you can find a path to the TPM device directory — some of these are symbolic links, so the regular `find` command doesn't work, and you'll just have to look around until you find the `pcrs` file.

For this activity, report the full path to the TPM directory in the `/sys` filesystem, give the output of the `caps` file (and any interpretation you have of the manufacturer code), and give the full set of PCR values that you copied from your terminal window. Finally, give the full path to the TPM device starting with `/sys/class/...` — identify which of the path components is actually a directory name, and which is a symbolic link.

Activity 3 (Accessing the TPM using the "tpm tools"): There is a set of programs referred to as the "TPM Tools" that provide access to the TPM through programs, rather than having to go directly to the device files. These tools are very rudimentary, and are primarily for doing basic TPM management — taking ownership and that sort of thing. Since the TPMs in the lab are already "owned," you won't be using those functions, but several other commands are useful. Run the "`tpm_version`" command to get some information about the TPM in the system. Note that the "TPM Vendor ID" is a few readable characters — can you figure out who manufactured the TPM in your system? (Hint: the TPMs in the lab were manufactured by ST Microelectronics, Winbond, Atmel, and Infineon.) Do you see any relation between the "TPM Vendor ID" string and the "Manufacturer" code in the previous activity (this code is also the last line of the `tpm_version` output). The other command to try is the `tpm_getpubek` command — try running this and see what happens. It is useful to know that the owner password on all the lab machines is "`SPANOwner`" — does that allow you to access the public part of the endorsement key? Try running `tpm_getpubek` with the `-u` option to see if that helps.

For this activity, give the output of the `tpm_version` command and your guess of the manufacturer (and why you made this guess). Can you make any sense out of the "Manufacturer" code now? Next, report on the results of the `tpm_getpubek` command. Did the owner password work? What does the `-u` option do, and explain why it made a difference (hint: we briefly discussed this in class).

Questions: Answer the following questions regarding basic design principles and capabilities of the TPM.

1. "Secure storage" refers to keys and data stored in a tree with the SRK (Storage Root Key) at the root. Can a non-migratable key be stored under a migratable parent key? Why or why not?

2. Recall that the system integrity measurements, stored in the PCRs, are based on the idea that the initial portion of the BIOS was not modifiable — this is the "Core Root of Trust for Measurement." Describe how the full system can be compromised if this small part of the BIOS *can* be modified.

3. Platform Configuration Registers (PCRs) are designed to hold measurements of system data, but can be used for other purposes as well. For example, you can create a "ratchet" for storing keys as follows (recall that a ratchet is a gear that can turn one direction, but not the other): rather than extend a PCR using a hash of code running in the system, you can extend it by a constant value. It will then go through a predictable sequence of values, but once it leaves a value it cannot be returned to that value unless the system is reset. Now values can be sealed to these PCR values so that they are only available when that PCR value is present — once the ratchet is advanced, the value can no longer be unsealed (until the system is rebooted). This could be used, for example, to make a value accessible to the operating system while it is booting, but then make it completely inaccessible after the system comes completely up. Can you think of a good use for this capability? Note that there is no "right answer" here — I want to see if you can be creative!