Quiz: security-fouten

```
#include <stdio.h>
#include <string.h>

int main(void)
{
    char buff[15];
    int pass = 0;

    printf("\n Enter the password : \n");
    gets(buff);

    if(strcmp(buff, "thegeekstuff"))
    {
        printf ("\n Wrong Password \n");
    }
    else
    {
        printf ("\n Correct Password \n");
        pass = 1;
    }

    if(pass)
    {
       /* Now Give root or admin rights to user*/
        printf ("\n Root privileges given to the user \n");
    }

    return 0;
}
```

- Welke programmeer-taal?
- Welke fout?

http://www.thegeekstuff.com/2013/06/buffer-overflow/

Antwoord: buffer overflow

Here is an example :

```
$ ./bfrovrflw

 Enter the password :
hhhhhhhhhhhhhhhhhhhh

 Wrong Password

 Root privileges given to the user
```

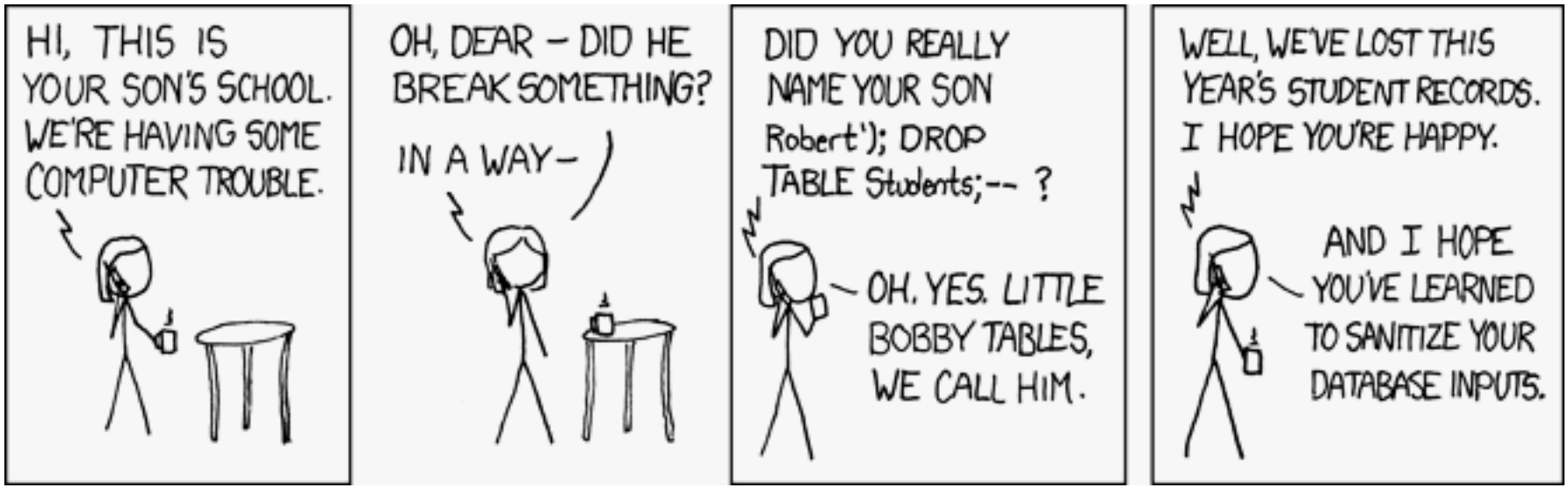
In the above example, even after entering a wrong password, the program worked as if you gave the correct password.

There is a logic behind the output above. What attacker did was, he/she supplied an input of length greater than what buffer can hold and at a particular length of input the buffer overflow so took place that it overwrote the memory of integer ‘pass’. So despite of a wrong password, the value of ‘pass’ became non zero and hence root privileges were granted to an attacker.

Vraag 2: welke vulnerability?

