ATT&CK Scripts

Navigator Layer Update

Caleb Little

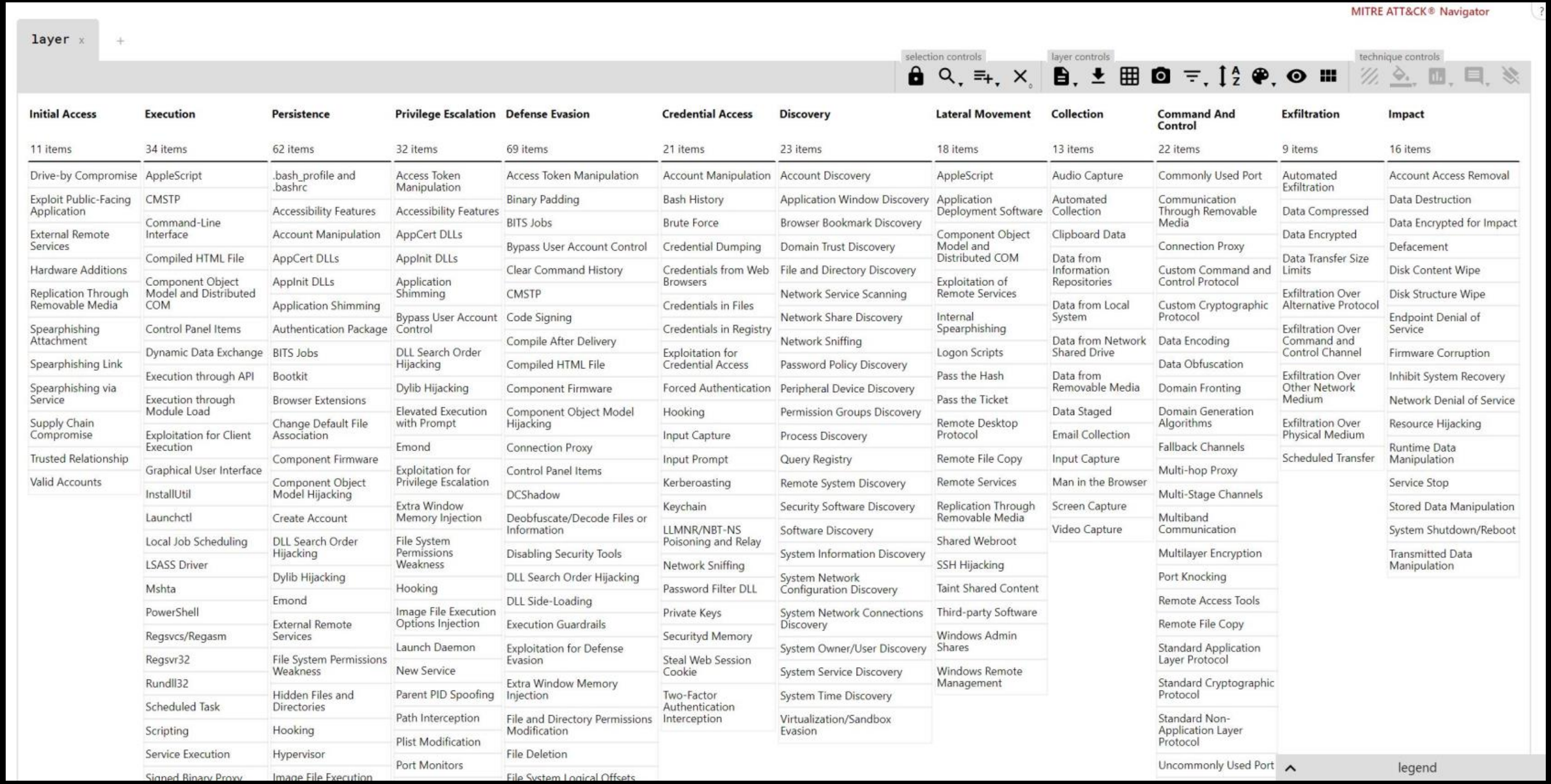
@MITREattack

ATT&CK Scripts

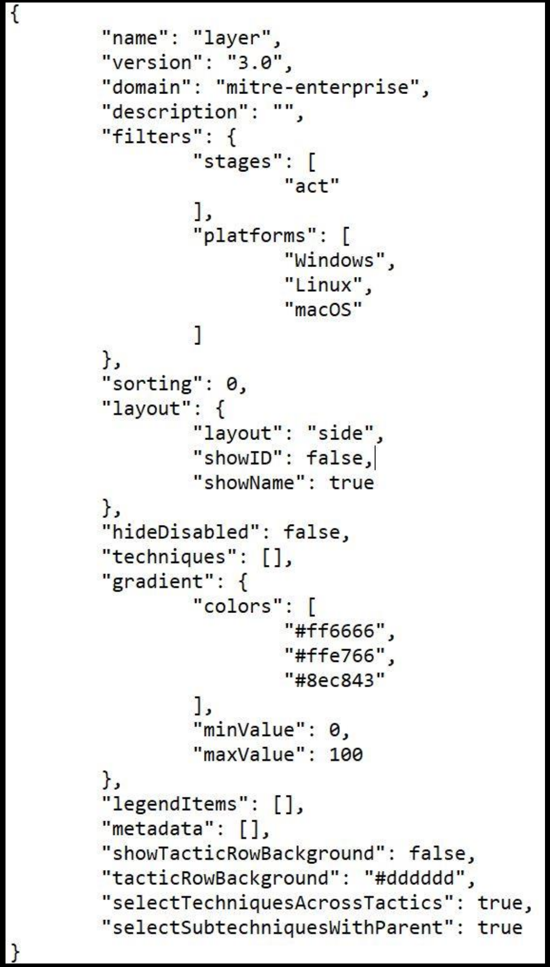
- **Python scripts to improve user interaction with ATT&CK content**
  - Compare two versions of ATT&CK
  - List and visualize techniques associated with a data source
  - Jupyter Notebooks for ATT&CK Training
  - ATT&CK Navigator layer utilities (NEW)

