elmec

CVE-2026-51366: SQL Injection Leading to Remote Code Execution



Scritto da **Andrea Ferrario**

Summary

Field	Value
CVE ID	CVE-2026-51366
Product	Vedo Suite (Bottinelli Informatica)
Affected Version(s)	v1.2.5 (and possibly earlier/other versions using the same endpoint logic — unconfirmed)
Vulnerability Class	CWE-89: SQL Injection
Secondary Impact	CWE-78: OS Command Injection (via xp_cmdshell)
Attack Vector	Network (HTTP GET)
Privileges Required	Low (authenticated application user)
User Interaction	None
Backend	Microsoft SQL Server
Estimated CVSS 3.1	9.9 (Critical) — AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H

Affected Product Description

Vedo Suite is a management/administration application developed by Bottinelli Informatica. As part of its web-facing functionality, it exposes an internal chat feature through the /api_vedo/chat REST-style endpoint. This endpoint accepts a utente_chat parameter via HTTP GET, presumably used to identify or filter chat messages by user.

Tools Used

The findings described in this advisory were identified and validated using the following tools:

- **Burp Suite** — used for intercepting and manipulating HTTP requests to the /api_vedo/chat endpoint, manually crafting and replaying test payloads against the utente_chat parameter, and observing response timing/behavior differences.
- **sqlmap** — used to confirm and automate exploitation of the identified injection point, including detection of the injectable parameter, identification of the stacked-queries and time-based blind techniques, database fingerprinting, and enumeration of schema/metadata.

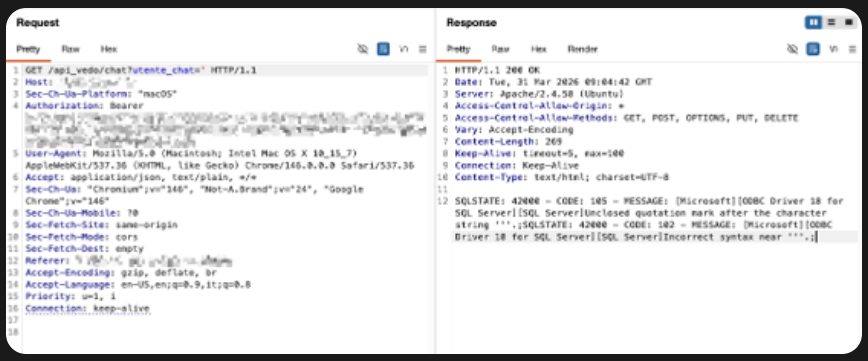


Figure 1: Burp Suite intercepting the vulnerable GET request to /api_vedo/chat, highlighting the utente_chat parameter.



Figure 2: sqlmap confirming the injectable parameter and identifying available techniques (stacked queries, time-based blind).

Technical Analysis

Root Cause

Cerca nel blog

using the value of the utente_chat GET parameter without adequate input validation, output encoding, or parameterization. This is a classic case of **dynamic SQL query construction via string concatenation**, where user-controlled input is inserted directly into the query text rather than being passed as a bound parameter.

Ultimi post pubblicati

Because the application does not use parameterized queries (e.g., via prepared statements, stored procedures with typed parameters, or an ORM enforcing parameter binding), an attacker who controls the utente_chat value can alter the logical structure of the SQL statement executed by the backend.

The advisory indicates the vulnerability is **not limited to a single injection point**: other endpoints and parameters that follow the same insecure query-construction pattern are also likely affected, suggesting a systemic coding pattern issue (e.g., a shared data-access layer or helper function that is inherently unsafe) rather than an isolated bug in one code path.

Exploitation Characteristics

Two exploitation techniques are specifically noted as effective against this target:

1. **Stacked queries** — Microsoft SQL Server, via most common drivers (e.g., ODBC, ADO.NET with SqlCommand when not parameterized), permits the

50annidiElmec About AI as400
ASSET MANAGEMENT automation award
backup business continuity career cloud
CMDB competenze comunicato connect
connettività container CSR cybersecurity
data center data management
data protection data security
device as a service devops dex
digital employee esperience
digital infrastructure digital workplace
disaster recovery elmec Elmec Vision
events formazione governance help desk
HR hybrid cloud ibm Infrastructure IT
IoT IT operation macchine virtuali
managed Licensing Service Managed Workplace
myelmec network news noleggio
noleggio operativo patch patch management
Platform Services postazioni di lavoro
press release SAP sase sdwan
security servizi gestiti sistemi informatici
Sostenibilità Sponsored Blog Post Storage
ticketing virtual machine vpn workplace
ztna

Escalation to Remote Code Execution

instance. xp_cmdshell is an extended stored procedure that allows execution of arbitrary operating system shell

that allows execution of arbitrary operating system shell commands with the privileges of the SQL Server service account.

