
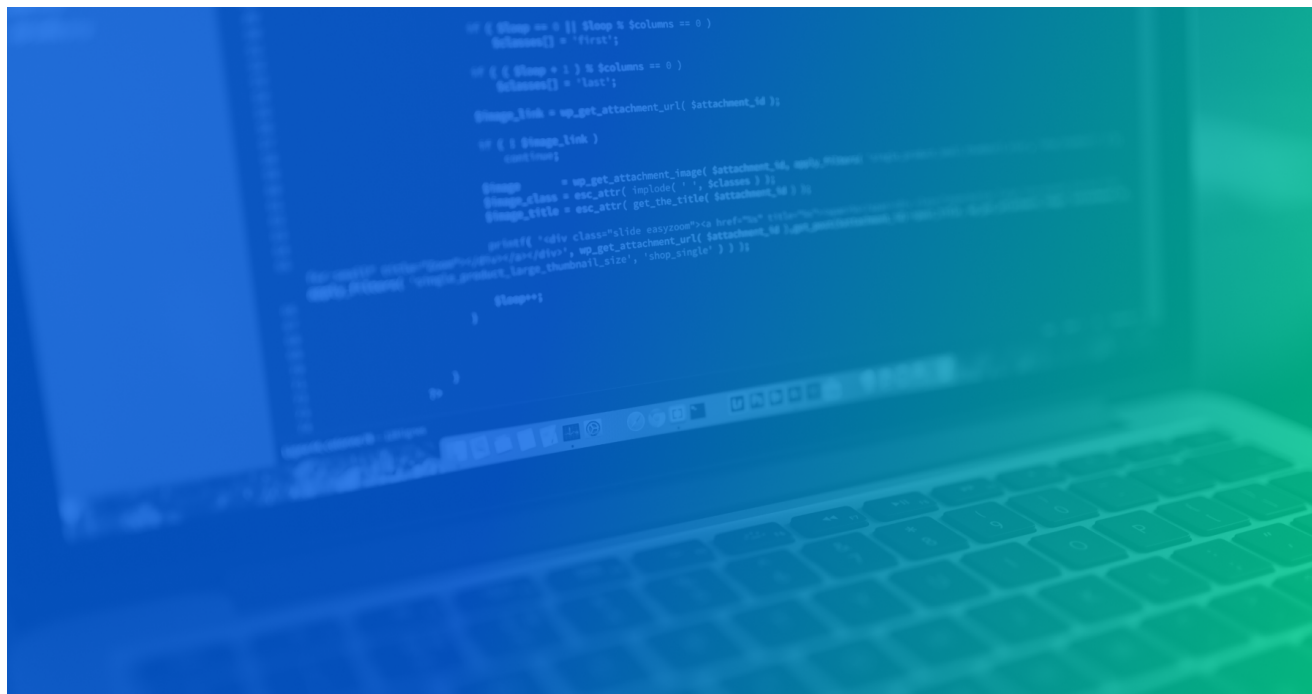


Threat Advisory: Hackers Are Exploiting a Vulnerability in Popular Billing Software to Deploy Ransomware

 huntress.com/blog/threat-advisory-hackers-are-exploiting-a-vulnerability-in-popular-billing-software-to-deploy-ransomware

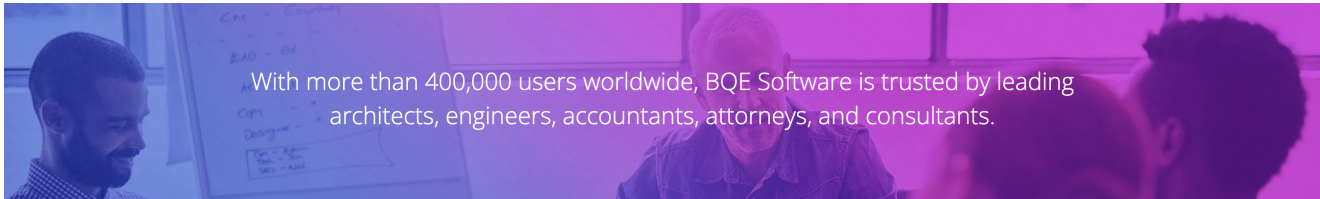


Hackers are constantly looking for low-hanging fruit and vulnerabilities that can be exploited—and they’re not always poking around in “big” mainstream applications like Office.