https://github.com/mitre-attack/attack-scripts

ATT&CK Navigator Layers



Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command And Control	Exfiltration	Impact
11 items	34 items	62 items	32 items	69 items	21 items	23 items	18 items	13 items	22 items	9 items	16 items
Drive-by Compromise	AppleScript	.bash_profile and .bashrc	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery	AppleScript	Audio Capture	Commonly Used Port	Automated Exfiltration	Account Access Removal
Exploit Public-Facing Application	CMSTP	Accessibility Features	Accessibility Features	Binary Padding	Bash History	Application Window Discovery	Application Deployment Software	Automated Collection	Communication Through Removable Media	Data Compressed	Data Destruction
External Remote Services	Command-Line Interface	Account Manipulation	AppCert DLLs	BITS Jobs	Brute Force	Browser Bookmark Discovery	Component Object Model and Distributed COM	Clipboard Data	Connection Proxy	Data Encrypted	Data Encrypted for Impact
Hardware Additions	Compiled HTML File	AppCert DLLs	AppInit DLLs	Bypass User Account Control	Credential Dumping	Domain Trust Discovery	Exploitation of Remote Services	Data from Information Repositories	Custom Command and Control Protocol	Data Transfer Size Limits	Defacement
Replication Through Removable Media	Component Object Model and Distributed COM	AppInit DLLs	Application Shimming	Clear Command History	Credentials from Web Browsers	File and Directory Discovery	Internal Spearphishing	Data from Local System	Custom Cryptographic Protocol	Exfiltration Over Alternative Protocol	Disk Content Wipe
Spearphishing Attachment	Control Panel Items	Application Shimming	Bypass User Account Control	CMSTP	Credentials in Files	Network Service Scanning	Logon Scripts	Data from Network Shared Drive	Data Encoding	Exfiltration Over Command and Control Channel	Disk Structure Wipe
Spearphishing Link	Dynamic Data Exchange	Authentication Package	DLL Search Order Hijacking	Code Signing	Credentials in Registry	Network Share Discovery	Pass the Hash	Data from Removable Media	Data Obfuscation	Exfiltration Over Other Network Medium	Endpoint Denial of Service
Spearphishing via Service	Execution through API	BITS Jobs	Dylib Hijacking	Compile After Delivery	Exploitation for Credential Access	Network Sniffing	Pass the Ticket	Data Staged	Domain Fronting	Exfiltration Over Physical Medium	Firmware Corruption
Supply Chain Compromise	Execution through Module Load	Bootkit	Elevated Execution with Prompt	Compiled HTML File	Forced Authentication	Password Policy Discovery	Remote Desktop Protocol	Email Collection	Domain Generation Algorithms	Scheduled Transfer	Inhibit System Recovery
Trusted Relationship	Exploitation for Client Execution	Browser Extensions	Emond	Component Firmware	Hooking	Peripheral Device Discovery	Remote File Copy	Input Capture	Fallback Channels		Network Denial of Service
Valid Accounts	Graphical User Interface	Change Default File Association	Exploitation for Privilege Escalation	Component Object Model Hijacking	Input Capture	Permission Groups Discovery	Remote Services	Man in the Browser	Multi-hop Proxy		Resource Hijacking
	InstallUtil	Component Firmware	Extra Window Memory Injection	Connection Proxy	Input Prompt	Process Discovery	Replication Through Removable Media	Screen Capture	Multi-Stage Channels		Runtime Data Manipulation
	Launchctl	Component Object Model Hijacking	File System Permissions Weakness	Control Panel Items	Kerberoasting	Query Registry	Shared Webroot	Video Capture	Multiband Communication		Service Stop
	Local Job Scheduling	Create Account	Hooking	DCShadow	Keychain	Remote System Discovery	SSH Hijacking		Multilayer Encryption		Stored Data Manipulation
	LSASS Driver	DLL Search Order Hijacking	Image File Execution Options Injection	Deobfuscate/Decode Files or Information	LLMNR/NBT-NS Poisoning and Relay	Security Software Discovery	Taint Shared Content		Port Knocking		System Shutdown/Reboot
	Mshta	Dylib Hijacking	Launch Daemon	Disabling Security Tools	Network Sniffing	Software Discovery	Third-party Software		Remote Access Tools		Transmitted Data Manipulation
	PowerShell	Emond	New Service	DLL Search Order Hijacking	Password Filter DLL	System Information Discovery	Windows Admin Shares		Remote File Copy		
	Regsvcs/Regasm	External Remote Services	Parent PID Spoofing	DLL Side-Loading	Private Keys	System Network Configuration Discovery	Windows Remote Management		Standard Application Layer Protocol		
	Regsvr32	File System Permissions Weakness	Path Interception	Execution Guardrails	Securityd Memory	System Network Connections Discovery			Standard Cryptographic Protocol		
	Rundll32	Hidden Files and Directories	Plist Modification	Exploitation for Defense Evasion	Steal Web Session Cookie	System Owner/User Discovery			Standard Non-Application Layer Protocol		
	Scheduled Task	Hooking	Port Monitors	Extra Window Memory Injection	Two-Factor Authentication Interception	System Service Discovery			Uncommonly Used Port		
	Scripting	Hypervisor		File and Directory Permissions Modification		System Time Discovery					
	Service Execution	Image File Execution		File Deletion		Virtualization/Sandbox Evasion					
	Signed Binary Proxy			File System Logical Offsets							



```
{
        "name": "layer",
        "version": "3.0",
        "domain": "mitre-enterprise",
        "description": "",
        "filters": {
                "stages": [
                        "act"
                ],
                "platforms": [
                        "Windows",
                        "Linux",
                        "macOS"
                ]
        },
        "sorting": 0,
        "layout": {
                "layout": "side",
                "showID": false,
                "showName": true
        },
        "hideDisabled": false,
        "techniques": [],
        "gradient": {
                "colors": [
                        "#ff6666",
                        "#ffe766",
                        "#8ec843"
                ],
                "minValue": 0,
                "maxValue": 100
        },
        "legendItems": [],
        "metadata": [],
        "showTacticRowBackground": false,
        "tacticRowBackground": "#dddddd",
        "selectTechniquesAcrossTactics": true,
        "selectSubtechniquesWithParent": true
}
```