Because stacked queries are possible, an authenticated attacker who can inject SQL can invoke xp_cmdshell directly within the same request used for the original injection, without needing a separate exfiltration channel. This effectively converts a SQL injection finding into full remote code execution (RCE) on the host running the SQL Server instance, subject only to the OS-level privileges of the service account.

xp_cmdshell is disabled by default in modern SQL Server installations; its presence here indicates either a legacy configuration, a deliberate administrative decision (e.g., for scheduled maintenance scripts), or an oversight during deployment — any of which significantly increases the real-world exploitability and blast radius of this vulnerability.

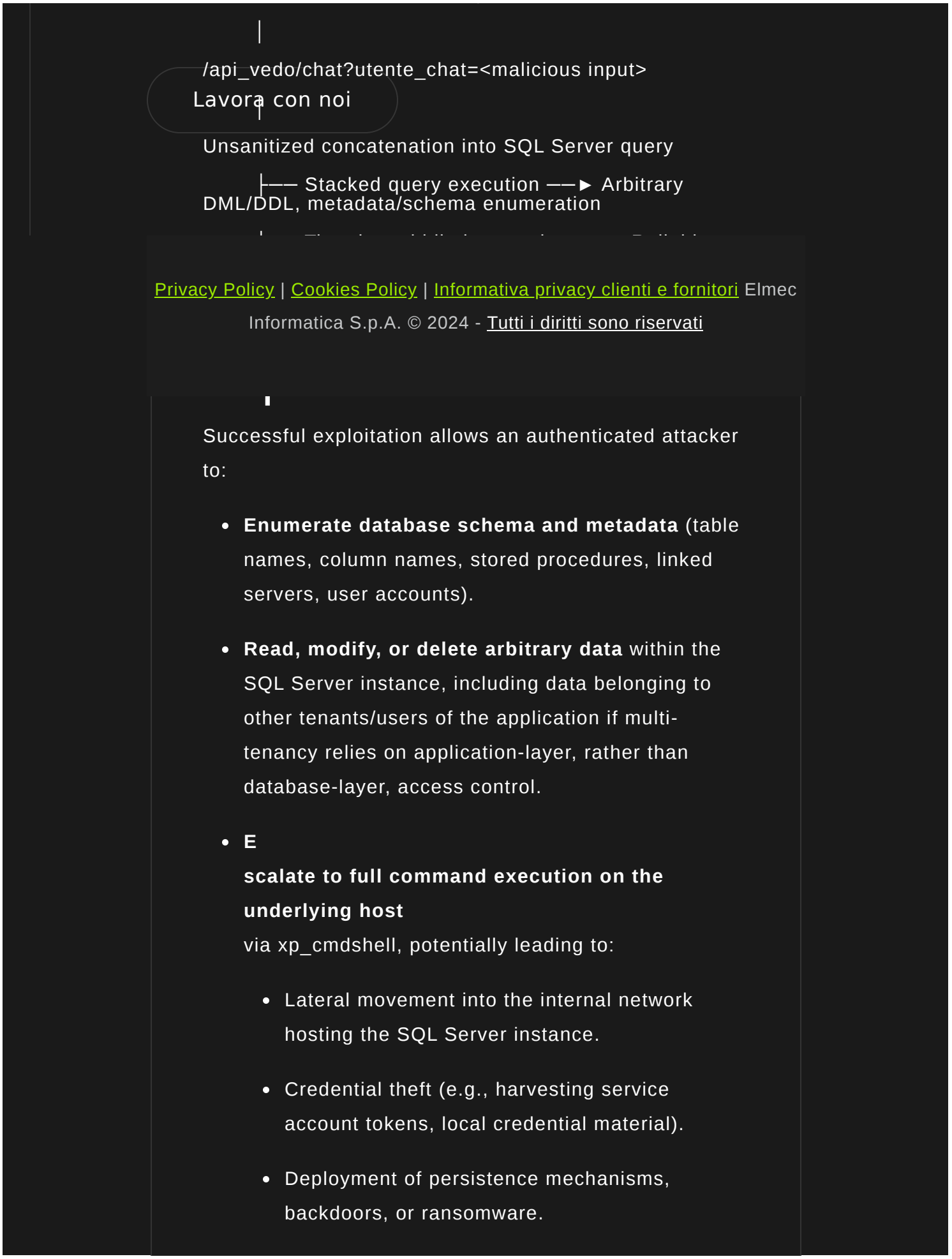


Figure 4: sqlmap leveraging xp_cmdshell through the identified injection point to obtain an OS-level command execution (os-shell)

Attack Chain Summary

Authenticated HTTP GET request

elmec
LinkedIn Instagram Facebook
+39 0332.802 111
Note legali
Assistenza tecnica
CHI SIAMO
Non possiamo parlare di tutto ma possiamo dirvi che dal 1971 quando ancora l'informatica era agli albori abbiamo iniziato a lavorare nel mondo dell'informatica: c'era la perforazione e c'eravamo noi.
Guarda come siamo diventati grandi.



