DFS by DreamGRC: API Documentation

Table of Contents

1. Introduction
2. Authentication
3. API Endpoints
4. Case Management
5. Evidence Management
6. Analysis Management
7. Incident Reconstruction
8. Correlation Analysis
9. Report Generation
10. Request/Response Format
11. Error Handling
12. API Usage Examples

Introduction

The DFS by DreamGRC API provides programmatic access to all platform functionality, enabling you to: - Automate digital forensic investigations - Integrate with your existing security tools - Create custom forensic workflows - Build dashboards and reporting systems

This API follows RESTful design principles with JSON as the primary data exchange format.

Authentication

Authentication with the API can be done in two ways:

Session-Based Authentication (Current Implementation)

The current implementation uses session-based authentication. You need to:

1. Use the login endpoint to authenticate
2. Maintain a session cookie for subsequent requests

```
POST /api/v1/auth/login
Content-Type: application/json

{
  "username": "your_username",
  "password": "your_password"
}
```

After successful login, your browser will store a session cookie that will be sent with subsequent requests.

API Key Authentication (Coming Soon)

In future releases, all API requests will support authentication using API keys included in the Authorization header:

```
Authorization: Bearer YOUR_API_KEY
```

When available, API keys can be obtained through: 1. Logging in to your DFS by DreamGRC account 2. Navigating to Settings > API Keys 3. Clicking "Generate New API Key"

Basic Authentication

For initial integration and testing, you can also use basic authentication:

```
POST /api/v1/auth/login
Content-Type: application/json

```

```
{
  "username": "your_username",
  "password": "your_password"
}
```

Response:

```
{
  "success": true,
  "user_id": 123,
  "username": "your_username",
  "is_admin": false
}
```

API Endpoints

Case Management

List all cases

```
GET /api/v1/cases
```

Response:

```
[
  {
    "id": 123,
    "name": "Network Intrusion Investigation",
    "description": "Investigation into unauthorized access to
    "status": "open",
    "priority": "high",
    "created_at": "2025-03-15T14:22:31Z",
    "evidence_count": 12
  },
  {...}
]
```

Get case details

```
GET /api/v1/cases/{case_id}
```

Response:

```
{
  "id": 123,
  "name": "Network Intrusion Investigation",
  "description": "Investigation into unauthorized access to f
  "status": "open",
  "priority": "high",
  "created_at": "2025-03-15T14:22:31Z",
  "evidence_files": [
    {
      "id": 456,
      "filename": "server_memory.raw",
      "file_type": "memory_dump",
      "file_size": 8589934592,
      "md5_hash": "d41d8cd98f00b204e9800998ecf8427e",
      "sha1_hash": "da39a3ee5e6b4b0d3255bfef95601890afd80709"
      "sha256_hash": "e3b0c44298fc1c149afbf4c8996fb92427ae41e
      "upload_date": "2025-03-15T14:30:45Z"
    },
    {...}
  ],
  "analysis_results": [
    {
      "id": 789,
      "evidence_id": 456,
      "tool_name": "volatility",
      "command": "process_list",
      "execution_time": 12.8,
      "status": "completed",
      "created_at": "2025-03-15T14:35:22Z"
    },
    {...}
  ]
}
```

Create a new case

```
POST /api/v1/cases/create
Content-Type: application/json

{
  "name": "Ransomware Investigation",
  "description": "Investigation into ransomware attack on HR
  "status": "open",
  "priority": "critical"
}
```

Response:

```
{
  "success": true,
  "id": 124,
  "name": "Ransomware Investigation"
}
```

Evidence Management

Upload evidence to a case

```
POST /api/v1/evidence/upload
Content-Type: multipart/form-data
```

```
case_id=123
file=@/path/to/local/file.pcap
```

Response:

```
{
  "success": true,
  "id": 457,
  "filename": "network_capture.pcap",
  "file_type": "network_capture",
  "file_size": 1245184,
  "file_hash_md5": "7d793037a0760186574b0282f2f435e7"
}
```