Master: https://mitre-attack.github.io/attack-navigator/

Sub-techniques Beta: https://mitre-attack.github.io/attack-navigator/beta/

Navigator Layer Scripts

- **Overall objective:** Make it easier to generate and manipulate layers programmatically

- **Core module**
  - Python classes facilitating Layer IO and format validation
  - Uses Layer File Format Version 3.0 (sub-techniques Beta support)

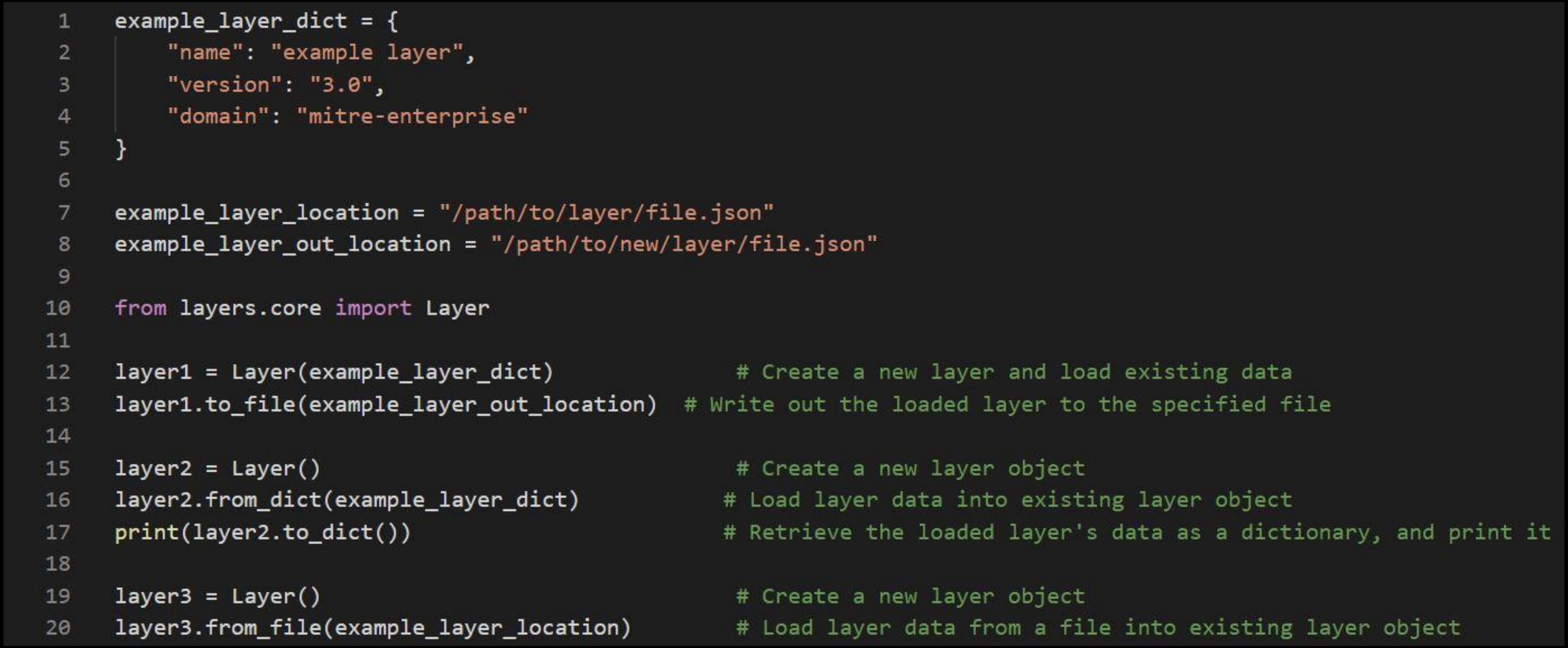
- **Manipulators module**
  - LayerOps manipulator allows lambda manipulations of layer files

