Configure Secure Access Security Profiles API Advanced

Contents

Issue

When attempting to configure Cisco Secure Access Security Profiles through the public API, users encounter limitations where only basic profile creation with default settings is possible. The API does not expose granular configuration options for advanced security features including:

- TLS decryption settings
- Threat category enforcement controls
- File type blocking and inspection
- Malware inspection settings

Users report receiving "403 Forbidden" errors when attempting to configure decryption settings through standard API calls. The documented public API endpoints appear to lack the detailed Security Profile configuration parameters that are available through the GUI interface, limiting programmatic management capabilities.

Environment

- Cisco Secure Access platform
- Security Profiles API implementation
- All software versions affected
- Public API documentation endpoints

Resolution

The advanced Security Profile configuration capabilities are available through the Cisco Secure Access API, but require specific implementation approaches and field naming conventions that differ from the documented public API examples. These resolution steps address each configuration area:

Basic Security Profile Creation Workflow

The proper workflow for creating Security Profiles with advanced settings follows a two-step process:

Step 1: Create the basic Security Profile using POST.

```
POST /api/v1/security-profiles
{
  "name": "profile_name",
  "description": "profile_description"
}
```

Step 2: Configure advanced options using PUT with the profile ID.

```
PUT /api/v1/security-profiles/{profile_id}
{
  "advanced_settings": {
    "decryption_settings": {...},
    "file_settings": {...},
    "threat_settings": {...}
  }
}
```

File Inspection and File Type Blocking Configuration

File inspection and file type blocking can be configured using these API field names within the PUT request:

- **fileEngines** - Controls file inspection capabilities.
- **fileTypeControlStatus** - Enables or disables file type blocking.

Example configuration for file settings:

```
{
  "fileEngines": "enabled",
  "fileTypeControlStatus": "active",
  "blockedFileTypes": ["exe", "zip", "rar"]
}
```

TLS Decryption Configuration

TLS decryption configuration requires referencing a Do Not Decrypt list ID rather than using a dedicated POST endpoint. The configuration is implemented through the PUT API request using this approach:

Step 1: Identify the Do Not Decrypt list ID from existing configuration.

Step 2: Reference the list ID in the PUT API request.

```
{
  "seetingGroupBypassInspectionGroupID": "do_not_decrypt_list_id",
  "decryptionEnabled": true
}
```

This resolves the "403 Forbidden" error that occurs when attempting to configure decryption through other API methods.

Threat Category Configuration

Threat category configuration is available through an undocumented API endpoint that becomes accessible after successful decryption configuration. The endpoint allows for granular threat category control but requires the Security Profile to be properly configured with decryption settings first.

API Field Name Reference

This table maps common Security Profile features to their corresponding API field names:

Feature	API Field Name	Description
TLS Decryption	seetingGroupBypassInspectionGroupID	References Do Not Decrypt list ID
File Inspection	fileEngines	Controls file inspection capabilities
File Type Blocking	fileTypeControlStatus	Enables file type restrictions

Cause

The limitation was primarily due to incomplete API documentation and non-intuitive field naming conventions rather than actual functional restrictions. The advanced Security Profile configuration capabilities exist within the API but are implemented using:

- Non-obvious field names that differ from GUI terminology
- A two-step workflow (POST for creation, PUT for advanced configuration)
- Dependencies on existing configuration objects (such as Do Not Decrypt lists)
- Undocumented API endpoints for certain features

The public API documentation does not clearly expose these implementation details, leading to the perception of API limitations when the functionality is actually available through alternative approaches.

Related Content

- Cisco Secure Access Security Profiles API Documentation
- Cisco Technical Support & Downloads