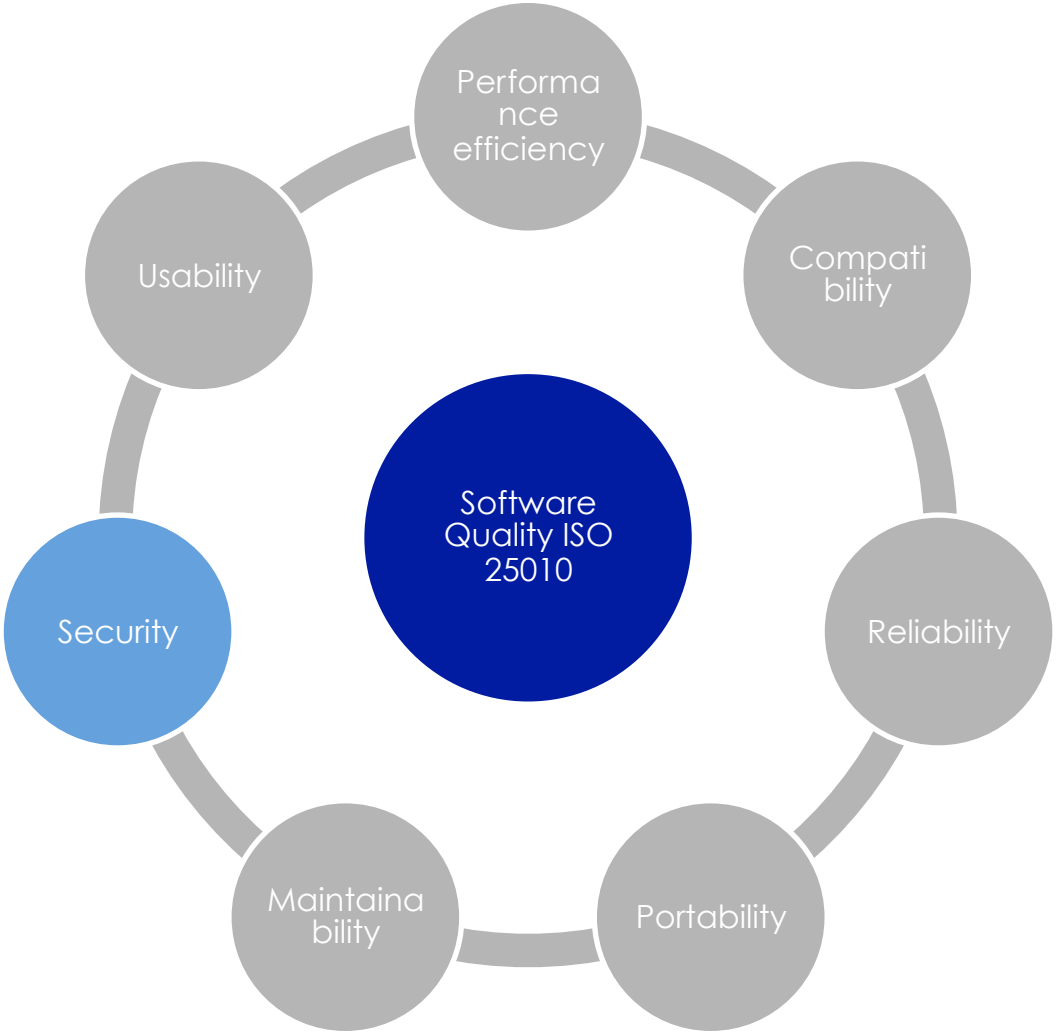
statement := "**SELECT** * **FROM** userinfo
WHERE id =" + input1 + ";"

Voorbeeld exploit




303
Exterion
if (request.getParameter("admin") != null) {
writer.write(" Welcome, administrator. ");
} else {
String username = request.getParameter("uname");
String pwd = request.getParameter("pwd");
String query = "select * from user where username='" + username + "' and password='" + pwd + "'";
ResultSet rs = connection.createStatement().executeQuery(query);
if (rs.next()) {
writer.write(" Welcome, " + request.getParameter("uname")+" ");
} else {
writer.write(" Failed login for username " + request.getParameter("uname"));
}
}
Vind jij alle OWASP top 10 kwetsbaarheden
in deze code voordat je trein vertrekt?