More coming soon…

Layer IO Methods

Method [x = Layer()]	Functionality
x.from_str()	Loads layer from string
x.from_dict()	Loads layer from dictionary
x.from_file()	Loads layer from filepath
x.to_file()	Saves layer to filepath
x.to_dict()	Retrieves dictionary representation
x.to_str()	Retrieves string representation

Examples – Layer IO



```python
example_layer_dict = {
    "name": "example layer",
    "version": "3.0",
    "domain": "mitre-enterprise"
}

example_layer_location = "/path/to/layer/file.json"
example_layer_out_location = "/path/to/new/layer/file.json"

from layers.core import Layer

layer1 = Layer(example_layer_dict)              # Create a new layer and load existing data
layer1.to_file(example_layer_out_location)  # Write out the loaded layer to the specified file

layer2 = Layer()                                # Create a new layer object
layer2.from_dict(example_layer_dict)           # Load layer data into existing layer object
print(layer2.to_dict())                        # Retrieve the loaded layer's data as a dictionary, and print it

layer3 = Layer()                                # Create a new layer object
layer3.from_file(example_layer_location)        # Load layer data from a file into existing layer object
```

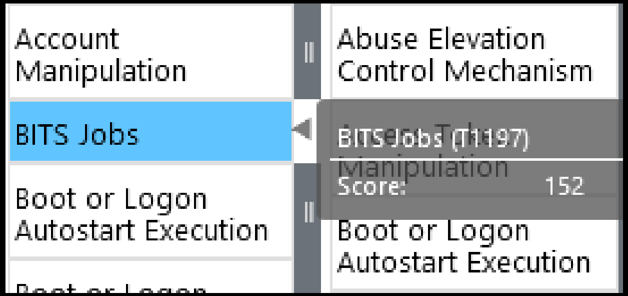
LayerOps

- **Python implementation of existing Navigator Interface** ("create Layer from other layers")
- **Allows users to combine layer files**
- **Example use cases:**
  – Average scores between multiple layers
  – Concatenate comments
  – Boolean operation, e.g output 1 where the score is >75

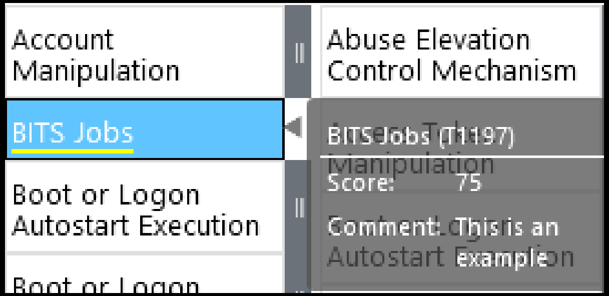
LayerOps

- **User inputs lambda functions to define operations**
- **LayerOps is a python class**
  - Instantiate a LayerOps with a defined operation, e.g ”average scores”
  - Run same LayerOps instance on different sets of layers, e.g average layers A, B, and then separately X, Y, Z with the same LayerOps instance

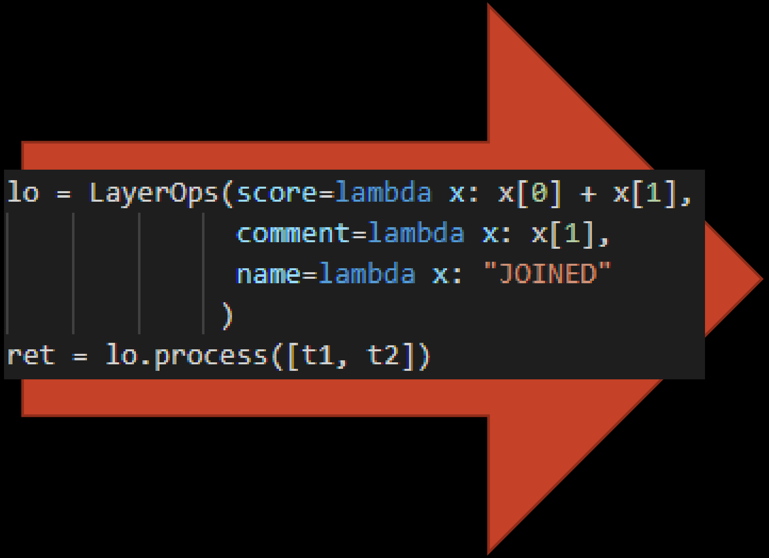
Visual Example


Account Manipulation
BITS Jobs
Boot or Logon Autostart Execution
Abuse Elevation Control Mechanism
Boot or Logon Autostart Execution

