Parse and Log Username from HTTP requests

Use Case:

HTTP logging is essential from most of the Web/App deployments where you want to capture the traffic details passing through. Authorization header in HTTP request carries the username which is important data to log for accountability of changes. This example shows how to parse and log the username from the Authorization header of HTTP Requests.

F5 iRules:

```
# parse_username_from_http_requests_rule

#

# Description: Capture username from web login attempts to a web application

# Look for the basic authorization header in any request and look in POST request
s

# Need to tailor the POST request searches to just specific URIs if possible

# Requires a blank stream profile on the virtual server in order to use the STREA
M:: commands

#

when RULE_INIT {


        # Log debug messages to /var/log/ltm? 1=yes, 0=only syslog event

        set ::username_debug 2


        # User name parameter name which is searched for in the POST data

        set ::user_name "UserIdentifier"


        # Regular expression which matches the user name and value in POST reques
ts

        set ::user_name_regex [subst -nocommands {${::user_name}=[^&]+}]

        if {$::username_debug}{log local0. " \$:<!--:user_name_regex: $::user_nam
e_regex"}-->


}
```

```
when HTTP_REQUEST {


        if {$::username_debug}{log local0. "[IP::client_addr]:[TCP::client_port]:
New HTTP [HTTP::method] request to [HTTP::host][HTTP::uri]"}



        # Check if there is an Authorization header value

        # This could come at any time in the application flow so check for it in
every request

        if {[HTTP::header value Authorization] ne ""}{



                # Format should be: Basic dXNlcm5hbWU6cGFzc3dvcmQ=

                #   where the token is a base64 encoding of user:pass

                if {$::username_debug}{log local0. "[IP::client_addr]:[TCP::clien
t_port]:\

                        Base64 decoded user<!--:pass: [b64decode [lindex [HTTP::h
eader Authorization] 1]],\-->

                        user: [getfield [b64decode [lindex [HTTP::header Authoriz
ation] 1]] ":" 1]"}



                # Preferred order with the Cookies last (but Secerno current expe
ects the cookies and then the username)

                log local0. "CLIENT=[IP::client_addr]:[TCP::client_port],USERNAME
=[getfield [b64decode [lindex [HTTP::header Authorization] 1]] ":" 1]"

        }



        # Check if request is a POST with a "login" parameter in the URI

        if {[HTTP::method] eq "POST" && ([URI::query [HTTP::uri] "login"] ne "")}
{



                if {$::username_debug}{log local0. "[IP::client_addr]:[TCP::clien
t_port]: Matched method and parameter check"}



                # Use a stream profile to look for the username parameter value i
n the payload

                STREAM::expression "@$::user_name_regex@ @"
```

```
                STREAM::enable

                if {$::username_debug}{log local0. "[IP::client_addr]:[TCP::clien
t_port]: Enabling stream"}

        } else {

                STREAM::disable

                if {$::username_debug}{log local0. "[IP::client_addr]:[TCP::clien
t_port]: Disabling stream"}

        }

}

when STREAM_MATCHED {



        if {$::username_debug}{log local0. "[IP::client_addr]:[TCP::client_port]:
Found match: [STREAM::match]"}



        # Prevent the username from being replaced by calling STREAM::replace wit
h no replacement string

        STREAM::replace



        # Parse the user name from the stream, with the stream expression regex m
atching username=value.

        #   Use getfield to get the value



        # Preferred order with the Cookies last (but Secerno current expeects the
cookies and then the username)

        log local0. "CLIENT=[IP::client_addr]:[TCP::client_port],USERNAME=[getfie
ld [STREAM::match] "=" 2]"



        # Disable further matches on this request as there will only be one usern
ame to find per request

        STREAM::disable

}
```

NetScaler Solution:

add ns variabe v111 -type text(1000) -init ""

add ns assignment a111 -variable $v111 -set '

HTTP.REQ.HEADER(\"Authorization\").AFTER_STR(\"Basic \").B64deCODE.BEFORE_STR(\":\")'

add rewrite policy set_variable "HTTP.REQ.HEADER(\"Authorization\").EXISTS" a111

add auditmessageaction mact1 INFO '"The user name is : "+$v111' -by yes

set syslogparams -userdefined yes

add rewrite action newact1 insert_http_header UserName '$v111' -by yes

add rewrite policy check_encode $v111.eq("").not newact1 -logaction mact1

In NetScaler, both parsing of the header and logging the username to syslog can be controlled through the advance policy framework. These simple actions make the solution very practical as you do not need to worry about