BLOGS & STORIES

SpiderLabs Blog

Attracting more than a half-million annual readers, this is the security community's go-to destination for technical breakdowns of the latest threats, critical vulnerability disclosures and cutting-edge research.

All Your Databases Belong To Me! A Blind SQLi Case Study

April 22, 2021 Andreas Georgiou

*"**All your base are belong to us**", Zero game 1992*

Intro

The following blog post does not include any novel attack vectors. On the contrary, it serves as a humble reminder that the same software bugs discovered more than a decade ago are also found in commercial software products in 2021. It also highlights once more the necessity of conducting security assessments on a regular basis.

It follows the story of how a rather usual infrastructure penetration test was concluded with two database compromises and two 2 CVEs submissions.

The Attack

The scope of this engagement consisted of a large set of IP ranges of the client. The reconnaissance phase started by enumerating the targets and OS fingerprinting. AttackSurfaceMapper (https://github.com/superhedgy/AttackSurfaceMapper) was then used to scan for subdomains and discover additional hostnames. By utilizing the discovered hostnames it was possible to extend the attack surface and explore web applications which many times are not accessible directly using the IP address.

The first database was compromised due to a web server configuration error. The MSSQL server appeared to be performing regular backups and archiving the backups on an HTTP server which was exposed on the internet and required no authentication. SpiderLabs found the insecure '.bak' files and performed a full restore. There is no need to elaborate further apart from mentioning that the reconstructed database contained critical business data.

During the reconnaissance phase, a number of hosts serving web applications were spotted. Among the web applications that caught my attention was a web login of an application called 'Aurall (https://aurall.eu/)'. According to the vendor's manual "AURALL is a computer system based on open architecture PC for recording, storage, administration, and management of telephone conversations."

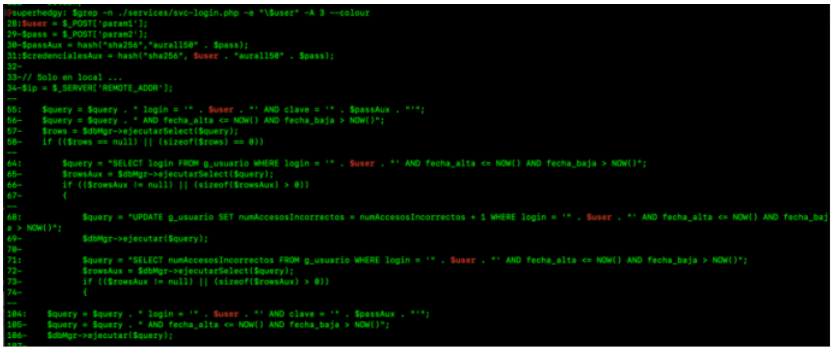
In this case, a login page for the **AurallRECMonitor** module was exposed to the Internet. Testing it against the usual set of tricks and techniques including password spraying didn't result in anything. However, a trivial web server configuration error allowed SpiderLabs to download the '. `git` ' directory hosted on the target host and extract the PHP source code files of AurallRECMonitor.

Application developers usually store database connection details including the database name/host/username and password in a file. In this case, it was found in the 'appconfig.php' file. However, since the MySQL service was only accepting connections from the localhost, it was not possible to leverage the login credentials remotely.

The Vulnerable Software

Having access to the source code allows the tester to review the code directly and find security vulnerabilities without the need to blindly test different application components. With the help of grep commands, possible SQL injection points can be discovered.

```
$ grep -iorE "^.*?query.*?SELECT.*?WHERE.*?" --colour ./git/* -B 3
```

