



Capture all traffic from SOE

⚠️ **DISCLAIMER: EDUCATIONAL PURPOSE ONLY** [🔗](#)

This article is provided strictly for educational purposes. Modifying system configurations or tampering with corporate security measures without explicit permission is prohibited and may violate organizational policies, employment agreements, and applicable laws.

This content demonstrates how corporate network defenses might be circumvented, specifically to educate security professionals about potential vulnerabilities. Organizations should implement a defense-in-depth approach with multiple layers of security to protect traffic sent and received from endpoints, including protection against threats from internal users (staff/insiders).

Always obtain proper authorization before implementing any techniques described in this article.

DISCLAIMER: EDUCATIONAL PURPOSE ONLY

Modifying Global Proxy Settings via MDM for Traffic Capture

Introduction

What is Charles Proxy?

Understanding Proxy Auto-Configuration (PAC)

Current vs. Desired Traffic Flow

Current Traffic Flow:

Desired Traffic Flow:

Step-by-Step Implementation

Step 1: Create a Simple PAC File

Step 2: Set Up a local web server hosting the PAC File

Step 3: Locate the MDM-Managed Proxy Settings

Step 4: Modify the System Configuration Files

Before making changes, back up the original files:

Convert the binary plist files to XML format for editing:

Edit the files to replace the enterprise PAC URL with your local one:

Replace ProxyAutoConfigURLString value from the enterprise PAC URL to

Convert the files back to binary format:

Verify the changes (this will not perform any conversion...):

Step 5: Protect Changes from Being Overwritten

Step 6: Verify the Configuration

Troubleshooting

Reverting Changes:

Conclusion

Modifying Global Proxy Settings via MDM for Traffic Capture [🔗](#)

Introduction [🔗](#)

In corporate environments, security teams often need to inspect network traffic for troubleshooting, security analysis, or development purposes. While many organizations implement enterprise-wide proxy solutions, there are scenarios where security professionals need to capture and analyze all traffic from a specific workstation, including system-level communications that might bypass standard proxy configurations.

This article explains how to modify global proxy settings managed by Mobile Device Management (MDM) solutions to route all workstation traffic through Charles Proxy, a popular HTTP debugging proxy application. By understanding this process, security professionals can better implement defense-in-depth strategies to protect corporate networks.

What is Charles Proxy? [🔗](#)

Charles is an HTTP proxy, HTTP monitor, and reverse proxy that enables developers and security professionals to view HTTP and SSL/HTTPS traffic between their machine and the Internet. It allows for inspection and modification of requests and responses, making it valuable for debugging, testing, and security analysis.

Understanding Proxy Auto-Configuration (PAC) [🔗](#)

A Proxy Auto-Configuration (PAC) file is a JavaScript function that determines whether web browser requests go directly to the destination or through a proxy server. In corporate environments, these PAC files are often centrally managed through MDM solutions and cannot be easily modified by end-users.

Current vs. Desired Traffic Flow [🔗](#)

Current Traffic Flow: [🔗](#)

In a typical corporate environment with MDM-enforced proxy settings, traffic flows as follows:

❌ GUI & System Applications: The traffic goes to the Enterprise Proxy or connects directly based on rules in the Enterprise PAC File, this is enforced by a MDM Profile (JAMF) or GPO. Those settings are typically immutable and taking precedences over any other settings. The traffic cannot be “manually” changed, or at least, it isn't meant to be changed...

✅ Command Line (CLI) Applications: Traffic can be manually configured to go through Charles → Alpaca → Enterprise Proxy or Direct based on Enterprise PAC File

✅ Firefox Browser: Traffic can be manually configured to go through Charles → Alpaca → Enterprise Proxy or Direct based on Enterprise PAC File

Desired Traffic Flow: [🔗](#)

Our goal is to modify the system to route ALL traffic through Charles for inspection:

✅ ALL Traffic (GUI, System, CLI, Browsers): Charles → Alpaca → Enterprise Proxy or Direct based on Enterprise PAC File

❗ IMPORTANT NOTE: This configuration will break applications that do not support or tolerate SSL interception, such as those using Mutual TLS (M-TLS), Expect-CT, or Certificate Pinning. If such connections fail, you'll need to exclude the problematic domains in Charles Proxy settings.

Step-by-Step Implementation [🔗](#)

Step 1: Create a Simple PAC File [🔗](#)

First, we need to create a PAC file that directs all traffic to the local Charles proxy:

```
1 function FindProxyForURL(url, host) {  
2     return "PROXY localhost:8888";  
3 }
```

This simple PAC file instructs the system to route all traffic through the Charles proxy running on localhost port 8888.

