Vulnerability Analysis: CVE-2026-63077 Unauthenticated Remote Code Execution in JetBrains TeamCity

Author: Synthex **Site:** denizhalil.com CVSS 9.8 HIGH **Date:** August 2026

INTRODUCTION

On August 5, 2026, cybersecurity researchers disclosed a critical security vulnerability designated as **CVE-2026-63077** (CVSS v3.1 Score: 9.8 - Critical) affecting JetBrains TeamCity On-Premises installations. TeamCity is one of the world's most widely adopted Continuous Integration and Continuous Deployment (CI/CD) server solutions, serving as the core infrastructure for source code compilation, secret storage, and automated deployment pipelines across enterprise environments. CVE-2026-63077 represents an unauthenticated Remote Code Execution (RCE) vulnerability that allows threat actors to bypass authentication controls and execute arbitrary system commands on the host operating system with the privileges of the TeamCity process. Because CI/CD infrastructure holds extensive secrets—including cloud provider credentials, private SSH keys, and source code access—a breach of this nature poses catastrophic supply chain risks. Following evidence of active exploitation in wild campaigns, the US Cybersecurity and Infrastructure Security Agency (CISA) added CVE-2026-63077 to its Known Exploited Vulnerabilities (KEV) catalog.

LEARNING OBJECTIVES

After reviewing this technical article, security professionals, DevSecOps engineers, and system administrators will be able to:

- Understand the root cause and architectural mechanisms enabling CVE-2026-63077 exploitation.
- Identify the specific HTTP communication protocols and Java deserialization vectors involved.
- Evaluate the impact of CI/CD server compromise on organizational supply chain security.
- Implement effective patching, mitigation, and network hardening strategies to neutralize the threat.

WHAT IS REMOTE CODE EXECUTION IN JETBRAINS TEAMCITY (CVE-2026-63077)

JetBrains TeamCity relies on a distributed architecture where a centralized server orchestrates build tasks across multiple remote build agents. To coordinate these tasks, build agents continuously send HTTP(S) requests to server-side endpoints, polling for new jobs, reporting status updates, and uploading build artifacts. To allow seamless agent onboarding without manual user authorization, specific communication endpoints—particularly those under the `/app/agents/v1/` route—were intentionally designed to process incoming requests without requiring traditional administrative authentication or user session tokens. The vulnerability designated as CVE-2026-63077 is a critical, unauthenticated Remote Code Execution (RCE) flaw arising from how the TeamCity server processes data incoming through these open agent polling endpoints. When receiving HTTP requests from build agents, the server uses XML serialization mechanisms to reconstruct client-side Java objects. However, because these endpoints lack robust access controls and rely

on insecure unmarshaling practices, an unauthenticated attacker on the network can masquerade as a build agent and send malicious XML data directly to the server.

When the TeamCity server processes this untrusted input, the application automatically deserializes the payload without first validating the safety of the object types being instantiated. This allows a remote attacker to trigger arbitrary Java class execution, leading directly to operating system command execution under the privileges of the TeamCity server process. Because CI/CD platforms typically possess elevated network privileges, stored cloud provider keys, and source code access, successfully exploiting this vulnerability allows adversaries to establish a persistent foothold, compromise intellectual property, and initiate widespread software supply chain attacks.

Key characteristics and impact factors of this vulnerability include:

- **Unauthenticated Access:** Attackers do not require valid user credentials, session cookies, or prior agent registration tokens to reach the vulnerable endpoint handlers.
- **Network Attack Vector:** Exploitation occurs entirely over network protocols (HTTP/HTTPS), making any internet-facing TeamCity instance directly vulnerable to remote automated exploitation.
- **Elevated Operational Risk:** Code executed via this vulnerability inherits the permissions of the host account running the TeamCity service, often enabling total host takeover or container escape.
- **Supply Chain Exposure:** Gaining unauthorized control over the primary CI/CD server enables threat actors to modify source code repositories, tamper with build artifacts, and inject malicious code into downstream software releases.

```
// Vulnerable Server-Side Request Handling (Conceptual Example)
@POST
@Path("/app/agents/v1/poll")
@Consumes(MediaType.APPLICATION_XML)
public Response handleAgentPoll(InputStream rawPayload) {
    // VULNERABILITY: Unauthenticated endpoint accepting raw XML stream
    XStream xstream = new XStream();

    // An explicit allowlist was specified...
    xstream.allowTypes(new Class[]{ AgentPollResponse.class, BuildStatus.class });

    // BUG: Missing 'xstream.addPermission(NoTypePermission.NONE);'
    // Without resetting permissions, XStream defaults allow arbitrary gadget chains!

    // Deserializing untrusted input executes attacker-controlled gadget chains
    Object deserializedObj = xstream.fromXML(rawPayload);

    return Response.ok().build();
}
```

TECHNICAL DETAIL: HOW THE VULNERABILITY WORKS

Under the hood, CVE-2026-63077 is rooted in insecure Java object deserialization within TeamCity's agent communications module, which uses the **XStream** library to serialize and unmarshal XML data payload

traffic. Distributed build agents routinely exchange status updates, job polling signals, and environment metadata with the central server via REST endpoints under the `/app/agents/v1/` route. Because new build agents must be able to register dynamically without pre-existing administrator session cookies, these specific routes were deliberately designed to accept incoming HTTP requests without enforcing strict authentication checks. The critical flaw lies in how the TeamCity server configured its XStream parser to process these incoming unauthenticated XML streams. Although JetBrains developers attempted to restrict accepted classes by defining explicit allowlists (such as specifying expected agent status objects), they omitted the essential security initialization step: clearing XStream's default permissive fallback rules via `xstream.addPermission(NoTypePermission.NONE)`. Consequently, the security allowlist acted only as an additive filter rather than a restrictive boundary, leaving XStream's underlying object creation engine exposed to arbitrary Java class instantiation.

