ELEC5616 AS3

Yuming JIANG 460444981

1.2 iCubeKinect

⑴ We think it is not possible to use the symmetric cipher. Because it is very difficult for iCubeKinect to give the key to its user securely. And there will also be too difficult to handle a lot of keys if there is a large user base.

With using asymmetric cipher, iCubeKinect can simply hard coded the public key into the DVD-Reading-Machine or just store the public key in the cloud. When user want to verify their DVD games, they can get the public key very easily.

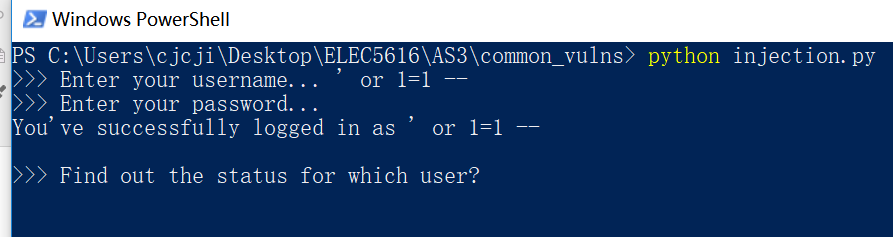
⑵ We found that the cert is stored in the DVD. Anyone can get the cert from buying one DVD from iCubeKinect. With this cert, the hacker can store anything in one DVD and put the same cert into that fake DVD. With that correct cert, user can not find that the DVD is not from iCubeKinect.

⑶ We think the “content\_hash” can not be made static in the cert file stored in the DVD. The DVD-Reading-Machine should generate the “content\_hash” with all of the content stored in DVD when user play the DVD with the machine. With this, the hacker can not change the content in the DVD easily even the hacker can get the correct cert from the DVD.

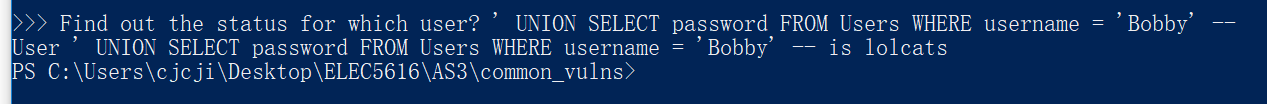
We think the security vulnerability can be less with the using SHA-512. Because, with MD5, the hacker can brute-force get the same MD5 hash result with hacker’s own contents as the MD5 result with iCubeKinect’s content. If change MD5 to SHA-512, the hacker will pay much more to find the correct hash result.

2. SQL Exploits

⑴ The SQL statement to check the username and the password is: *SELECT 1 FROM Users WHERE username = '%s' AND password = '%s'.* In SQL, all the things after “--” will be seemed as comments. So the hacker can type in the username as “’ *or 1=1 --* ” and type in anything for the password place. The final SQL statement is: *SELECT 1 FROM Users WHERE username = ‘ ‘ or 1=1 -- ‘ AND password = ‘ ‘*. “*--‘ AND password = ‘ ‘*” becomes the comment. And the where check statement “*username = ‘ ‘ or 1=1*” is always true. So the hacker can log in without knowing the username and the password.



⑵ The status query is *SELECT status FROM Users WHERE username = ‘%s’.* The hacker can type in “’ UNION SELECT password FROM Users WHERE username = ‘Bobby’ --”. So the statement becomes *SELECT status FROM Users WHERE username = ‘’ UNION SELECT password FROM Users WHERE username = ‘Bobby’ --‘.* The statement after the operator UNION will also be executed, because of this, the password for Bobby is shown.



⑶ This is not a difficult security problem to fix. It is so common because a lot of people who design the application do not know the knowledge about SQL and Database system.

There are four main ways to prevent this type of attack.

① When design the application, use parameterized statements to write all the query statements.

② Just escape all the characters that have a special meaning in SQL. For example, a single quote in a parameter must be replaced by two single quotes.

③ The developer can fix the input parameter to some strict pattern. If the user’s input does not follow the pattern, the query will be denied.

④ The developer can also limit the permissions on the database login by the application to only what is needed. So some core tables can not be accessed by the hacker.

⑷ I think SQL injection vulnerability is more severe than buffer over flow exploit. Because the developer who is dealing with the buffer over flow exploit always have the knowledge of c. They can handle the problem by themselves. However, when developers dealing with the SQL injection, the developers may only know javascript and php and have no idea about SQL and database. They should ask other experts in database field to help them solve the SQL injection problem.