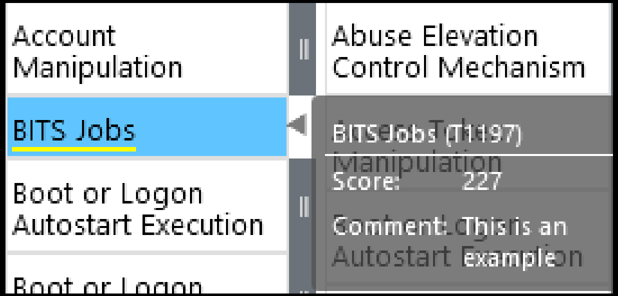
BITS Jobs (T1197)
Score: 152


Account Manipulation
BITS Jobs
Boot or Logon Autostart Execution
Abuse Elevation Control Mechanism

BITS Jobs (T1197)
Score: 75
Comment: This is an example



```
lo = LayerOps(score=lambda x: x[0] + x[1],
              comment=lambda x: x[1],
              name=lambda x: "JOINED"
             )
ret = lo.process([t1, t2])
```


Account Manipulation
BITS Jobs
Boot or Logon Autostart Execution
Abuse Elevation Control Mechanism

BITS Jobs (T1197)
Score: 227
Comment: This is an example

LayerOps API

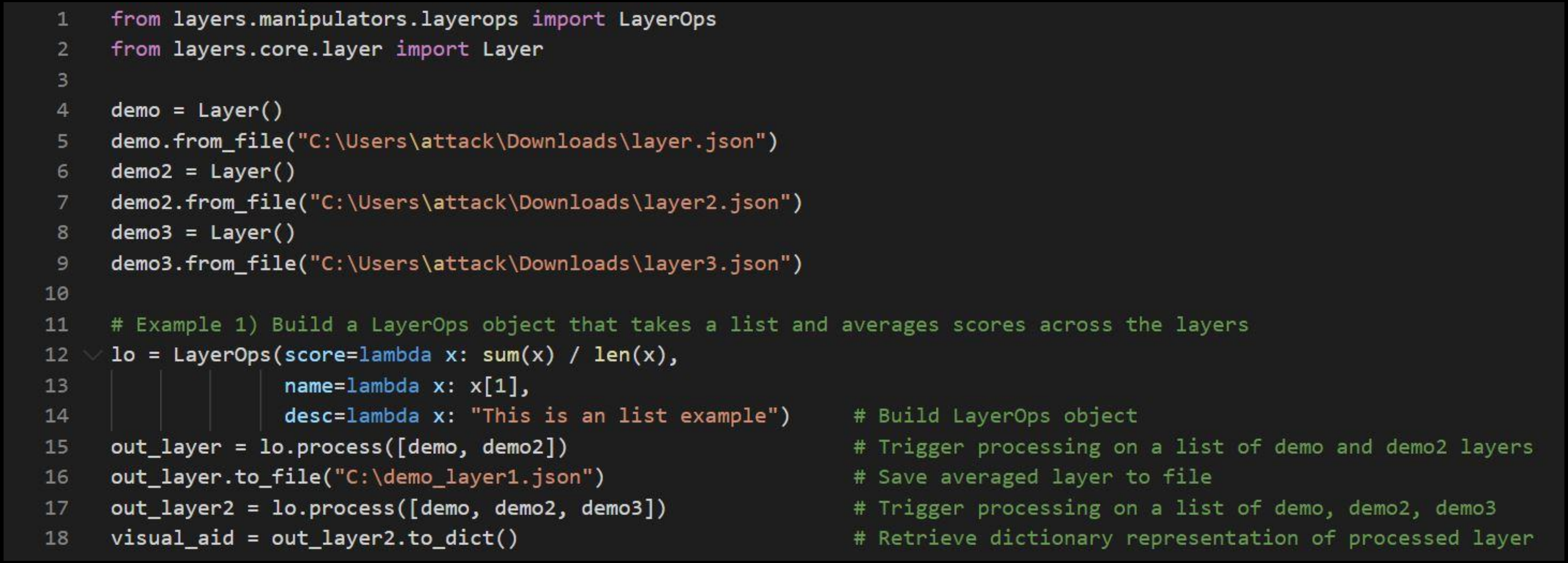
Constructor Lambda Inputs	Functionality
Score	Processes Technique Scores
Comment	Processes Technique Comments
Enabled	Processes Technique Enabled Status
Colors	Processes Technique Colors
Metadata	Processes Technique Metadata (Metadata objects)
Name	Processes Layer Name
Description	Processes Layer Description

Example Usage	Functionality
x = LayerOps(score=lambda x: …)	Defines Operating Lambdas
x.process([layer1, layer2…])	Applies lambda to input