Get evidence metadata

```
GET /api/v1/evidence/{evidence_id}
```

Response:

```
{
  "id": 457,
  "case_id": 123,
  "filename": "network_capture.pcap",
  "file_type": "network_capture",
  "file_size": 1245184,
  "file_hash_md5": "7d793037a0760186574b0282f2f435e7",
  "file_hash_sha1": "2fd4e1c67a2d28fced849ee1bb76e7391b93eb12
  "file_hash_sha256": "9f86d081884c7d659a2feaa0c55ad015a3bf4f
  "upload_date": "2025-03-16T09:12:33Z"
}
```

Download evidence file

```
GET /api/v1/evidence/{evidence_id}/download
```

Response: The evidence file as a downloadable attachment.

Analysis Management

Analyze an evidence file

```
POST /api/v1/analysis/analyze/{evidence_id}
Content-Type: application/json

{
  "tool_name": "volatility",
  "command": "process_list"
}
```

Response:

```
{
  "success": true,
  "result_id": 790,
  "tool_name": "volatility",
  "command": "process_list",
  "status": "running"
}
```

Get analysis result

```
GET /api/v1/analysis/result/{result_id}
```

Response:

```
{
  "id": 790,
  "evidence_id": 456,
  "case_id": 123,
  "tool_name": "volatility",
  "command": "process_list",
  "execution_time": 14.2,
  "status": "completed",
  "created_at": "2025-03-16T10:15:22Z",
  "result_data": {
    "processes": [
      {
        "pid": 4,
        "ppid": 0,
        "name": "System",
        "start_time": "2025-03-10T08:15:32Z",
        "path": ""
      },
      {
        "pid": 1234,
        "ppid": 788,
        "name": "suspicious.exe",
        "start_time": "2025-03-15T14:22:31Z",
        "path": "C:\\Windows\\Temp\\suspicious.exe"
      },
      {...}
    ],
    "suspicious_indicators": [
      "Process suspicious.exe running from Temp directory",
      "Unusual parent-child relationship for PID 1234"
    ],
    "summary": "Memory analysis identified 127 running proces
  }
}
```

Validate analysis capabilities

```
POST /api/v1/analysis/validate
Content-Type: multipart/form-data

file=@/path/to/local/file.mem
file_type=memory_dump
tool_name=volatility
command=process_list
```

Response:

```
{
  "status": "success",
  "capabilities": {
    "tools_available": ["volatility", "wireshark", "sleuthkit
    "memory_analysis": true,
    "network_analysis": true,
    "file_system_analysis": true,
    "log_analysis": true
  },
  "sample_result": {
    "process_count": 42,
    "network_connections": 15,
    "suspicious_processes": 2
  },
  "validation": {
    "timestamp": "2025-03-16T11:30:22Z",
    "original_filename": "sample.mem",
    "tool_name": "volatility",
    "command": "process_list"
  }
}
```

Incident Reconstruction

List all workflows

```
GET /api/v1/incident/workflows?case_id=123
```

Response:

```
[
  {
    "id": 456,
    "name": "Network Intrusion Analysis",
    "description": "Workflow to analyze network intrusion in
    "status": "in_progress",
    "current_step": 3,
    "case_id": 123,
    "created_at": "2025-03-18T09:05:12Z",
    "updated_at": "2025-03-18T14:33:27Z",
    "progress": "3/7"
  },
  {...}
]
```

