Outline for May 24, 2005

Reading: §22.4–22.5, §22.7, §23.1–4

Discussion

A recurring question in computer security is how the discoverer of a vulnerability in a program or computer system should report it to the responsible party (in this context, the vendor of the program or system). The SANS Organization has proposed the following, which they are calling the Fisher Plan. This description is from SANS NewsBites, Vol. 4, Num. 50 (Dec. 12, 2002):

> There would be established a reporting center for new vulnerabilities, either inside the government or outside (that's one of the things that needs to be decided) along with reporting guidelines that required sufficient data to ensure the person doing the reporting has found something real. All reports will be recorded and immediately passed both to the vendor (which may have already received it from the person who found it) and multiple government or government-funded centers of excellence such as CERT/CC and the research group of the National Infrastructure Protection Center.
>
> The centers of excellence would identify how critical the vulnerability might be and would set a priority for correcting the problem. (The scale is yet to be determined but can be modeled after SANS Critical Vulnerability Analysis rating scale or the CERT/CC rating scale or both).
>
> Government officials will monitor the vendor's progress and exert appropriate high-level pressure on the vendors for rapid response of important vulnerabilities.
>
> When a method of eliminating the vulnerability is found, it will be published by the vendor and at the same time, the person or organization that found the vulnerability will be awarded both public recognition and a sum of money which may come from the government or may be provided by the SANS Institute. (Financial remuneration is controversial; your feedback would be appreciated.)

What are the good points about this plan? What are its drawbacks?

Outline

1. Malicious logic
   a. Logic Bombs, Worms (Schoch and Hupp)
2. Ideal: program to detect malicious logic
   a. Can be shown: not possible to be precise in most general case
   b. Can detect all such programs if willing to accept false positives
   c. Can constrain case enough to locate specific malicious logic
   d. Can use:
      i. Type chcking (data vs. instructions)
      ii. Limiting rights (sandboxing)
      iii. Limiting sharing
      iv. Preventing or detecting changes to files
      v. Prevent code from acting beyond specification (proof carrying code)
      vi. Check statistical characteristics of programs (more authors than known, constructs in object files not corresponding to anything in the source)