Objective C static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your OBJECTIVE C code

All rules 315 | Vulnerability 10 | Bug 75 | Security Hotspot 18 | Code Smell 212 | Quick Fix 13

Tags

Search by name...

Using hardcoded IP addresses is security-sensitive

Analyze your code

Security Hotspot Minor cert owasp

Hardcoding IP addresses is security-sensitive. It has led in the past to the following vulnerabilities:

- CVE-2006-5901
- CVE-2005-3725

Today's services have an ever-changing architecture due to their scaling and redundancy needs. It is a mistake to think that a service will always have the same IP address. When it does change, the hardcoded IP will have to be modified too. This will have an impact on the product development, delivery, and deployment:

- The developers will have to do a rapid fix every time this happens, instead of having an operation team change a configuration file.
- It misleads to use the same address in every environment (dev, sys, qa, prod).

Last but not least it has an effect on application security. Attackers might be able to decompile the code and thereby discover a potentially sensitive address. They can perform a Denial of Service attack on the service, try to get access to the system, or try to spoof the IP address to bypass security checks. Such attacks can always be possible, but in the case of a hardcoded IP address solving the issue will take more time, which will increase an attack's impact.

Ask Yourself Whether

The disclosed IP address is sensitive, e.g.:

- Can give information to an attacker about the network topology.
- It's a personal (assigned to an identifiable person) IP address.

There is a risk if you answered yes to any of these questions.

Recommended Secure Coding Practices

Don't hard-code the IP address in the source code, instead make it configurable with environment variables, configuration files, or a similar approach. Alternatively, if confidentially is not required a domain name can be used since it allows to change the destination quickly without having to rebuild the software.

Sensitive Code Example

```
dbi_conn conn = dbi_conn_new("mysql");
string host = "10.10.0.1"; // Sensitive
dbi_conn_set_option(conn, "host", host.c_str());
dbi_conn_set_option(conn, "host", "10.10.0.1"); // Sensitive
```

Compliant Solution

```
dbi_conn conn = dbi_conn_new("mysql");
string host = getDatabaseHost(); // Compliant
dbi_conn_set_option(conn, "host", host.c_str()); // Compliant
```

Exceptions

No issue is reported for the following cases because they are not considered sensitive:

- Loopback addresses 127.0.0.0/8 in CIDR notation (from 127.0.0.0 to 127.255.255.255)
- Broadcast address 255.255.255.255

- Non routable address 0.0.0.0
- Strings of the form `2.5.<number>.<number>` as they often match Object Identifiers (OID).

See

- OWASP Top 10 2021 Category A1 - Broken Access Control
- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- CERT, MSC03-J. - Never hard code sensitive information

Available In:

sonarcloud sonarqube Developer Edition

Freed memory should not be used

Bug

Memory locations should not be released more than once

Bug

Memory access should be explicitly bounded to prevent buffer overflows

Bug

Printf-style format strings should not lead to unexpected behavior at runtime

Bug

Recursion should not be infinite

Bug

Resources should be closed

Bug

Hard-coded credentials are security-sensitive

Security Hotspot

"goto" should jump to labels declared later in the same function

Code Smell

Only standard forms of the "defined" directive should be used

Code Smell

Switch labels should not be nested inside non-switch blocks

Code Smell