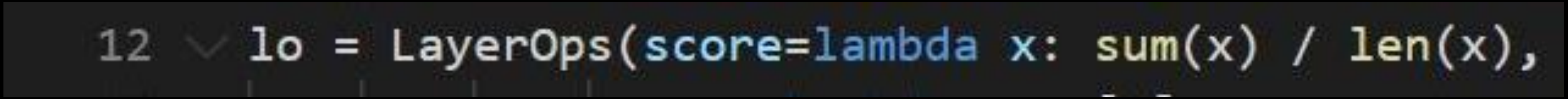
ATT&CK™

MITRE

Examples – LayerOps (1)

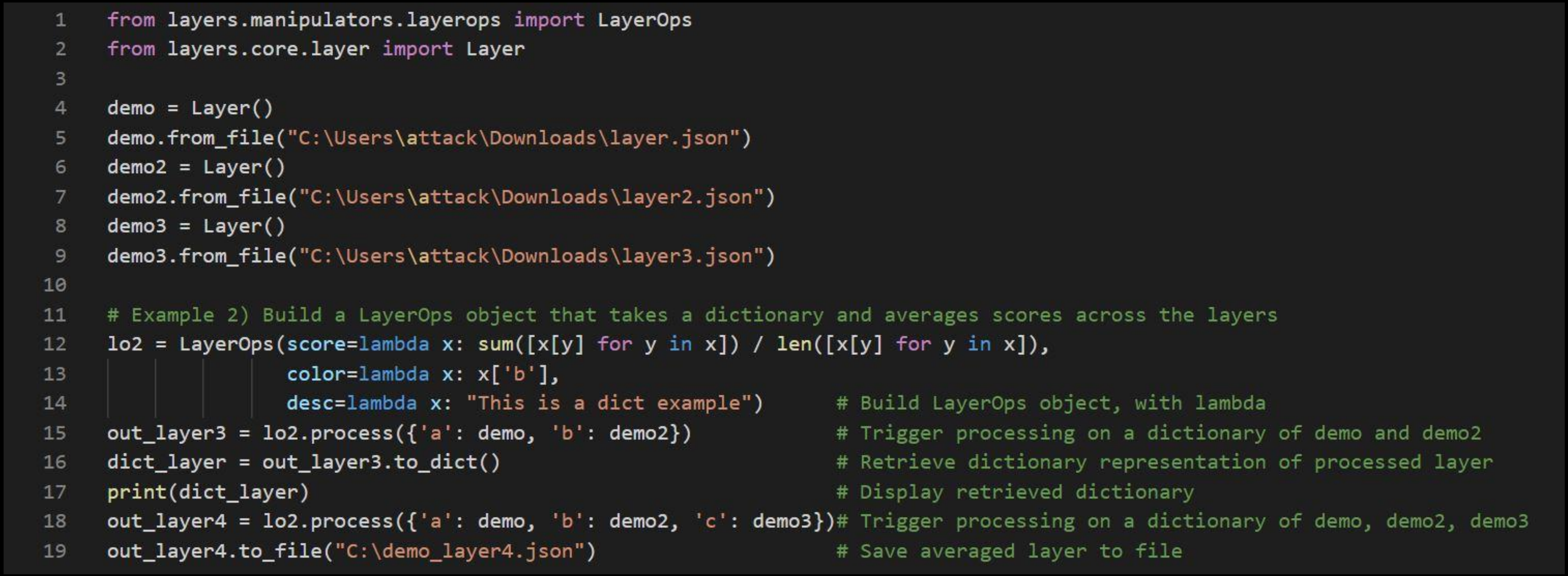


```python
from layers.manipulators.layerops import LayerOps
from layers.core.layer import Layer

demo = Layer()
demo.from_file("C:\Users\attack\Downloads\layer.json")
demo2 = Layer()
demo2.from_file("C:\Users\attack\Downloads\layer2.json")
demo3 = Layer()
demo3.from_file("C:\Users\attack\Downloads\layer3.json")

# Example 1) Build a LayerOps object that takes a list and averages scores across the layers
lo = LayerOps(score=lambda x: sum(x) / len(x),
              name=lambda x: x[1],
              desc=lambda x: "This is an list example")     # Build LayerOps object
out_layer = lo.process([demo, demo2])                        # Trigger processing on a list of demo and demo2 layers
out_layer.to_file("C:\demo_layer1.json")                     # Save averaged layer to file
out_layer2 = lo.process([demo, demo2, demo3])                # Trigger processing on a list of demo, demo2, demo3
visual_aid = out_layer2.to_dict()                            # Retrieve dictionary representation of processed layer
```



```python
lo = LayerOps(score=lambda x: sum(x) / len(x),
```