Sometimes, a productivity tool or even an add-on can be the door that hackers step through to gain access to an environment and carry out their next move. Huntress recently discovered one such vulnerability in a time and billing system called BillQuick.

What Did We Find?

The Huntress ThreatOps team discovered a critical vulnerability in multiple versions of BillQuick Web Suite, a time and billing system from BQE Software. Hackers were able to successfully exploit [CVE-2021-42258](#)—using it to gain initial access to a US engineering company—and deploy ransomware across the victim’s network. Considering BQE’s self-proclaimed user base of 400,000 users worldwide, a malicious campaign targeting their customer base is concerning.



Our team was able to successfully recreate this SQL injection-based attack and can confirm that hackers can use this to access customers' BillQuick data and run malicious commands on their on-premises Windows servers. We have been in close contact with the BQE team to notify them of this vulnerability, assess the code changes [implemented in WebSuite 2021 version 22.0.9.1](#) and work to address multiple security concerns we raised over their BillQuick and Core offerings (more to come on these when patches are available).

- [CVE-2021-42344](#)
- [CVE-2021-42345](#)
- [CVE-2021-42346](#)
- [CVE-2021-42571](#)
- [CVE-2021-42572](#)
- [CVE-2021-42573](#)
- [CVE-2021-42741](#)
- [CVE-2021-42742](#)

The Red Flag

Our spidey senses were first set off after a number of our Ransomware Canary files were tripped within an engineering company's environment that was managed by one of our partners. While investigating the incident, we discovered Microsoft Defender antivirus alerts indicating malicious activity as the `MSSQLSERVER$` service account. This indicated the possibility of a web application being exploited in order to gain initial access. The server in question hosted BillQuick Web Suite 2020 (WS2020), and the connection logs indicated a foreign IP repeatedly sending POST requests to the web server logon endpoint leading up to the initial compromise.

From this context, we suspected that a bad actor was attempting to exploit BillQuick—so naturally, we began reverse-engineering the web application to trace the attacker's steps.

Vulnerability Analysis

After downloading a free copy of WS2020 from the BQE website, we installed it locally and began to investigate. During static analysis of the server-side code, the Huntress team identified concatenated SQL queries. Essentially, this function allows a user to control the query that's sent to the MSSQL database—which in this case, enables blind SQL injection via the application's main login form.

With help from our partner, we were able to recreate the victim's environment and validate simple security tools like sqlmap easily obtained sensitive data from the BillQuick server without authentication. Because these versions of BillQuick used the `sa` (System Administrator) MSSQL user for database authentication, this SQL injection also allowed the use of the `xp_cmdshell` procedure to remotely execute code on the underlying Windows operating system.

Showcasing the SQL injection in the login page

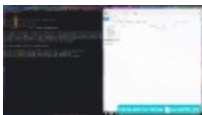
Let's walk through how we were able to recreate the SQL injection vulnerability in BillQuick. The below video showcases how easy it is to trigger this vulnerability by submitting a login request with invalid characters in the username field.



Simply navigating to the login page and entering a single quote (') can trigger this bug. Further, the error handlers for this page display a full traceback, which could contain sensitive information about the server-side code.

Scanning the application endpoint

Sqlmap, an open source cybersecurity testing tool, can be used to test for and exploit these types of vulnerabilities. The tool is able to automatically detect SQL injection vulnerabilities, generate queries to leak sensitive data from the backend database, and in certain cases, gain remote code execution.

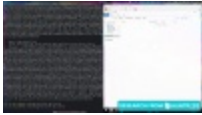


Here, we showcase an initial scan of the login endpoint. The file `login-request.txt` contains a raw HTTP request which performs an attempted login. There is nothing inherently malicious about this request, but `sqlmap` is capable of mutating this request to identify potential injection vulnerabilities.