Successful exploitation allows an authenticated attacker to:

- **Enumerate database schema and metadata** (table names, column names, stored procedures, linked servers, user accounts).
- **Read, modify, or delete arbitrary data** within the SQL Server instance, including data belonging to other tenants/users of the application if multi-tenancy relies on application-layer, rather than database-layer, access control.
- **E**
  scalate to full command execution on the underlying host
  via xp_cmdshell, potentially leading to:
  - Lateral movement into the internal network hosting the SQL Server instance.
  - Credential theft (e.g., harvesting service account tokens, local credential material).
  - Deployment of persistence mechanisms, backdoors, or ransomware.

- Full compromise of confidentiality, integrity, and availability of both the application and the host system.

Given that the vulnerability requires only a low-privileged authenticated session (not administrative access) to the Vedo Suite application itself, the effective barrier to exploitation is low in any environment where legitimate low-privilege user accounts exist — including potentially compromised or shared credentials, or malicious insiders.

CWE Mappings

- **CWE-89** — Improper Neutralization of Special Elements used in an SQL Command (‘SQL Injection’)
- **CWE-78** — Improper Neutralization of Special Elements used in an OS Command (‘OS Command Injection’) — via xp_cmdshell
- **CWE-284** — Improper Access Control (contributing factor, if low-privilege users can reach sensitive backend functionality)
- **CWE-1352** (as a broader pattern) — dynamic SQL construction without parameterization, likely repeated across multiple endpoints per the advisory’s own note

Mitigation and Remediation

1. **Deploy a Web Application Firewall (WAF)** with SQL injection detection rules as an interim compensating control until a code-level fix is available and applied.

2. **Restrict network-level access** to the affected application to trusted networks/VPN where feasible.

3. **Enable SQL Server Audit / Extended Events** to log execution of xp_cmdshell and stacked-query patterns, and monitor logs and network traffic for anomalous query behavior or unexpected xp_cmdshell invocations.

4. **Review credentials and audit logs** for the affected SQL Server instance, and rotate credentials where there is any possibility the flaw was exploited prior to patching.

5. **Apply vendor patches as soon as they are released**, and track the vendor's advisory for confirmation of the fixed version.

Generally speaking, in order to avoid this type of vulnerabilities developers should:

- **Use parameterized queries / prepared statements** for all database access, without exception. String concatenation of user input into SQL text should be replaced with parameter binding (e.g., SqlParameter in ADO.NET, or the equivalent mechanism in the ORM/data-access layer in use).
- **Audit all endpoints and parameters**, given that the underlying pattern may be systemic, a full review of the data-access layer — including any shared query-building utility functions — is warranted.
- **Avoid using xp_cmdshell** unless strictly required for a documented operational purpose. Where required, its use should be restricted through least-privilege SQL Server service accounts (never

LocalSystem/Domain Admin) and explicit GRANT/DENY controls limiting execution to only the accounts/roles that need it — never the application's own database login.

- **Apply the principle of least privilege** to the application's database account. The account used by the web application should not have permissions to execute extended stored procedures, alter schema, or access objects outside its functional need.
- **Implement strict input validation** as defense-in-depth (allow-listing expected formats for parameters), treating it as a secondary control rather than a substitute for parameterization.

Detection Guidance

Indicators that may suggest attempted or successful exploitation include:

- HTTP GET requests to /api_vedo/chat where utente_chat contains SQL metacharacters (', ;, --, /* */), SQL keywords (WAITFOR, EXEC, xp_cmdshell, UNION SELECT), or unusually long/encoded payloads.
- SQL Server error or audit logs showing execution of xp_cmdshell originating from the application's service account outside of expected maintenance windows.
- Anomalous response-time patterns on the affected endpoint consistent with time-based blind injection probing (many sequential requests with variable, deliberately induced delays).
- Unexpected child processes spawned by

sqlservr.exe (a strong indicator of xp_cmdshell abuse) on the database host.

References

- CWE-89: Improper Neutralization of Special Elements used in an SQL Command — **HTTPS://CWE.MITRE.ORG**
- CWE-78: Improper Neutralization of Special Elements used in an OS Command — **HTTPS://CWE.MITRE.ORG**
- Microsoft Docs — xp_cmdshell Server Configuration Option — **HTTPS://LEARN.MICROSOFT.COM**
- OWASP SQL Injection Prevention Cheat Sheet — **HTTPS://CHEATSHEETSERIES.OWASP.ORG**

Credits

Andrea Ferrario, Stefano Andreatta, CybergON