Vulnerabilities caused by faulty settings of web application components

A V Gorban¹, R I Aliev² and K Yu Zhigalov³

¹ Irkutsk National Research Technical University, 83, Lermontova str., Irkutsk, 664074, Russia
² Chechen State University, 17a, Dudaev Boulevard, Groznyy, 364015, Russia
³ V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Science

E-mail: kafedra.bi@mail.ru

Abstract. Any web application intrinsically consists of many components (bases, frameworks, loaders). Errors occurring in these components can have multiple implications and even a minor component can reduce the overall security of the application. The paper will be useful for both information security experts and casual users.

1. Introduction
Any web application operates over several different layers of the OSI model with different components to operate at different layers, and of course, vulnerability is likely to exist within different layers [1, 2].

It is important that any web application consists of several structural parts.

![Web app configuration](Image)

**Figure 1.** Web app configuration

This includes the client side (can be a mobile or desktop computer), network devices that connect computers to the server side (firewall or server filter), web services, an application server that implements the business logic (application server is not an obligatory part since it controls workflows,
and is required only for large systems), databases, and various services. There is a likelihood to make a mistake in any of these components and, accordingly, cause a vulnerability [3-5].

The problems can be divided into:
- those caused by incorrectly written code;
- those that exist in plug-in components.

The diagram displays the structure of the application as seen by the administrator, i.e. as some nodes that perform their roles in these applications, and as seen by the app developer, the application looks as follows:

![App as seen by the programmer](image)

Figure 2. App as seen by the programmer

This is a fairly large number of components, with part of the application server code at the top, the frameworks responsible for running the application – coming below, database components and query conversion – at the very bottom, and applications that work separately with certain objects (e.g. only with databases or conversion queries).

2. **Invalid differentiation of rights**

Invalid differentiation of access rights is a fairly common problem that may occur. Any application requires some kind of administration and these interfaces should not be accessible by anyone other than the administrator. First, it should be clear which interfaces are present in a particular application server, since this is not always obvious [6, 7].

How do access-rights problems come up?
- Firstly, the application has to be managed somehow, and this should be done by a limited number of people.
- Secondly, the application must have access to system resources, yet limited.
- There are two major problems with invalid differentiation of rights:
  - Interface accessibility;
• Invalid user access rights that are used to run components. If possible, all rights granted to a component should be minimized. A vulnerability is often caused by a situation when the administrator downloading a software installer simply clicks “next” and accepts all possible options, without realizing which function the application should be allowed to perform and which should be disabled. Accordingly, this practice of accepting default settings often leads to vulnerabilities. How it all ends up:

Figure 3. Hacker’s signature

Therefore, when installing any component and connecting it, you need to be aware what it does, what potential security threats it can pose, and, if possible, minimize the part that is not required in the application.

3. An example of vulnerability in the form of interface accessibility
Accessibility of management interfaces, including virtually dead network ports, URLs.

Figure 4. Network ports

The example shows network scanning results, displaying the open ports for SSH remote management, as well as the port for the MySQL database management. With proper security, it is impossible to gain access from a desktop computer in a non-administrator network (access should be provided to a specific range of IP addresses).

4. Over-granting of privileges
An example of over-granted privileges is the following entry in the password file (on Linux systems):
www-data:x:33:33:www-data:/var/www/:bin/sh

Where www-data is the user the server is running with, a /bin/sh is the console permissions.

A casual user the server is managing should not be permitted to use the server console. However, in a situation with incorrectly granted privileges, care should be taken when restricting rights, since in some cases user rights disabled may result in a server failure at large.

One example of a misconfigured application is the display of errors that occur when working with the server. In particular, this is the stack trace output in JAVA applications. Disclosure of information in error messages (framework, PL, DBMS, server).

From the errors, you can determine which methods are used to transfer data, usernames, finding the web directory. It is therefore essential that the error output is switched off.

There are systems for collaborative code development (git, svn) that, once misused, often cause a vulnerability. For example, an administrator, without checking for a vulnerability, simply copies the code in its entirety and runs it on the server. Often, fragments of such code contain access to the root directories of the server on which they are run. This slavish copying makes the root directory accessible from the Internet. Having access to the root directory does not make it difficult to access the files that the server itself runs on.

![Index directory](image)

**Index of */jv*/.svn Index of */.git**

- **Parent Directory**
- HEAD
- config
- description
- hooks
- index
- info/
- log
- objects/
- packed-refs
- refs/
- shallow

Responsibility for such vulnerabilities lies with both the developer who left the vulnerability and the administrator who allowed the vulnerability to be exploited.

For git, you can use the git – clean command, which removes unneeded directories.

Vulnerabilities are also found on client hardware, concerning, in particular, the use of auto-completion (i.e. in login/password fields) and caching of the data used, the lack of protection against XSS, clickjacking (creating a copy of the site with a transparent page to superimpose attacker’s forms to intercept data).

