Joel van Egmond
10102094

CPSC 525 Final - Secretstash Exploit Writeup

What is secretstash, and what is the goal?

secretstash is a setuid program that stashes secrets for you in my directory where no one (except me!) can see them. Just in case someone does manage to get into my account though, the secrets are also encrypted with a xor stream cipher (variant of the vigenere cipher), using a personalized key for every user. It is important to note that since this is a xor cipher, encryption and decryption are the *same* operation.

secretstash has two modes, stash and pop:

Stash reads a secret from stdin, encrypts it, and stashes it in a folder given by the user's uid, as a file named after the passed in secret name.
Ex: echo -n "joel is cool" | ./secretstash "coolsecret"

Pop retrieves a secret from your uid folder, decrypts it, and prints it to stdout.
Ex: ./secretstash -p "coolsecret"

Your goal is to retrieve, decrypt, and read someone else's secret.

NOTE: For some reason the buffer overflow stopped working if I made secrets bigger than 16 chars (maybe because of some optimization or caching?). I didn't have time to figure out why so unfortunately your secrets have to be a lot shorter than I wanted :(

Vulnerabilities

There are three vulnerabilities in in secretstash that need to be used to accomplish the goal. Here I'll describe them, and then in the exploit guide I'll explain how to use them.

1. There is a buffer overflow on line 82 in the stash function. The programmer has accidentally used the MAX_SECRET_LENGTH constant for the array size, but is reading in MAX_SECRET_BUFF characters with getline(). MAX_SECRET_LENGTH is the number of characters without a null-terminator, whereas MAX_SECRET_BUFF does include a null-terminator. This means a secret that's too long can overflow by one byte into the key array. We're going to use this to stash secrets without encrypting them.
2. There is a formatstring vulnerability on line 113 in the pop function. This can be used to mess up a bunch of things, but we're going to use it to mess with the uid pointer.
3. Our xorcipher encryption algorithm is vulnerable to a chosen plaintext attack. Since the cipher is just xoring the plaintext with the key (which is just repeated for the length of the plaintext), if we can get both a plaintext and a ciphertext, we can xor them together to reveal the key. And that's what we're going to use this vulnerability to do.

Joel van Egmond
10102094

The exploit guide

Here, our friend fakeroot (uid 1284) has stored a secret using secretstash, and we're going to reveal it. It's located at "/home/joel.vanegmond/secretstash/secrets/1284/dontlook" and I'm sure it is full of embarrassing stuff. Unfortunately we're not fakeroot, so we're unable to pop their secrets since we have a different uid and don't have their key. But maybe we can figure it out?

First off, we want to be able to pop someone else's secrets. To do this, we're going to take advantage of the fstring mentioned in vulnerability #2. This fstring prints the name of the secret being popped, so we can pass the malicious string through the args. We're going to use it to alter the uid, since uid is a pointer we can easily do this. With some %p%p%p… adventuring in gdb, we find that we can write a 0 to the uid using '%134$n'. Now we want to convert this to popping fakeroot's "dontlook". Secrets aren't allowed to have the '/' character in them (for linux filesystem reasons) and get truncated before the character if they do, but not before the fstring line! Thus we can pop "dontlook" with the string "dontlook/%134\$n" - except we can't, since this will write 9 to the uid (dontlook/ is 9 chars). We want the uid to be 1284, so "dontlook/%1275c%134\$n" will do the trick.

```
joel.vanegmond@mocha:~/secretstash$ ./secretstash -p "dontlook/%1275c%134\$n"
Happi!hpngkfc4!
```

Well that's a little garbled, but we got something! Looking at line 118, we can see that while we may be popping someone else's secret, we're decrypting it with *our* key. But it isn't useless! If we can figure out our key we could use it to turn this back into a ciphertext with our handy dandy xorcipher helper program (included in the .tar). But how do we figure out our key?

We can figure out our key using the buffer overflow mentioned in vulnerability #1. By stashing a secret that is exactly 16 characters long, getline() will add a null-terminator and overflow it into the key array. The key is turned into a string, so a null-terminator as the first character will make it empty, thus not encrypting the secret and storing it as plaintext. And since encrypting and decrypting are the same operation, when we pop this secret it will "decrypt" the plaintext, giving us the corresponding ciphertext! For this example I used a file with 16 zeros in it.

```
joel.vanegmond@mocha:~/secretstash$ echo -n "0000000000000000" >> zpt.txt
joel.vanegmond@mocha:~/secretstash$ cat zpt.txt | ./secretstash "zero"
zero stashed!
joel.vanegmond@mocha:~/secretstash$ ./secretstash -p "zero"
Popping zero
\Y\_T^_]WU^QF[illegible]U
```

Now is where vulnerability #3 comes in. By xoring our plaintext and ciphertext together (using the handy dandy xorfiles helper - also included in the .tar) we can reveal our key - "lilodnomgenav.le". Yep, looks like the base key is "lilo" (my cat's name), and it's appended to your username ("joel.vanegmond") but reversed. Super secure. We can use our newfound knowledge to turn that garbled secret from earlier back into the dontlook ciphertext (also using the xorcipher helper).

```
joel.vanegmond@mocha:~/secretstash$ ./secretstash -p "zero" >> zct.txt
Popping zero

joel.vanegmond@mocha:~/secretstash$ ./xorfiles zpt.txt zct.txt
lilodnomgenav.le
```

Final step: we know our key and how it was generated, so with some simple pattern recognition we can figure out that fakeroot's key is "lilotoorekaf". Use that to decrypt our ciphertext from the previous step (again using xorcipher), and voila, we have fakeroot's deepest darkest secret.

```
joel.vanegmond@mocha:~/secretstash$ cat garbled.txt
Happi!hpngkfc4!joel.vanegmond@mocha:~/secretstash$
joel.vanegmond@mocha:~/secretstash$ cat garbled.txt | ./xorcipher "lilodnomgenav
.leoj" | ./xorcipher "lilotoorekaf"

Happy holidays!
```