Project: e-Voting (2017)

This project will give you a chance to explore both sides of the security issues involved in building and verifying electronic voting systems. In this particular case the scenario is one where voting machines are used at a voting centre and people come into the centre to cast their vote. We will not be considering online voting and the many ways that it can fail.

In particular, here we are considering direct recording voting machines (DRE) that record votes electronically with no paper trail. Electronic voting systems promise greater accuracy of recording, increased accessibility for people with various physical disabilities, and no ambiguity. They also put tremendous amounts of trust in the computer to correctly record the votes - if the computer system is wrong, there is no written backup to fall back on. Therefore, security flaws in DRE systems can be called, without exaggeration, threats to democracy.

You are permitted to work in pairs but must submit your own individual reports.

You are not allowed to obfuscate the **entire** codebase.

You will demonstrate your hack to Ian/Masood during the study break.

1. Premise

You are the development team for Hack-A-Vote Election Systems, a major election vendor that has contracts with several U.S. states and counties to supply voting machines. In the spirit of the recently signed TTPA treaty, New Zealand has decided to adopt these machines for the next New Zealand elections even though it has required us to make some minor changes to our Government structure to conform to the software.

Unfortunately for democracy, you are of questionable moral character, and you've been offered a large sum of money to sabotage Hack-A-Vote's flagship election product.

2. Assignment

There are any number of ways you could use your position as the supplier of voting machine technology to introduce problems into an election. The obvious thing to do is steal the election by modifying your machine to occasionally change votes to a preferred candidate. There are other nasty things you might do, though -- if all of your machines produce uncertain or garbage results, you could cause electoral chaos. You might try to selectively disenfranchise some group of voters you don't like.

Be creative; find a way to modify the machines that will create problems of your choice in the electoral system.

You'll implement your changes against Hack-a-Vote's current product. The user interface has already passed approval testing, **so don't modify what is seen by an ordinary voter**.

Note that it's important to keep your changes small in scope and hard to detect via a source code inspection, because later, your classmates will be scrutinizing your codebase trying to figure out what you did. Cleverness in hiding your changes is therefore crucial to your success.

You can introduce multiple backdoors in you wish, the tradeoff though is that if we find any suspicious backdooring we will reject your code and you might delay the election but you won't subvert it. This means one trigger and one backdoor that are so reliably coded that you guarantee you can demonstrate them to working to us is the best strategy.

You might want to spread your changes throughout the code base and also don't forget that it's not just the code that you might modify. It is also permissible to have multiple backdoors inserted into the system.

A key thing is that you only have to worry about your changes being spotted before the election; afterward, your benefactors have arranged for you to be flown to a location with great weather and no extradition treaties with the New Zealand Government.

Nonetheless the better backdoors will subtly but definitely influence who wins the election rather than simply cause a denial-of-service. You can implement both but you must trade-off power against increased chance of detection by the auditors in the next phase.

As a general guideline, I would expect there to be no more than 150 lines modified or added.

3. Documenting the System

You paymasters want you to document your work as well as provide a demonstration (you will do this to them during the break) before they are willing to hand over your airline tickets and money in untraceable bills.

You should write short report (maximum of five pages) covering the following:

3.1 Purpose

Which of these purposes does you backdoor implement?

- *direct vote-manipulation hack*
- *attacks aimed at breaking the authentication mechanism for PINs or administrative access*
- *hacks directed at defeating voter anonymity (this allow vote bribing)*
- *denial-of-service that is not triggered until the last moments of the election or only discovered after the election*

3.2 Trigger

Describe the conditions for enabling or activating the backdoor. Length is going to be modification-specific. For example, you may need to provide command line switches and explain what each one does.

You can assume that either your paymasters can bribe people running the machines to make small changes locally, run programs with extra command line options or simply *send one of their agents to each of the locations while pretending to be a legitimate voter.*

3.3 Design and Implementation

Identify the key parts of the code base that have been changed and add a commentary.

I would expect maybe a couple of paragraphs giving an overview of code structure, indication of how many lines have been changed and a discussion of the general approach that connects the purpose of the backdoor with the changes to the code (for example, if the desired outcome is for Party Z to always have the majority votes you will would need to explain how your changes modify the results).

You should include a walkthrough of the key parts of the code with differences from the original and modified code highlighted.

3.4 Self-Assessment

Briefly self-assess how well the backdoor is hidden (in terms of design and implementation) from the voter using the system and from a potential auditor.

Briefly self-assess whether your implementation is one based upon the article about Hack-a-Vote or somehow extends it and/or goes beyond what was described.

HINT. Consider the whole system when hiding your hack.

4. Assessment

The criteria for comparing the submissions are:

a. Creativity of solution (for example, an obvious inclusion of IF MY_CANDIDATE GOTO WIN would rank lower than one involving multiple components each subtly changed as would a simple implementation of the simple hack from the article).
b. How well does it work (you get to demonstrate it to me during the break, you can get partial credit if it doesn't work btw).
c. Ability to communicate what was done (report reads well, covers all required aspects and is spell checked).

The overall result will be a letter grade based upon the criteria above.

You should submit only ONE backdoor, your best most subtle one.

In general hacks other than direct vote rigging would gain a better grade than a less obvious hack.

5. What to Submit

You should submit **two** files.

The first is a file called **yourname-vote.tar.gz** which represents your modified version of the Hack-a-Vote system, where yourname is your username (you may use ZIP instead of tar/gzip. Do not use anything bizarre and proprietary like StuffIt or WinRAR; your peers will hate you.)

The second is a report in PDF format that describes the nefarious vulnerabilities that you've introduced, and why. Note that an outline is available for this that indicates what needs to be submitted.

Please keep your introduced vulnerabilities and associated code modifications private until after the end of Phase Two. You don't know which group is going to get your code to analyze, and the less they know, the better off you are.

APPENDIX A. Installing Hack-a-Vote

I have successfully run this code under JDK 1.8 on School systems.

When giving us the code make sure that it works with JDK 1.8 and you must assume we will rebuild the code each time.

Move the **hackavote.zip** to its own directory and unzip it.

Read the README, it tells you how to use the system. The README provides a good explanation of how to use system. However, the short version is:

- The build system to use is ant, you should use **ant clean** to remove old compiled classes and **ant** to compile the code (ant is installed on the School systems).
- The quick and dirty approach is to trun all code from the **build** directory, run **java Console** to display the voter PINs.
- Copy the configuration file form from the parent directory (**cp ../form**) into the build directory because **BallotControl** assumes that the configuration file form is in the same directory as the executable. Now cast a vote by running **java BallotControl**.
- When you have voted early and often, stop the election. You can do this by entering a voter PIN into **BallotControl**, entering the administrator password (**secret**) and clicking on the button election over. The results are written to the directory **ballotbox**.
- The final results are displayed on the screen when you end the election, these should match what is recorded in the ballotbox for a valid election.

NOTE THAT THE CODE IS OLD SO YOU WILL BE GET MANY WARNINGS.