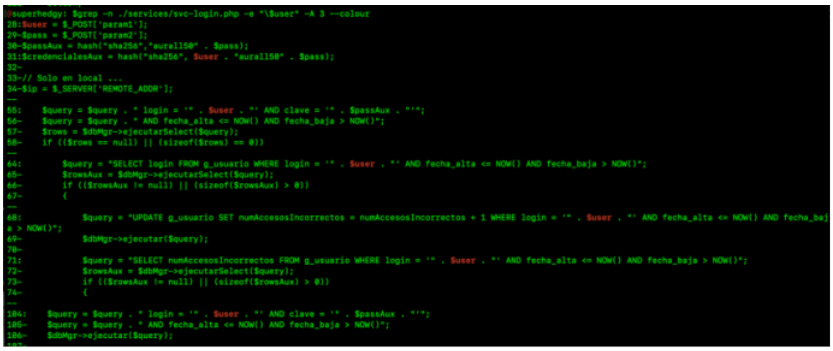


(https://npercoco.typepad.com/.a/6a0133f264aa62970b0278802460ee200d-popup)

Figure 2.0 - The grep command output shows possible malicious injection locations for further investigation.

We chose to focus on the 'svc-login.php' component responsible for handling the authentication requests. After reading through the source code it was clear that 'param1' and 'param2' parameters were accepting user input which was used to execute MySQL queries on the backend database. More specifically, variable 'param1' was storing the username value which could be manipulated to execute arbitrary SQL statements.

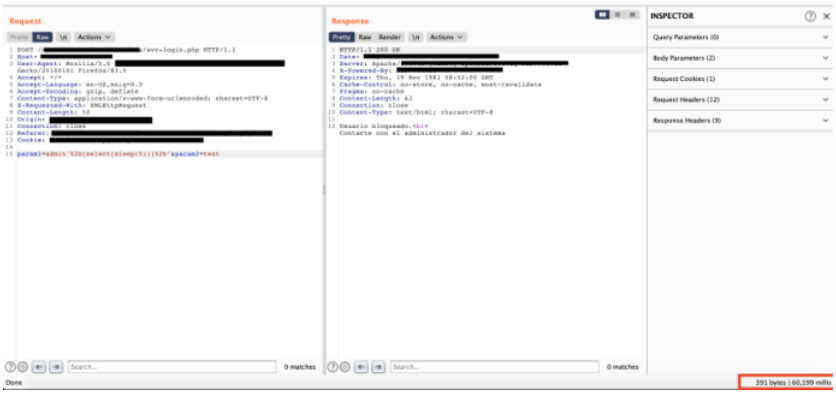
```
$ grep -rnw ./* -e "param1" -e "\$user" -A 3 --colour
```



(https://npercoco.typepad.com/.a/6a0133f264aa62970b0278802460ee200d-popup)

Figure 2.0 - The grep command output shows possible malicious injection locations for further investigation.

A Proof of Concept (PoC) SQL injection containing a `sleep()` statement was crafted and launched against the login page of the web application. This vulnerability was disclosed as CVE-2021–25899 (https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-25899).



(https://npercoco.typepad.com/.a/6a0133f264aa62970b027880246094200d-pi)

Figure 3.0 - A view of a proxy software showing the PoC payload executed on the AURALL server.

This was a "blind" SQL Injection vulnerability, meaning that the application did not return any error strings or any other significant data during the attack. Data can be inferred one bit at a time based on subtle differences in the server's response, or in this case the amount of time taken by the server to generate a response. Having access to the source code likely made the creation of PoC easier. It should be noted that the time it took the server to reply was well over 6,000 milliseconds and closer to 60,000 milliseconds, more than ten times the value used in the payload, "sleep(5)". This behavior didn't affect the exploitation and can be attributed to the fact that 'param1' value was used in multiple parts of the source code. As a result, it caused the SQL statements with the sleep function to be executed several times and cause additional delays.

Unfortunately for us, it was not possible to execute arbitrary commands on the remote server and escalate privilege due to the nature of the target environment. The combination of PHP and MySQL database technologies running in the background meant that 'Stacked Queries' were not supported and therefore only one SQL statement per query could be executed. In this case, the original query was a SELECT() statement. A series of consecutive time-based SELECT() queries inferred the content of the database and essentially exfiltrated all the tables and records.