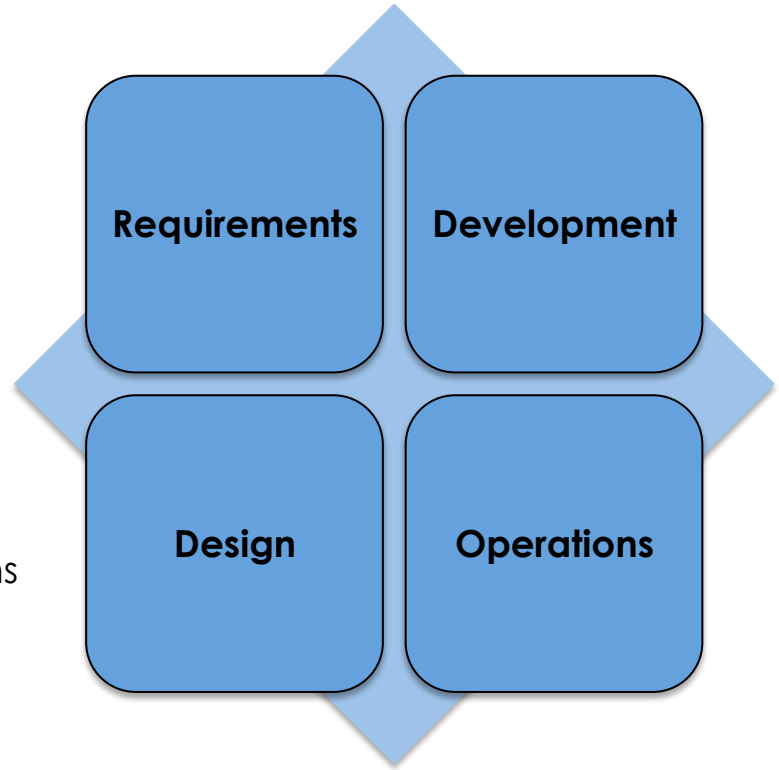
Security is een van de nonfunctionals uit ISO-standaard



Security wordt vooral bepaald door organisatorische aspecten

Requirement factors
Business vision and high level requirements
Business risk analysis

Design factors
Security requirements
Threat surface analysis
Security-by-design solutions



Development factors
Developer and tester training
Development and test process
Development and test tools

Operational factors
Organization security
Infrastructure security
Monitoring and response

Open standaarden voor security zijn en worden ontwikkeld

Open standards for security

- SANS security controls and common vulnerabilities
- Microsoft SDLC practices
- OWASP top 10 vulnerabilities

(Nog?) niet open:

- ISO 27001

OWASP top 10 2013

OWASP top 10 2013

1. Injection
2. Broken authentication and session management
3. Cross site scripting
4. Insecure direct object reference
5. Security misconfiguration
6. Sensitive data exposure
7. Missing function level access control
8. Cross-site request forgery
9. Using known vulnerable components
10. Unvalidated redirects and forwards

SANS security controls

SANS security controls
1: Inventory of Authorized and Unauthorized Devices
2: Inventory of Authorized and Unauthorized Software
3: Secure Configurations for Hardware and Software on Mobile Devices, Laptops, Workstations, and Servers
4: Continuous Vulnerability Assessment and Remediation
5: Malware Defenses
6: Application Software Security
7: Wireless Access Control
8: Data Recovery Capability
9: Security Skills Assessment and Appropriate Training to Fill Gaps
10: Secure Configurations for Network Devices such as Firewalls, Routers, and Switches
11: Limitation and Control of Network Ports, Protocols, and Services
12: Controlled Use of Administrative Privileges
13: Boundary Defense
14: Maintenance, Monitoring, and Analysis of Audit Logs
15: Controlled Access Based on the Need to Know
16: Account Monitoring and Control
17: Data Protection
18: Incident Response and Management
19: Secure Network Engineering
20: Penetration Tests and Red Team Exercises