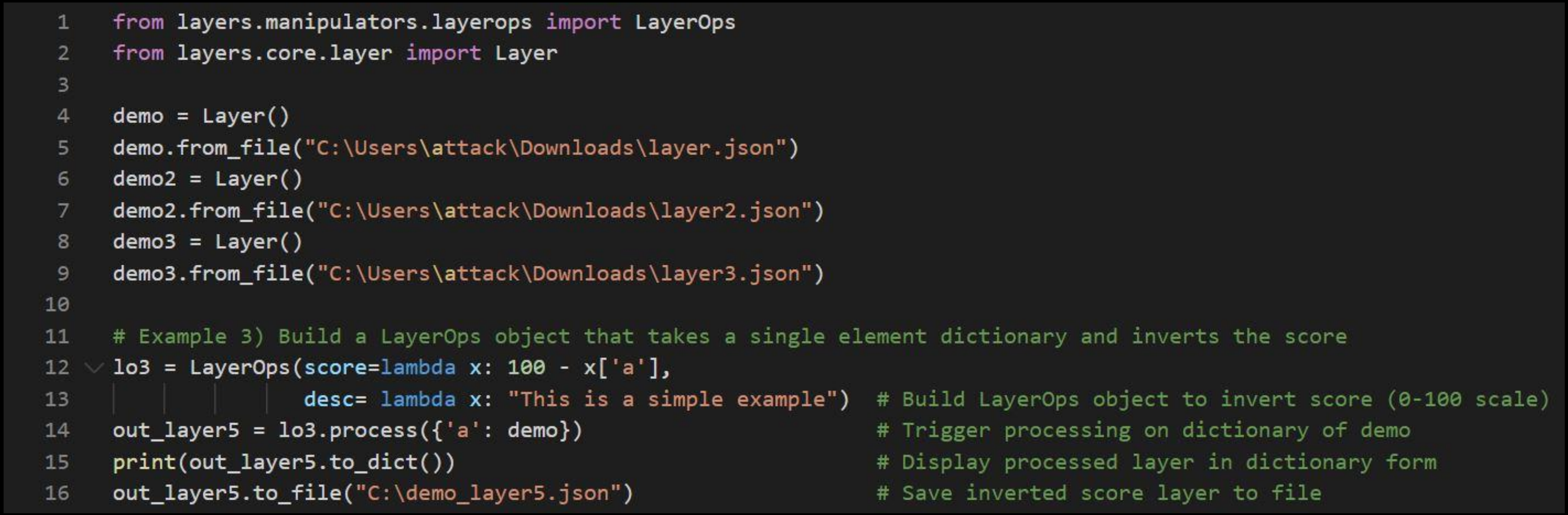
Examples – LayerOps (2)



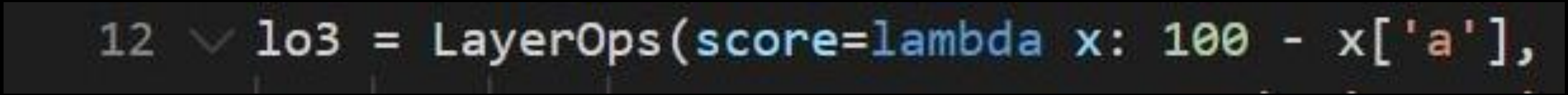
```python
from layers.manipulators.layerops import LayerOps
from layers.core.layer import Layer

demo = Layer()
demo.from_file("C:\Users\attack\Downloads\layer.json")
demo2 = Layer()
demo2.from_file("C:\Users\attack\Downloads\layer2.json")
demo3 = Layer()
demo3.from_file("C:\Users\attack\Downloads\layer3.json")

# Example 2) Build a LayerOps object that takes a dictionary and averages scores across the layers
lo2 = LayerOps(score=lambda x: sum([x[y] for y in x]) / len([x[y] for y in x]),
               color=lambda x: x['b'],
               desc=lambda x: "This is a dict example")      # Build LayerOps object, with lambda
out_layer3 = lo2.process({'a': demo, 'b': demo2})            # Trigger processing on a dictionary of demo and demo2
dict_layer = out_layer3.to_dict()                            # Retrieve dictionary representation of processed layer
print(dict_layer)                                            # Display retrieved dictionary
out_layer4 = lo2.process({'a': demo, 'b': demo2, 'c': demo3})# Trigger processing on a dictionary of demo, demo2, demo3
out_layer4.to_file("C:\demo_layer4.json")                    # Save averaged layer to file
```

LayerOps(score=lambda x: sum([x[y] for y in x]) / len([x[y] for y in x]),

Examples – LayerOps (3)