	ASP	ASP.NET	PHP
MSSQL	✓	✓	✓
MySQL	✗	✓	✗
PostgreSQL	✓	✓	✓
Oracle	✗	✗	✓

(https://npercoco.typepad.com/.a/6a0133f264aa62970b027880246045200d-popup)

Table 1.0 - A Table showing which technologies support stacked queries.

The application also suffered from multiple instances of bad programming practice including login credentials hardcoded in the source code, CVE-2021–25898 (https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-25898).

```
$pass = $_POST['param2'];
$passAux = hash("sha256","aurall50" . $pass);
$credencialesAux = hash("sha256", $user . "aurall50" . $pass);

// Solo en local ...
$ip = $_SERVER['REMOTE_ADDR'];
if ($ip == "127.0.0.1") $ip = "localhost";
if ($ip == "::1") $ip = "localhost";
if (($credencialesAux == "9756a5dbf769873[redacted]') && ($ip == "localhost"))
{
    setDatosUsuario("0","0","adminvoid","[redacted]","[redacted]","SUPERADMINISTRADOR","600");
    echo "OK";
    return;
}
// El resto ...
else
{
    // Conectamos con la BD
    $dbMgr = new DBMgr();
    if (!$dbMgr->conectar()) {
        echo "No es posible conectar en este momento";
        return;
    }

    // Obtenemos los datos del usuario
    $query = "SELECT centro,cod_usuario,login,nombre,email,clase,tiempo_sesion,numAccesosIncorrectos FROM g_usuario WHERE";
```

(https://npercoco.typepad.com/.a/6a0133f264aa62970b027880246067200d-pi)

Figure 4.0 - A screenshot shows an example of hardcoded credentials stored in the '$credecialesAux' parameter.

The screenshot in Figure 4 depicts the vulnerable PHP source code of the application in the 'svc-login.php' file. The hash of the User/Password pair is compared against a SHA256 hash value already hardcoded in the application.

The Interesting Part

As more companies transition to the cloud so do vendors. The current trend is picked up by many software vendors who sell their products using the Software as a service (SaaS) delivery model.

It appeared that the same software vendor was hosting multiple instances of the same vulnerable application under a domain name structure that can easily be guessed or found. In short, it was possible to guess the right FQDN and find the login panels for more vulnerable instances used by different companies.

```
company_name1.asoftwarevendor.com
company_name2.asoftwarevendor.com
```

I won't provide any further details on how SpiderLabs discovered this issue in order to maintain some privacy here. In any case, it was trivially discovered using the standard enumeration techniques and a bit of guessing. It should be noted that this analysis was mainly done using passive information already available on the Internet.

EOF

Even many years after SQLi was first demonstrated in a 1998 issue of Phrack (http://phrack.org/issues/54/8.html#article) magazine, injections are still one of the leading OWASP Top 10 (https://owasp.org/www-project-top-ten/) web application vulnerabilities. For those who might not be familiar, OWASP is an organization promoting Web Application security by providing guidelines and best practice secure development. The first version of OWASP Top 10 in 2013 listed "Injection" vulnerabilities right on the top of the list and you can see from the current list that not much has changed.

Rank	OWASP Top 10 (2013)	OWASP Top 10 (Current)
A1	Injection	Injection
A2	Broken Authentication and Session Management	Broken Authentication
A3	Cross-Site Scripting (XSS)	Sensitive Data Exposure
A4	Insecure Direct Object References	XML External Entities (XXE)

A5	Security Misconfiguration	Broken Access Control
A6	Sensitive Data Exposure	Security Misconfiguration
A7	Missing Function Level Access Control	Cross-Site Scripting (XSS)
A8	Cross-Site Request Forgery (CSRF)	Insecure Deserialization
A9	Using Components with Known Vulnerabilities	Using Components with Known Vulnerabilities
A10	Unvalidated Redirects and Forwards	Insufficient Logging & Monitoring

Table 2.0 - A comparison of OWASP Top 10 (2013) against the latest official update.

Over the years great progress has been made and application security has been incorporated in all stages of the development process. However, classic security vulnerabilities such as injections will probably change but remain a risk for a long time.

