README

Metadata: Arbitrary code execution using env-vars on a vulnerability in autograde.sh

Discovery: Reading through autograde.sh it was clear that there were many environment variables used, and because of the hint in the handout regarding that file, I took a closer look at this. Using trial and error, I realized I could create the /tmp/cs666_bin before the file does and change the permissions.

Impact: I compile a go script locally which simply runs and prints out "/home/whoami", and then I move it to /tmp and call it cs666_bin. (This go script is envexploit.go and is compiled by running "make env" in the same directory as the makefile). I them use chmod remove all permissions; i.e. "chmod 000 /tmp/cs666_bin". Then, I handin a blank main.go/KEY. The autograde.sh script builds this to $TMP, then attempts to move this build to /tmp/cs666_bin. Because of the permissions, this fails and nothing is written. From there, the autograde script changes the permissions of /tmp/cs666_bin so that it can be ingested and run, and then the output is printed from the portion of the handin program that writes to the database. The fourth line of this prints the egid, which is 1004, the egid of the TA group, indicating that this go script is run with TA group permissions and thus I could execute arbitrary code as the TA group.

Mitigation: To mitigate this, the script can remove the file stored at $PERM (i.e. /tmp/cs666_bin) before attempting to CAT to it. This could create a race-condition vulnerability, but this could be further prevented by making the name of $PERM somewhat random to prevent a file from being easily created.

Metadata: Data Theft using env-vars and symlinkt on handin.go and main.go

Discovery: In handin.go, I noticed that $PWD was used instead of the function "pwd", and that $PWD is easily changed by me before I run this script. I also played around with the handin script and discovered that if you submit a tar-ed build, you can pass a symlink and it will follow this.

Impact: First, I "cd /course/cs1660", then I set PWD=/home/alice, and from there I run the handin script. This tries to tar everything, and it prints out the contents of many of the subdirectories, including the handin directory, which it is allowed to do because the handin script is run with TA group permissions. This shows the metadata of everyone else's handins. I use the address of Bob's handin to create a symlink to it, and then from there can run cs666_handin from my home directory in the form handin_cs666 <path to bob's tar-ed handi>. I've successfully handed in Bob's handin, which constitutes Data Theft.

Mitigation: For the env-vars fix, the script should use pwd(1) instead of $PWD, since this is not changeable by the user, and the env-vars vulnerability is a relatively powerful metadata extraction. To mitigate the symlinkt, the handin script must parse the path of any tar handins, and verify that they are not within the handin folder (i.e. sanitize inputs).

Metadata: Arbitrary code execution thru path-byp and list-conf on blocklist_path.go and blocklist.go

Discovery: The blocklist attempts to block certain packages for handins, but the handout suggested that it’s possible to circumvent this. I went to TA hours for help brainstorming alternate ways to write packages, and it was suggested that I use a longer path beginning with “..” to access “os” and “fmt”. I also tried other packages should as “C”, but ultimately thru trial and error decided on these.

Impact: I submit a totally empty KEY file and my main.go file (included) to the handin. My main.go file imports “os” and “fmt” (which are supposed to be illegal packages), and then uses this to print out the output of /home/whoami. This prints 1004, the egid of the TA group, since this Go file is run by the handin script which has TA permissions. Any script run with an egid of the TA group has TA permissions, so in theory I could execute arbitrary code as the TA group.

Mitigation: Instead of simply parsing the inputs and comparing with the blocklist, blocklist.go should compare the location of the package with the address provided by the user; i.e. print the path of the package I provide using “pwd”. Better path sanitization, or a more complete blocklist (i.e. one that includes variants) would prevent this attack.

Metadata: Data modification thru perm-conf on autograde.go

Discovery: I realized that most of the tmp files created by autograde.go can be accessed by multiple threads. Even though it’s tricky to time scripts to run at exactly the same time, I can pause the execution of a thread to mess with the files that have been created. The handout suggested I think about the tmp files, so I used ctrl-Z to stop the handin script after a certain amount of time.

Impact: I run the handin script on a copy of the demo, and I used ctrl-Z to stop the handin script after the handin has been processed and while the autograder is running. At this point, my handin is visible within the tmp directory, and the build is also visible (this is actually data exfiltration, since I should not be able to view this). Because I have permission to modify these files and write/execute permissions for /tmp, I can remove them, causing the autograder to crash. I then use “fg 1” to resume the process and see that the autograder crashes.

Mitigation: All files within the tmp directory should have permissions set so that I cannot modify them. Alice should not have execute permissions on the /tmp directory so that she cannot delete files (which would cause the autograder to crash like I show in my demo).