Typical web browser problems:
- Auto completion and caching;
- Lack of security flags or headers (protection against client-side attacks: Clickjacking, XSS);
- Invalid setting of cookie scope.

Another example is the accessibility of cookies from the outside:
Set-Cookie: ...
Path=/
Domain:.bank.com

In the above example, cookies are available from a wide range of domains (the dot before bank specifies a whole range of subdomain names).

Other quite common problems are the presence of files generated during development in system directories (local copies of working scenarios, standard non-protected directories) [8]. An important aspect is proper configuration of the server, because quite often the developer or server administrator leaves the MultiView mode (Apache httpd) enabled or the indexing of directories that lets an attacker
find out the contents of the server (file names, directory names, etc.). To make matters worse, the silliest mistake is to have default users.

However, one of the most hazardous vulnerabilities is to integrate developer mode, which makes it easier for developers to create and immediately displays errors. It can lead to the disclosure of information of all kinds. Through this vulnerability, it is easy to get access to the system files, see the login/password pairs used in debugging (most probably they will be valid, since not every administrator thinks about deleting debug information, hoping for the developer’s common sense).

5. Faulty settings
Another common problem is faulty settings in the configuration file of the web server for virtual hosts (a mechanism that allows a server with one IP address to contain several different domains). The http v 1.1 protocol requires a mandatory “host” field in the client request header, which specifies the domain name of the virtual server.

When it comes to an application server, there is such a thing as “three-tier application architecture”. The two-tier architecture consists of a client while on the server side there is a database as the second tier. The three-tier architecture also has an application server. No one ever thinks about it, despite having great potential for an attack within the network (having a local connection), because it has broad functionality, usually has many management interfaces, open ports and a lot of default accounts). When attacking a company’s local network, the application server and the corresponding management consoles are the first to search for, wherefrom the default system management records are derived. An example application server is IBM WebSphere that has a standard 9060 port that hosts the console (Fig. 13).

IBM WebSphere:
http://host:9060/ibm/console (9043 for https), like wasadmin:wasadmin

![Figure 6. Account login page](image)

IBM WebSphere products do not have a single default account, but have fairly standard username/password pairs (i.e. wasadmin: wasadmin). Accordingly, the problem is to provide an opportunity to manage the application server due to its accessibility. Such an interface can be found using nmap and nse scripts:

Nmap: http-webspere-console-brute.nse
Nmap –p9060 –sV –sT –script=http-webspere-console-brute.nse –script-args ‘userdb=user.txt, passwd.txt’ 31.33.73.1

6. Conclusion
Based on the information presented, you can conclude that correct app settings is the key to successful protection of the system as such. The most crucial setting errors are those in the differentiation of access permissions and over-granted privileges.
References

[1] Karlina A I, Kondrat'ev V V, Kolosov A D, Balanovskiy A E and, Ivanov N A 2019 Production of new nanostructures for modification of steels and cast irons *IOP Conference Series: Materials Science and Engineering* 560(1) 012183

[2] Ershov V A, Kondratiev V V, Karlina A I, Kolosov A D and Sysoev I A 2018 Selection of control system parameters for production of nanostructures concentrates *Journal of Physics: Conference Series* 1118(1) 012014

[3] Kondratiev V V, Karlina A I, Guseva E A, Konstantinova M V and Kleshnin A A 2018 Processing and Application of Ultra disperse Wastes of Silicon Production in Construction *IOP Conference Series: Materials Science and Engineering* 463(3) 032068

[4] Gura D A, Gura A Y, Volkova T A and Lipilin D A 2020 Recreational space as a factor of sustainable development of coastal geosystems of the Krasnodar region *IOP Conference Series: Earth and Environmental Science* 579(1) 012007 doi:10.1088/1755-1315/579/1/012007

[5] Suslov K V, Solonina N N and Smirnov A S 2011 Smart meters for distributed filtering of high harmonics in Smart Grid *International Conference on Power Engineering, Energy and Electrical Drives* 6036552

[6] Buryanina N, Korolyuk Y, Koryakina M, Lesnykh E and Suslov K 2019 Algoritm of Current Protection Based on Three Instantaneous-Value Samples *Proceedings of IEEE PES Innovative Smart Grid Technologies Europe* 8905542

[7] Sivtsov A V, Yolkin K S, Kashlev I Më Karlina A I 2020 Processes in the Charge and Hearth Zones of Furnace Working Spaces and Problems in Controlling the Batch Dosing Mode during the Smelting of Industrial Silicon and High-Silicon Ferroalloys. *Metallurgist* 64(5-6) 396–403

[8] Ershov V A, Karlina A I, Karlina Y I 2021 Process Control System for Producing a Silica-Based Modifying Additive. *Metallurgist* 65(3-4) 446–453