Step 2: Set Up a local web server hosting the PAC File [🔗](#)

Since the MDM expects a URL for the PAC file (not a file:// path), we need to serve our PAC file via HTTP. We'll create a Node.js server for this purpose:

```
1 const http = require('http');  
2 const fs = require('fs');  
3 const path = require('path');  
4  
5 // Define the path to the proxy.pac file  
6 const homeDir = process.env.HOME || process.env.HOMEPATH || process.env.USERPROFILE;  
7 const proxyPacPath = path.join(homeDir, '.config', 'charles', 'proxy.pac');  
8  
9 // Create the HTTP server  
10 const server = http.createServer((req, res) => {  
11     // Check if the request is for /proxy.pac  
12     if (req.url === '/proxy.pac') {  
13         // Read the file and serve its content  
14         fs.readFile(proxyPacPath, (err, data) => {  
15             if (err) {  
16                 res.writeHead(404, { 'Content-Type': 'text/plain' });  
17                 res.end('File not found');  
18             } else {  
19                 res.writeHead(200, { 'Content-Type': 'application/x-ns-proxy-autoconfig' });  
20                 res.end(data);  
21             }  
22         });  
23     } else {  
24         // Handle any other route
```

```

25     res.writeHead(404, { 'Content-Type': 'text/plain' });
26     res.end('Not found');
27   }
28 });
29
30 // Have the server listen on port 80
31 server.listen(80, () => {
32   console.log('Server is listening on port 80');
33 });

```

Save this as `node_pac_server.js` and run it in the background:

You can create a `KeepAlive` `LaunchAgent` or `Daemon` if you want this to always be available.

```

1 nohup node $HOME/.config/charles/node_pac_server.js > server.log 2>&1 &

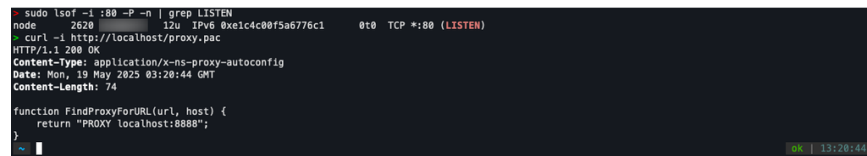
```

You can verify the server is working by checking:

```
sudo lsof -i :80 -P -n | grep LISTEN
```

or

```
curl -i http://localhost/proxy.pac
```



A terminal window showing the output of the curl command. It displays the HTTP response from the proxy server, including the status line 'HTTP/1.1 200 OK', the content type 'Content-Type: application/x-ns-proxy-autoconfig', and the date 'Date: Mon, 19 May 2025 03:20:44 GMT'. The response body contains a JavaScript function 'FindProxyForURL' that returns 'PROXY localhost:8888'.

successfull ouput

Step 3: Locate the MDM-Managed Proxy Settings [🔗](#)

To modify the global proxy settings, we first need to find where the PAC URL is defined in the MDM configuration:

```

1 sudo su -
2 cd "/Library/Managed Preferences/"
3 grep -r "pac.internal.company/company.pac" .

```

This will show you which files contain references to the enterprise PAC file.

Typically with JAMF, these will be in: `/Library/Managed Preferences/com.apple.SystemConfiguration.plist`

Step 4: Modify the System Configuration Files [🔗](#)

Before making changes, back up the original files: [🔗](#)

```

1 ditto com.apple.SystemConfiguration.plist com.apple.SystemConfiguration.plist.bak
2 ditto ./${stat -f "%Su" /dev/console}/com.apple.SystemConfiguration.plist ./${stat -f "%Su" /dev/console}/com.apple.SystemConfiguration.plist.bak

```

Convert the binary plist files to XML format for editing: [🔗](#)

```

1 plutil -convert xml1 com.apple.SystemConfiguration.plist
2 plutil -convert xml1 ./${stat -f "%Su" /dev/console}/com.apple.SystemConfiguration.plist

```

Edit the files to replace the enterprise PAC URL with your local one: [🔗](#)

```

1 vim com.apple.SystemConfiguration.plist
2 vim ./${stat -f "%Su" /dev/console}/com.apple.SystemConfiguration.plist