In this case, the argument `-p txtID` tells `sqlmap` that we would like to test the `txtID` argument (this corresponds to the username input). `--time-sec` increases timeouts while sending requests (our BillQuick server was particularly slow to respond). The `--risk` and `--level` arguments adjust how aggressive `sqlmap` scanning will be. Because this is local testing, we specified higher risk and level values to better tap the full potential of `sqlmap`.

Multiple SQL injection points identified

At this point, `sqlmap` knows how to exploit the vulnerability. It is common to ask `sqlmap` to test multiple parameters, so it politely asks if we would like to continue testing others—in this case, we only asked it to test one parameter.

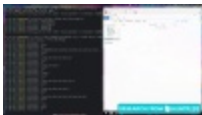


After scanning is complete, `sqlmap` will use the injection to fetch the details of the backend SQL database server. In this example, `sqlmap` correctly identified Microsoft IIS, Microsoft SQL Server 2019 and Windows 2019.

Next, we tell `sqlmap` to enumerate all databases on the SQL server; `sqlmap` remembers the previously identified vulnerabilities and will automatically use the most appropriate one. It quickly identifies the standard databases as well as `BQ2020` and `BQ2020ARK`.

Dumping SQL database tables

Now that we know the database name, we can begin dumping database data. In this case, we already know the name of a sensitive database table (`SecurityTable`) used by BillQuick Web Suite. This is easily identified by installing a local copy and inspecting the database. This table stores permission data and encoded passwords for all employees in the BillQuick database.



Just to reiterate: We have not authenticated, but we are still able to remotely leak highly sensitive employee information. As a billing management server, there is likely a lot more sensitive information living in the production database.

Gaining remote code execution

Leaking sensitive information is bad enough, but malicious actors are also gaining remote code execution with this vulnerability.



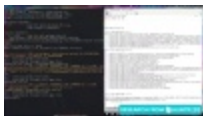
It is worth noting that if your database server is configured to block use of the `xp_cmdshell` extended stored procedure **and** BillQuick Web Suite was configured to use a least-privilege SQL user, remote code execution in this way would not be possible. However, BillQuick Web Suite setup information references the built-in `sa` account when discussing setup and installation multiple times. Any user with read/write access to the BillQuick database could be used, but it is common for system administrators to follow documentation verbatim.

In practice, Huntress has observed partners using BillQuick Web Suite with the built-in `sa` account, which allows full access to the back-end database server, including `xp_cmdshell`, regardless of configuration restrictions.

In this case, we use specially-crafted stacked SQL queries to execute the necessary commands for re-enabling the `xp_cmdshell` extended stored procedure and then execute code through `powershell.exe`. In the above video, we showcase writing to a file on the server host and spawning `calc.exe` as the `MSSQLSERVER$` service account.

Observing the sqlmap scanning in the logs

BillQuick Web Suite will typically write logs for exceptional conditions to `C:\BillQuickData\AppLog`. These logs are full tracebacks from the application when something goes wrong. While testing with `sqlmap`, many of these errors occurred and were logged here. This is a very useful place to check for past exploitation attempts or to debug your own testing.



It's worth noting, though, that this is not a sure-fire detection tool. Successful SQL injection queries will cause no errors to be logged. Further, some code paths within BillQuick Web Suite do not log exceptions to this file. However, the presence of shady or unusual failed SQL statements in your log file strongly suggests that someone has been poking where they shouldn't be.

...

Parting Thoughts

We really appreciate the BQE team's timely responses to these vulnerability notifications. In 2021, it's still extremely common for vendors to sweep cybersecurity issues under the rug; we have the impression that BQE is taking our feedback seriously.

With that said, this incident highlights a repeating pattern plaguing SMB software: well-established vendors are doing very little to proactively secure their applications and subject their unwitting customers to significant liability when sensitive data is inevitably leaked and/or ransomed.

Rather than stand idly by, Huntress is spearheading multiple SMB efforts to:

- Drive awareness of the code quality epidemic before hackers deliver a "great reckoning"
- Celebrate and destigmatize vendors who transparently disclose their corrected issues
- Incentivize security researchers to find and responsibly report vulnerabilities
- Hold vendors accountable for lagging security practices and unwelcoming behavior

As a community, we're going to be the security tide that raises all boats. It's time to rise up.



Caleb Stewart

Security Researcher at Huntress.