Vulnerability Analysis Report — Ivanti API Base64 Deserialization Vector

Author: Daniel C. Ulman for Carson & Saint
Date of Analysis: January 2024
Classification: Vulnerability Research / Potential 0-Day
Status: Confirmed behavior; exploitation depends on deployment configuration. Reported to CTO.
Products Affected: Ivanti Endpoint Manager / Ivanti Connect Secure (various builds)

1. Overview

During routine analysis of Ivanti API behavior, I identified an endpoint that accepts a large JSON POST body containing a Base64-encoded payload representing internal configuration or serialized object data.
The server deserializes and processes this payload without sufficient authorization or validation.
The structure of the request, the size of the encoded object, and the internal metadata discovered during decoding strongly indicate a potential unsafe deserialization or configuration import attack vector, which could allow unauthorized manipulation of system state and, depending on object types processed, potentially lead to remote code execution.
The endpoint’s acceptance of complex, pre-encoded system-level objects is not aligned with expected API behavior for unauthenticated or low-privilege clients.

2. Technical Details
2.1 Request Structure

The vulnerable endpoint accepts requests in the following format:

```
{
  "data": {
    "import": {
      "payload": "ewogICJ0eXBlIjogIkNvbmZpZ3VyYWJsZSIsCiAgIm9iamVjdHMiOiBbCiAgICA..."
    }
  },
  "options": {
    "overwrite": true
  }
}
```

The "payload" field contains a large Base64-encoded object. Typical samples ranged from 20 KB to 80 KB in length.

2.2 Decoded Payload

Decoding the Base64 blob revealed:

Structured configuration objects
Internal metadata fields
References to system components
Nested key/value structures
Serialized entities not intended for direct client submission

Examples of decoded content included:


```
{
  "type": "Configurable",
  "objects": [
    {
      "class": "System.Tasks",
      "id": "internal_policy_001",
      "properties": {
        "exec_path": "/usr/local/ivanti/taskrunner",
        "enabled": true,
        "priority": 10
      }
    }
  ]
}
```


This demonstrates clear access to internal Ivanti object classes — not user-facing data.

3. Impact Analysis
3.1 Unauthorized Deserialization

The server appears to deserialize the embedded object into internal system structures.
If object handlers are improperly validated, this may enable:

Import of unauthorized system configuration
Modification of internal policies
Overwriting sensitive operational metadata
Triggering server-side logic unexpectedly

3.2 Potential for Remote Code Execution

Depending on the allowed object classes and handlers, unsafe deserialization may lead to:

Execution of embedded commands
Instantiation of dangerous object types
Invocation of internal functions
Server-side file writes or script execution
While full RCE depends on environment and configuration, the vector exists.

3.3 Privilege Escalation

Even without RCE, the ability to import arbitrary configuration objects may allow:

Privilege elevation
Bypassing administrative controls
Altering security policies
Enabling or modifying system tasks
This significantly impacts the integrity of affected devices.

4. Attack Scenario

A malicious actor could:

Construct a JSON body containing a crafted Base64 object
Submit it to the vulnerable endpoint
Trigger the server to deserialize internal configuration or executable objects
Manipulate system behavior or escalate privileges
In certain implementations, this may be performed without authentication or with insufficient authorization checks.

5. Detection & Validation Approach

I validated this behavior through a controlled research workflow:

Extracted JSON payload
Decoded Base64 to inspect internal objects
Analyzed object structure and class references
Identified deserialization logic path
Confirmed the endpoint processed the payload
Verified lack of strict authorization checks
Modeled potential exploit paths (config import - unsafe handler - code execution)

This combined analysis strongly supports classification as a high-risk server-side deserialization or configuration import vulnerability.

6. Recommendations

Apply latest vendor patches
Disable or restrict access to import-related endpoints
Review server logs for unexpected import operations
Enforce strict API authentication and authorization

For scanning/detection (SAINT):

Create an HTTP signature that sends a minimal synthetic Base64 object

Observes whether the server accepts the payload, processes/deserializes it,
returns status codes indicating internal object handling, and flag vulnerable versions using known endpoint behaviors

7. Conclusion

The Ivanti API endpoint accepting large Base64-encoded internal objects represents a significant security risk. The combination of:

unauthorized deserialization
insufficient validation

privileged internal object types
potential execution paths
makes this vector suitable for exploitation under realistic conditions.

This vulnerability should be treated as a high-severity issue, especially on Internet-exposed systems.