```

```

/Library/Managed Preferences
> vim com.apple.SystemConfiguration.plist
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
3 <plist version="1.0">
4 <dict>
5   <key>PayloadUUID</key>
6   <string>_____</string>
7   <key>Proxies</key>
8   <dict>
9     <key>ExceptionsList</key>
10    <array>
11      <string>*.local</string>
12      <string>_____</string>
13      <string>_____</string>
14      <string>169.254/16</string>
15      <string>autodiscover</string>
16      <string>_____</string>
17      <string>_____</string>
18    </array>
19    <key>FallbackAllowed</key>
20    <integer>1</integer>
21    <key>ProxyAutoConfigEnable</key>
22    <integer>1</integer>
23    <key>ProxyAutoConfigURLString</key>
24    <string>http://pac._____.pac</string>
25    <key>ProxyCaptiveLoginAllowed</key>
26    <true/>
27  </dict>
28 </dict>
29 </plist>
30

```

Replace ProxyAutoConfigURLString value from the enterprise PAC URL to <http://localhost/proxy.pac> [↗](#)

`http://localhost/proxy.pac`

Convert the files back to binary format: [↗](#)

```

1 plutil -convert binary1 com.apple.SystemConfiguration.plist
2 plutil -convert binary1 ./$(stat -f "%Su" /dev/console)/com.apple.SystemConfiguration.plist

```

Verify the changes (this will not perform any conversion...): [↗](#)

```

1 plutil -convert xml1 -o - com.apple.SystemConfiguration.plist
2 plutil -convert xml1 -o - ./$(stat -f "%Su" /dev/console)/com.apple.SystemConfiguration.plist

```

Step 5: Protect Changes from Being Overwritten [↗](#)

To prevent the MDM from overwriting our changes, we can set system immutable flags on the files:

```

1 chflags schg com.apple.SystemConfiguration.plist
2 chflags schg ./$(stat -f "%Su" /dev/console)/com.apple.SystemConfiguration.plist

```

Reboot the system for changes to take effect: `sudo reboot`

Step 6: Verify the Configuration [↗](#)

After rebooting, check that the proxy settings are correctly applied: `scutil --proxy`

This should show that the system is using your local PAC file.

You should now see tons of traffic going through Charles

(all the MacOS system web queries essentially...)

Code	Method	Host	Path	Start	Duration	Size	Status
1	HEAD	www.netflix.com	/	13.38.20	210 ms	27 bytes	200
2	GET	api.github.com	/	13.38.20	154 ms	1.38 KB	200
3	POST	github.com	/formulas/brew/gpr-upload...	13.34.27	202 ms	29.58 KB	200
4	GET	formulae.brew.sh	/api/formulae/brew.sh	13.34.29	264 ms	4.50 MB	200
5	GET	formulae.brew.sh	/api/formulae/brew.sh	13.34.30	888 ms	1.58 MB	200
6	GET	formulae.brew.sh	/api/formulae/brew.sh	13.34.30	114 ms	1.79 KB	200
7	GET	formulae.brew.sh	/api/formulae/brew.sh	13.34.30	177 ms	1.73 KB	200
8	HEAD	www.google.com	/	13.38.13	483 ms	1.08 KB	200
9	HEAD	www.google.com	/	13.38.14	428 ms	1.88 KB	200
10	HEAD	www.facebook.com	/	13.38.14	370 ms	3.62 KB	200

Charles intercepting data...

Troubleshooting [🔗](#)

SSL Interception Issues:

Some applications and services use certificate pinning or other security measures that prevent SSL interception. If you encounter connection issues with specific applications, you'll need to exclude their domains from SSL interception in Charles:

1. Open Charles Proxy
2. Go to Proxy > SSL Proxying Settings
3. Add the problematic domains to the "Exclude" list

Reverting Changes: [🔗](#)

If you need to revert the changes or make further modifications:

Remove the system immutable flags:

```
1 chflags noschg /Library/Managed\ Preferences/com.apple.SystemConfiguration.plist
2 chflags noschg /Library/Managed\ Preferences/$(stat -f "%Su" /dev/console)/com.apple.SystemConfiguration.plist
```

Either restore from backup or edit the files again to revert the changes.

Conclusion [🔗](#)

By modifying the MDM-managed proxy settings, we can route all system traffic through Charles Proxy for inspection and analysis. This technique provides valuable insights for security professionals, developers, and system administrators who need to troubleshoot or analyse network traffic in corporate environments.

Security Implications:

Understanding this technique is crucial for security teams to:

1. Recognise potential vulnerabilities in their proxy configuration
2. Implement additional security measures to prevent unauthorized modifications
3. Consider defense-in-depth approaches that don't rely solely on proxy settings
4. Monitor for unauthorized changes to system configuration files

By implementing proper security controls and monitoring, organisations can better protect their networks while still allowing legitimate traffic inspection when necessary.