Semester Project: A Web-Based "Secure" Voting System

Tom Kelliher, CS 325

300 points

1 The Idea

This project is an experiment to study the difficulty of embedding malicious functionality in code such that it cannot be detected. Each group will design and build a voting machine. The machine should tally votes for candidates for various elected offices. The machine should count votes correctly. The machine should also keep an audit log of all of the votes that have been cast, including the time, and the ballot choices. There should be an interface for viewing the audit log.

The ballot you should use is as follows (you pick the names):

1. President: vote for one of three choices.
2. Congress: vote for one of five choices.
3. County Commissioners: vote for three of five choices.

The machines should enforce no overvoting (e.g. voting for two people for President when only one is allowed) and warn voters of undervotes (e.g. not voting for one office, or voting for only one commissioner).

You are free to add other features as you see fit. A few things to consider would be write-in candidates, showing just one contest per page, and a final page to allow the voter to verify their selections and make corrections before casting their ballot.

1.1 The Backdoor

Your system should contain a malicious back door. That is, there should be a way for a voter to secretly perform some unusual action such that they can alter the outcome of the election. At a minimum, the backdoor should enable a voter to bias the election somehow towards a specific presidential candidate. That is, pick a favorite candidate and identify them in advance in your project write-up, and that is the one that your system should favor when the backdoor is enabled. A truly successful attack will keep the audit log consistent with the vote totals. The backdoor should be dormant, that is, should do nothing, until a secret activation event (such as a particular sequence of clicks) activates it.

1.2 Some Ground Rules

Your system will be subjected to an independent testing authority (ITA) for certification. Any attempt to purposefully obfuscate the code would trigger alarms and possibly prison sentences, and at the very least the loss of a sale. So, the system must appear to be as legitimate as possible. So, for example, if you were to run your code through an automated obfuscator, it would never be certified, and you would receive a poor grade.

Use phoenix as your development platform. The user interface will be a simple sequence of Web form pages. Processing will be done via perl CGI scripts. You have full free reign as to how you implement your system's back end. You may use plain ASCII files for data storage or PostgreSQL. Subject to the constraints below that others have to be able to easily build your system from a CD you will provide, you may install and use additional supporting software in your account space.

You may use comments as part of the disguise to socially engineer the ITA. Remember, however, to avoid raising suspicion.

1.3 Due Dates

All parts will be graded.

1. On Wednesday, October 18, you will turn in hard copy of the following design documents:
   (a) a requirements specification,
   (b) diagrams of your user interface(s), and
   (c) a code skeleton.
2. On Monday, November 20, at the beginning of class, you will turn in the following:
   (a) A five page maximum writeup describing in excruciating detail how you implemented the backdoor. Explain both how it is activated and how you hid it. Be sure to indicate which candidate your system is designed to favor.
   (b) Two CDs: one containing everything needed to build a clean version of your system and the second containing everything necessary to build the trapdoor version of your system. For each of your systems, include on the CD an install script that builds the whole system. Also include a detailed README explaining how to use the install script. The idea is that it should only take us a few minutes to get your system up and running. Finally, each CD should include your three design documents.

   That day, each group will have 15–20 minutes to demo their voting system. At that point, you will NOT disclose the backdoor. You will simply show an election and that votes are tallied correctly. You will demonstrate the audit log and how it keeps track of what's going on.

   Shortly after you turn in this part, each group will receive two voting systems on CD. Your job for the remainder of the course will be to try to find the backdoor contained within ONE of the voting systems and identify the malicious behavior. One system you receive will be clean and one will contain a backdoor. You may use any tools you like to find the backdoor.
3. On the day of the final, at the beginning of the meeting, you will turn in a report for each of the systems given to you (three pages max per system), identifying the following:
   (a) How is the back door activated?
   (b) Which candidate(s) does the back door bias favor?
   (c) How was it implemented? Identify as much malicious code as possible.
   (d) Describe any tools you used and/or developed to do the analysis

Each group will have 15–20 minutes to demo the systems it analyzed and report the results of each ITA certification of each system Additionally, show off the back door, if you were able to find it. Show what happens to the audit log and the vote totals. If time permits and you found something really clever of one of the other group's work, discuss it.

1.4 Grading Criteria

1. Quality of initial deliverables (75 points)
   (a) Is your requirements specification complete and well thought-out?
   (b) Is your user interface complete and reasonable?
   (c) Does your code skeleton reflect your requirements specification and is it complete?
2. Quality of the final system (75 points)
   (a) Does the system work when no back door is used?
   (b) Quality of the audit mechanism.
   (c) Does the code appear to be doctored/obfuscated? (How "natural" does it look?)
3. Backdoor (75 points)
   (a) Is it totally obvious? Is it extremely well hidden?
   (b) How much damage does it do? I.e. does the preferred candidate always win, or do they just get some small advantage?
   (c) Is the integrity of the audit log preserved? I.e. does the audit log match the reported outcome of the election?
4. Analysis (75 points)
   (a) How thorough Was your ITA certification analysis?
   (b) Did you find the backdoor you were given?
   (c) How much malicious code were you able to uncover?