Example SANS 6: application software security

ID #	Description	Category
CSC 6-1 (NEW)	For all acquired application software, check that the version you are using is still supported by the vendor. If not, update to the most current version and install all relevant patches and vendor security recommendations.	Quick Win
CSC 6-2	Protect web applications by deploying web application firewalls (WAFs) that inspect all traffic flowing to the web application for common web application attacks,	Quick Win
CSC 6-3	For in-house developed software, ensure that explicit error checking is performed and documented for all input, including for size, data type, and acceptable ranges or formats.	Visibility/ Attribution
CSC 6-4	Test in-house-developed and third-party-procured web applications for common security weaknesses using automated remote web application scanners prior to deployment,.	Visibility/ Attribution
CSC 6-5	Do not display system error messages to end-users (output sanitization).	Visibility/ Attribution
CSC 6-6	Maintain separate environments for production and nonproduction systems. Developers should not typically have unmonitored access to production environments.	Visibility/ Attribution
CSC 6-7	Test in-house-developed web and other application software for coding errors and potential vulnerabilities prior to deployment using automated static code analysis software, as well as manual testing and inspection. In particular, input validation and output encoding routines of application software should be reviewed and tested.	Configuration/ Hygiene
CSC 6-8 (NEW)	For acquired application software, examine the product security process of the vendor (history of vulnerabilities, customer notification, patching/remediation) as part of the overall enterprise risk management process.	Configuration/ Hygiene
CSC 6-9	For applications that rely on a database, use standard hardening configuration templates. All systems that are part of critical business processes should also be tested.	Configuration/ Hygiene
CSC 6-10	Ensure that all software development personnel receive training in writing secure code for their specific development environment.	Configuration/ Hygiene
CSC 6-11	For in-house developed applications, ensure that development artifacts (sample data and scripts; unused libraries, components, debug code; or tools) are not included in the deployed software, or accessible in the production environment.	Configuration/ Hygiene

Microsoft Secure Development Life Cycle

Training	Require-ments	Design	Implemen-tation	Verificatio n	Release	Response

Advies

- Bespreek non-functionals van te voren en leg ze vast
- In het ideale geval vraagt de klant om deze zaken. In de praktijk moet IT-leverancier hier een voorstel doen
- Let op dat afspraken eenvoudig meetbaar en testbaar zijn. “Zo goed mogelijk” vraagt om problemen

Recente incidenten

Criminelen stelen wachtwoorden 157.000 klanten webwinkelketen Mapp

Door Arnoud Wokke, donderdag 16 april 2015 14:09, 102 reacties, 13.197 views • Feedback
Submitter: Blue Knight

Webshop-uitbater Mapp heeft alle gebruikerswachtwoorden gereset nadat e-mailadressen en wachtwoorden van 157.000 klanten waren gestolen. Totdat gebruikers een nieuw wachtwoord instellen, zijn accounts bij webwinkels als InternGeheugen.com en MemoryMan.nl onbruikbaar.

Mapp raadt klanten aan om, als zij het wachtwoord dat ze gebruikten bij het account bij Mapp ook elders gebruikten, het wachtwoord ook daar te wijzigen. De gestolen wachtwoorden zijn weliswaar versleuteld, maar de kans bestaat dat de dieven ze kunnen ontsleutelen. "De wachtwoorden waren wel gecodeerd, maar niet gesalt. De kans bestaat dus dat het te decoderen is", aldus Pieter-Paul Tersmette van Mapp tegen Tweakers.

De webwinkelketen heeft zelf klanten op de hoogte gesteld van de digitale diefstal via een mail. Klanten kunnen pas weer wat bestellen als zij met de link uit de mail een nieuw wachtwoord hebben ingesteld. Mapp is de eigenaar van webwinkels als InternGeheugen.com, Geheugen.nl en Harddisk.nl. Het gaat om gegevens van 157.000 klanten. "Daar zitten ook veel klanten bij die jaren geleden voor het laatst iets hebben besteld, dus veel van de mails komen terug, omdat het mailadres al niet meer bestaat." Andere gegevens, zoals adres en betalingsgegevens, zijn niet gestolen.

De dief of dieven konden de database in via een sql-injection, zegt Tersmette. "We hebben de hele nacht doorgewerkt om het lek te dichten en de beveiliging te verbeteren." Naast het dichten van het lek beveiligt de webwinkelketen wachtwoorden nu beter. "Dat doen we nu met een 256bits sha2-codering met dubbele salt, dus dat zal een stuk veiliger moeten zijn."