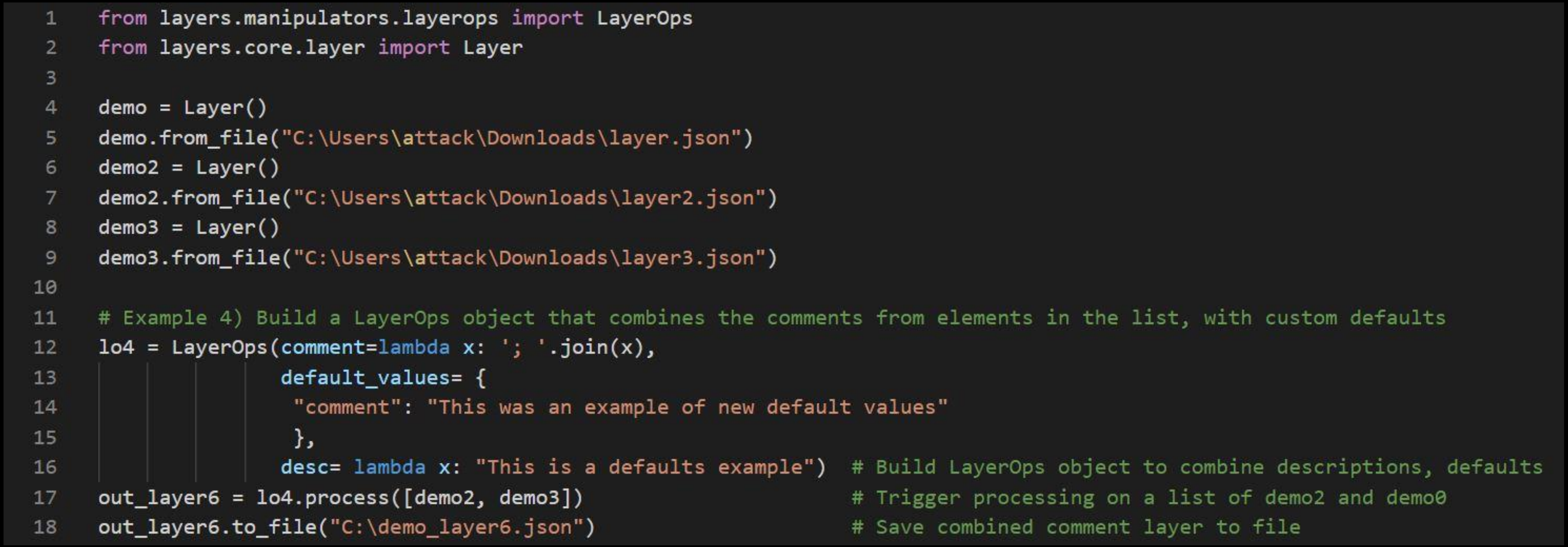


```python
from layers.manipulators.layerops import LayerOps
from layers.core.layer import Layer

demo = Layer()
demo.from_file("C:\Users\attack\Downloads\layer.json")
demo2 = Layer()
demo2.from_file("C:\Users\attack\Downloads\layer2.json")
demo3 = Layer()
demo3.from_file("C:\Users\attack\Downloads\layer3.json")

# Example 3) Build a LayerOps object that takes a single element dictionary and inverts the score
lo3 = LayerOps(score=lambda x: 100 - x['a'],
               desc= lambda x: "This is a simple example")  # Build LayerOps object to invert score (0-100 scale)
out_layer5 = lo3.process({'a': demo})                       # Trigger processing on dictionary of demo
print(out_layer5.to_dict())                                 # Display processed layer in dictionary form
out_layer5.to_file("C:\demo_layer5.json")                   # Save inverted score layer to file
```

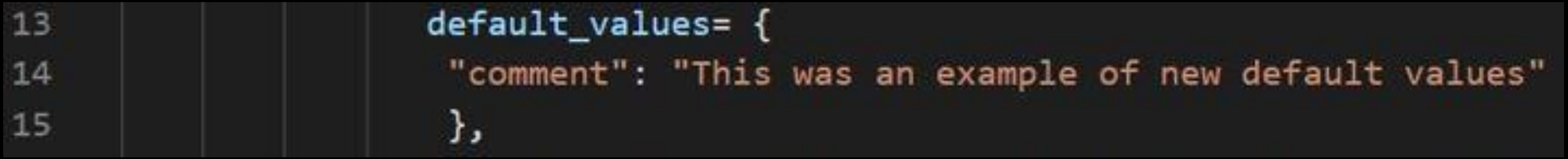


```python
lo3 = LayerOps(score=lambda x: 100 - x['a'],
```

Examples – LayerOps (4)



```python
from layers.manipulators.layerops import LayerOps
from layers.core.layer import Layer

demo = Layer()
demo.from_file("C:\Users\attack\Downloads\layer.json")
demo2 = Layer()
demo2.from_file("C:\Users\attack\Downloads\layer2.json")
demo3 = Layer()
demo3.from_file("C:\Users\attack\Downloads\layer3.json")

# Example 4) Build a LayerOps object that combines the comments from elements in the list, with custom defaults
lo4 = LayerOps(comment=lambda x: '; '.join(x),
               default_values= {
                "comment": "This was an example of new default values"
                },
               desc= lambda x: "This is a defaults example")  # Build LayerOps object to combine descriptions, defaults
out_layer6 = lo4.process([demo2, demo3])                      # Trigger processing on a list of demo2 and demo0
out_layer6.to_file("C:\demo_layer6.json")                     # Save combined comment layer to file
```



```python
default_values= {
 "comment": "This was an example of new default values"
 },
```

Future Layer Scripts

- **Layer exporters:**
  - Renderer (layer to SVG)
  - Excel Exporter (represent matrix layout in excel)
  - CSV Data
- **Layer Generators:**
  - Technique usage by a specific group/software
  - Summary of groups using each technique
- **Open an issue if you have any ideas!**
  - https://github.com/mitre-attack/attack-scripts/issues/

ATT&CK™

MITRE

ATT&CK®

attack@mitre.org

@MITREattack

MITRE