Get workflow details

```
GET /api/v1/incident/workflows/{workflow_id}
```

Response:

```
{
  "id": 456,
  "name": "Network Intrusion Analysis",
  "description": "Workflow to analyze network intrusion in fi
  "status": "in_progress",
  "current_step": 3,
  "case_id": 123,
  "created_at": "2025-03-18T09:05:12Z",
  "updated_at": "2025-03-18T14:33:27Z",
  "steps": [
    {
      "id": 1001,
      "step_number": 1,
      "name": "Evidence Collection",
      "description": "Gather all relevant memory dumps, netwo
      "status": "completed",
      "start_time": "2025-03-18T09:15:32Z",
      "end_time": "2025-03-18T10:22:15Z",
      "notes": "Collected 3 memory dumps, 2 network captures,
    },
    {
      "id": 1002,
      "step_number": 2,
      "name": "Memory Analysis",
      "description": "Analyze memory dumps to identify runnin
      "status": "completed",
      "start_time": "2025-03-18T10:30:00Z",
      "end_time": "2025-03-18T12:15:45Z",
      "notes": "Identified suspicious process PID 1234 with u
    },
    {
      "id": 1003,
      "step_number": 3,
      "name": "Network Traffic Analysis",
      "description": "Analyze network captures for suspicious
      "status": "in_progress",
      "start_time": "2025-03-18T13:05:22Z",
      "end_time": null,
      "notes": "Investigating connections to IP 203.0.113.42"
    },
    {...}
  ],
  "analysis_results": [
    {
      "id": 790,
      "evidence_id": 456,
      "evidence_name": "server_memory.raw",
      "tool_name": "volatility",
      "command": "process_list",
      "status": "completed",
      "execution_time": 14.2,
      "created_at": "2025-03-18T10:35:22Z"
    },
    {...}
  ]
}
```

Create a new workflow

```
POST /api/v1/incident/workflows/create
Content-Type: application/json

{
  "case_id": 123,
  "name": "Malware Investigation Workflow",
  "description": "Analysis of malware found on finance server
  "template": "malware_analysis"
}
```

Response:

```
{
  "success": true,
  "id": 457,
  "name": "Malware Investigation Workflow",
  "case_id": 123,
  "status": "not_started",
  "current_step": 1,
  "steps": [
    {
      "id": 1010,
      "step_number": 1,
      "name": "Initial Triage",
      "description": "Initial assessment of suspected malware
      "status": "not_started"
    },
    {
      "id": 1011,
      "step_number": 2,
      "name": "Static Analysis",
      "description": "Perform static analysis of malware samp
      "status": "not_started"
    },
    {...}
  ]
}
```

Update workflow step

```
PUT /api/v1/incident/workflows/{workflow_id}/steps/{step_id}
Content-Type: application/json

{
  "status": "completed",
  "notes": "Initial triage confirmed presence of ransomware v
}
```

Response:

```
{
  "success": true,
  "id": 1010,
  "step_number": 1,
  "name": "Initial Triage",
  "status": "completed",
  "notes": "Initial triage confirmed presence of ransomware v
}
```

Analyze evidence in workflow

```
POST /api/v1/incident/workflows/{workflow_id}/analyze
Content-Type: application/json

{
  "evidence_id": 456,
  "tool_name": "volatility",
  "command": "process_list",
  "step_id": 1003
}
```

Response:

```
{
  "success": true,
  "result_id": 791,
  "tool_name": "volatility",
  "command": "process_list",
  "status": "running"
}
```

Correlation Analysis

Run correlation analysis on evidence

```
POST /api/v1/incident/workflows/{workflow_id}/correlate
Content-Type: application/json

{
  "methods": ["temporal", "ioc", "behavior"],
  "min_confidence": 0.7,
  "step_id": 1004
}
```

Response:

```
{
  "success": true,
  "correlations_found": 12,
  "correlations": [
    {
      "correlation_type": "temporal",
      "confidence": 0.85,
      "source_type": "memory_dump",
      "target_type": "log_file"
    },
    {
      "correlation_type": "ioc",
      "confidence": 0.92,
      "source_type": "memory_dump",
      "target_type": "network_capture"
    },
    {...}
  ]
}
```