SpiderLabs followed our Responsible Disclosure Policy (https://www.trustwave.com/en-us/legal-documents/trustwave-spiderlabs-vulnerability-disclosure-policy/) for the CVE-2021–25899 (https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-25899) and CVE-2021–25898 (https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-25898) vulnerabilities. Trustwave attempted to contact the software vendor multiple times prior to releasing this blog post. However, Trustwave received no response from the vendor.

References:

- Blind time-based SQL Injection (CVE-2021–25899 (https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-25899))
- Hardcoded Credentials in the source code (CVE-2021–25898 (https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-25898))
- Phrack Magazine, Volume 8, Issue 54 Dec 25th, 1998, article 08 of 12 – (http://phrack.org/issues/54/8.html#article (http://phrack.org/issues/54/8.html#article))
- OWASP Top Ten - (https://owasp.org/www-project-top-ten/ (https://owasp.org/www-project-top-ten/))
- OWASP Top 10 - 2013 (https://owasp.org/www-pdf-archive/OWASP_Top_10_-_2013.pdf (https://owasp.org/www-pdf-archive/OWASP_Top_10_-_2013.pdf))
- AttackSurfaceMapper (https://github.com/superhedgy/AttackSurfaceMapper (https://github.com/superhedgy/AttackSurfaceMapper))
- TWSL2021-007 - Multiple Vulnerabilities in AURALL REC MONITOR (/en-us/resources/security-resources/security-advisories/?fid=28765)

Related SpiderLabs Blogs

(/en-us/resources/blogs/spiderlabs-blog/microsoft-teams-and-skype-logging-privacy-issue/)

Microsoft Teams and Skype Logging Privacy Issue (/en-us/resources/blogs/spiderlabs-blog/microsoft-teams-and-skype-logging-privacy-issue/)

SPIDERLABS BLOG

(/en-us/resources/blogs/spiderlabs-blog/servicenow-username-enumeration-vulnerability-cve-2021-45901/)

ServiceNow - Username Enumeration Vulnerability (CVE-2021-45901) (/en-us/resources/blogs/spiderlabs-blog/servicenow-username-enumeration-vulnerability-

SPIDERLABS BLOG

(/en-us/resources/blogs/spiderlabs-blog/sql-injection-in-wordpress-plugins-order-and-order-by-as-overlooked-injection-points/)

SQL Injection in WordPress Plugins: ORDER and ORDER BY as Overlooked Injection Points (/en-us/resources/blogs/spiderlabs-blog/sql-injection-in-

SPIDERLABS BLOG

(/en-us/resources/blogs/spiderlabs-blog/microsoft-teams-and-skype-logging-privacy-issue/)

Microsoft Teams and Skype Logging Privacy Issue (/en-us/resources/blogs/spiderlabs-blog/microsoft-teams-and-skype-logging-privacy-issue/)

SPIDERLABS BLOG

(/en-us/r
blog/serv
vulnerabi

Servi
Enum
(CVE
us/re
blog/
enum

SPIDERL

Stay Informed

Sign up to receive the latest security news and trends from Trustwave.

Business Email

SUBSCRIBE

English

(https://www.linkedin.com/company/trustwave)

(https://twitter.com/Trustwave)

(https://www.facebook.com/Trustwave/)

(https://www.youtube.com/channel/UC2CCqdrAxFv83NOdjhqA)

Leadership Team (/en-us/company/about-us/leadership/)

Our History (/en-us/company/about-us/our-history/)

News Releases (/en-us/company/newsroom/news/)

Media Coverage (/en-us/company/newsroom/media/)

Careers (https://jobs.jobvite.com/trustwave)

Global Locations (/en-us/company/global-locations/)

Awards & Accolades (/en-us/company/about-us/accolades/)

Trials & Evaluations (/en-us/resources/security-resources/special-offers/)

Contact (/en-us/company/contact/)

Support (/en-us/company/support/)

Security Advisories (/en-us/resources/security-resources/security-advisories/)

Software Updates (/en-us/resources/security-resources/software-updates/)