When an attacker transmits a maliciously crafted HTTP POST request containing XML-formatted gadget chains (such as dynamic proxy handlers, `java.lang.reflect.Proxy`, or standard JDK reflection utilities), XStream processes the payload without validation. During the unmarshaling phase, XStream automatically reconstructs the nested object structure, invoking object constructors, getter/setter methods, or `hashCode()` call chains. This chain reaction causes the server JVM to invoke sensitive runtime methods—such as `java.lang.ProcessBuilder.start()`—and immediately executes arbitrary system commands with the host permissions of the TeamCity process.

Key technical breakdown of the vulnerability flow:

- **Unauthenticated Endpoint Reachability:** The agent communication handlers beneath `/app/agents/v1/` process incoming XML payloads directly from untrusted network sources without requiring pre-shared secrets or valid authentication tokens.
- **Incomplete XStream Security Configuration:** Adding allowed types without invoking `NoTypePermission.NONE` leaves XStream's default permissive type resolution active, completely bypassing developer-intended type constraints.
- **Gadget Chain Triggering:** Attackers leverage well-known Java class behaviors during deserialization to transform standard method invocations into arbitrary execution paths within the JVM.
- **Direct Process Invocation:** Object unmarshaling seamlessly cascades into OS-level process execution, enabling arbitrary binary execution, shell commands, or reverse connection establishment.

REMEDIATION & MITIGATION

Organizations operating JetBrains TeamCity On-Premises must take swift and decisive action to address the threat posed by CVE-2026-63077. Because this unauthenticated remote code execution vulnerability is actively exploited in the wild and targetable over network protocols, relying solely on perimeter defenses is insufficient. Securing vulnerable build environments requires a combination of immediate vendor patching, application-level mitigation plugins, and aggressive network defense controls. Prioritizing full server upgrades provides the most complete protection, ensuring that the flawed XStream deserialization logic and underlying endpoints are natively fixed. However, in environments where maintenance windows or release cycles delay immediate full-version updates, security teams must deploy JetBrains' official patch plugins and enforce strict network access policies to block unauthorized access to build agent communication routes.

Key remediation actions and hardening measures include:

- **Upgrade On-Premises Servers Immediately:** Update TeamCity On-Premises installations to patched releases `2025.11.7` or `2026.1.3` (or later) to permanently eliminate the unsafe deserialization flaw.
- **Verify Cloud Instance Safety:** Confirm that managed **TeamCity Cloud** environments are running normally, as JetBrains automatically patched cloud-hosted infrastructure without requiring customer intervention.

- **Apply the Official Security Patch Plugin:** Install JetBrains' standalone **Security Patch Plugin** (compatible with TeamCity versions 2017.1 through 2026.1.2) if an immediate full upgrade cannot be performed due to operational constraints.
- **Utilize Automatic Patching Features:** Fetch and enable the security patch directly from the TeamCity administration console for servers running version `2024.03` or newer.
- **Restrict Internet Exposure:** Remove direct public internet access to TeamCity management interfaces and server endpoints, placing access behind corporate VPNs or Zero-Trust Network Access (ZTNA) solutions.
- **Implement IP Allowlisting for Agents:** Configure firewalls and Web Application Firewalls (WAFs) to restrict access to `/app/agents/*` endpoints strictly to known IP address ranges belonging to legitimate build agents.
- **Audit Web Server Access Logs:** Monitor access and execution logs for anomalous HTTP POST requests directed toward `/app/agents/v1/` originating from unknown or external IP addresses.
- **Rotate Build Secrets and Credentials:** Revoke and reissue cloud provider keys, access tokens, SSH credentials, and service accounts stored within TeamCity if suspicious activity or indicators of compromise are identified.

CONCLUSION

The discovery and active exploitation of CVE-2026-63077 underscore the critical operational risks associated with insecure object deserialization within core enterprise infrastructure. Modern Continuous Integration and Continuous Deployment (CI/CD) engines like JetBrains TeamCity serve as the central orchestration hubs for modern software development pipelines. Consequently, any security flaw that grants unauthenticated remote code execution on these systems effectively surrenders the keys to an organization's digital crown jewels. When adversaries achieve execution privileges on a CI/CD server, the impact extends far beyond local system compromise. Build environments routinely store and process highly sensitive assets, including source code repositories, deployment scripts, cloud provider access tokens, API keys, and SSH credentials. An attacker exploiting CVE-2026-63077 can quietly exfiltrate these secrets, establish persistent backdoors across internal corporate networks, or manipulate build outputs to inject malicious code directly into downstream production releases, resulting in devastating software supply chain attacks. Furthermore, the vulnerability highlights a persistent challenge in software engineering: the subtle security pitfalls of legacy serialization frameworks like XStream. While developers frequently attempt to mitigate deserialization risks using class allowlists, incomplete initialization—such as failing to reset default type permissions—can render defense mechanisms entirely ineffective against modern gadget chains. This flaw serves as a reminder that input validation and object unmarshaling at network boundaries require rigorous defense-in-depth engineering and strict zero-trust principles.

Ultimately, protecting enterprise environments against CVE-2026-63077 demands immediate, coordinated action between security and operational teams. Organizations operating TeamCity On-Premises must prioritize upgrading to patched releases (`2025.11.7` or `2026.1.3`) or deploying official security patch plugins without delay. Combining timely software remediation with robust network segmentation, agent IP allowlisting, and proactive log monitoring remains the only effective strategy to neutralize active threat actors and preserve software supply chain integrity.