Report Generation

Generate workflow report

```
POST /api/v1/incident/workflows/{workflow_id}/report
Content-Type: application/json

{
  "title": "Network Intrusion Investigation Report",
  "report_type": "detailed",
  "format": "pdf",
  "include_timeline": true,
  "include_ioc": true,
  "step_id": 1007
}
```

Response:

```
{
  "success": true,
  "report_id": 345,
  "title": "Network Intrusion Investigation Report",
  "format": "pdf",
  "created_at": "2025-03-20T15:45:22Z"
}
```

Request/Response Format

General Format

All API responses follow a consistent JSON format:

Success Response:

```
{
  "success": true,
  ... (additional data)
}
```

Error Response:

```
{
  "error": "Error message describing what went wrong"
}
```

Pagination

Endpoints that return multiple items support pagination:

```
GET /api/v1/cases?page=2&per_page=20
```

The response includes pagination metadata:

```
{
  "data": [...],
  "pagination": {
    "page": 2,
    "per_page": 20,
    "total_items": 156,
    "total_pages": 8
  }
}
```

Filtering and Sorting

Most list endpoints support filtering and sorting:

```
GET /api/v1/cases?status=open&priority=high&sort=created_at:d
```

Error Handling

HTTP Status Codes

The API uses standard HTTP status codes:

- **200 OK**: The request was successful
- **400 Bad Request**: The request was malformed or missing required parameters
- **401 Unauthorized**: Authentication failed or credentials not provided

- **403 Forbidden**: Authenticated user doesn't have permission for the operation
- **404 Not Found**: The requested resource doesn't exist
- **429 Too Many Requests**: Rate limit exceeded
- **500 Internal Server Error**: Server-side error

Error Response Format

```
{
  "error": "Detailed error message",
  "error_code": "ERROR_CODE",
  "details": {
    "field_name": "Specific field error message"
  }
}
```

API Usage Examples

Complete Investigation Workflow

This example shows how to automate a complete investigation workflow:

1. Create a new case

```
POST /api/v1/cases/create
Content-Type: application/json

{
  "name": "Suspicious Activity Investigation",
  "description": "Investigating suspicious activity on web se
  "priority": "high"
}
```

1. Upload evidence file

```
POST /api/v1/evidence/upload
Content-Type: multipart/form-data

case_id=125
file=@/path/to/server.mem
```

1. Create incident reconstruction workflow

```
POST /api/v1/incident/workflows/create
Content-Type: application/json

{
  "case_id": 125,
  "name": "Web Server Investigation",
  "template": "intrusion_analysis"
}
```

1. Analyze memory dump

```
POST /api/v1/incident/workflows/{workflow_id}/analyze
Content-Type: application/json

{
  "evidence_id": 460,
  "tool_name": "volatility",
  "command": "process_list",
  "step_id": 1020
}
```

1. Run correlation analysis

```
POST /api/v1/incident/workflows/{workflow_id}/correlate
Content-Type: application/json

{
  "methods": ["temporal", "ioc", "behavior"],
  "step_id": 1022
}
```

1. Generate detailed report

```
POST /api/v1/incident/workflows/{workflow_id}/report
Content-Type: application/json

{
  "title": "Web Server Intrusion Analysis",
  "report_type": "detailed",
  "format": "pdf",
  "include_timeline": true,
  "include_ioc": true,
  "step_id": 1024
}
```

This workflow can be fully automated using the API, allowing for integration with SOC platforms, SIEM systems, or custom security tools.

API Integration for SOC Teams

SOC teams can integrate the DFS by DreamGRC API with their existing tools and workflows:

- **SIEM Integration**: Automatically create cases and upload evidence when alerts meet certain criteria
- **Ticketing System Integration**: Update tickets with forensic findings
- **Automated Reporting**: Generate scheduled compliance reports
- **Custom Dashboards**: Build real-time monitoring dashboards using the API

For detailed integration examples and code samples, please contact the DFS by DreamGRC support team.