

1off.it Training Material

# Honeypot Trap

Next-Level filtering approaches with  
Flashstart DNS and MikroTik Scripting

Ver. 05/2024



## About me

# Alessandro Campanella

I have been working in the telecommunications field for over 20 years as a producer, consultant, and trainer, both in Italy and abroad.

I was among the pioneers of wireless technology, conducting experiments on my network and gathering the experiences of the many operators and integrators with whom I have collaborated over the years.

**MikroTik Evangelist**  
**Design, analysis, and consulting for ISPs**  
**MikroTik Training and Certification**



[a.campanella@1off.it](mailto:a.campanella@1off.it)



[@alessandrocampanella](https://t.me/alessandrocampanella)



# Module 1

## What is a Honeypot Trap?



# HoneyPot Trap

## Definition

- A honeypot is a **security mechanism** that creates a virtual trap to lure attackers.
- An intentionally compromised router allows attackers to **exploit vulnerabilities** so you can study them to **improve your security policies**.
- Honeypots are decoys intended to look like legitimate, vulnerable systems to **attract cybercriminals**.

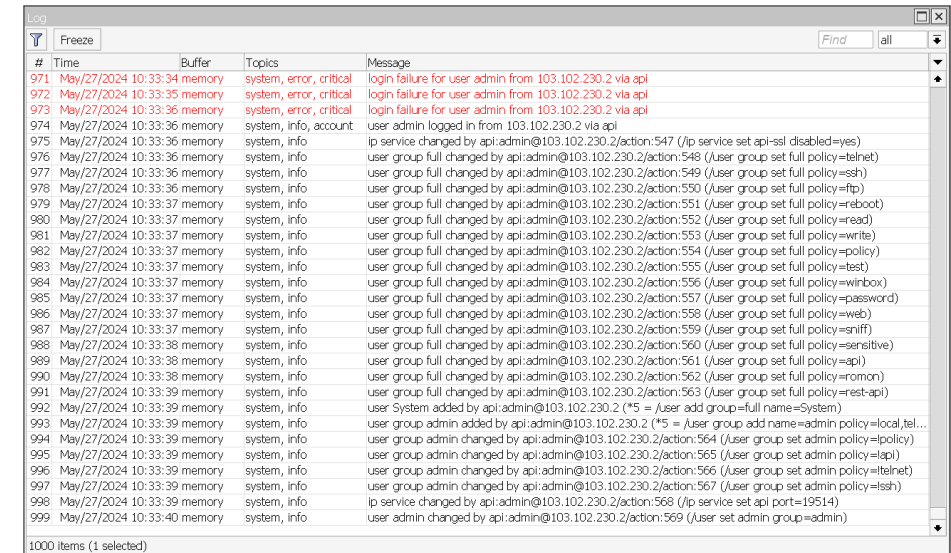
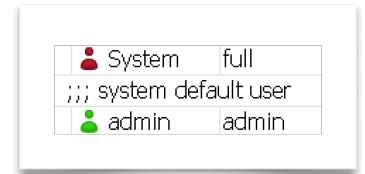
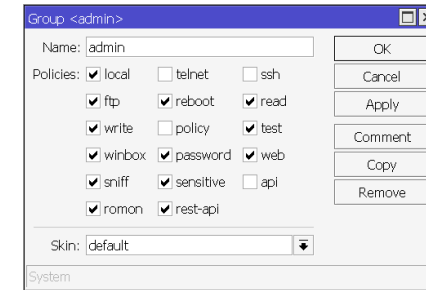




# HoneyPot Trap

## A Router with a weak password

- A public IP attracts malicious actors like honey attracts bears!
- Within minutes, the router is subjected to a **brute-force attack**.
- A new group is created that allows the admin user to manage the router but not delete the newly created user System
- All of this happens via API.





# Honeypot Trap

## One weak link to rule them all

- We can leverage these attack attempts to our advantage.
- First of all, we need to **analyze the sources** of this traffic and categorize them.
- The collected information can then be distributed to other routers to **prevent attacks**.





# Module 2

## Reverse DNS



# Reverse DNS

## What is reverse DNS?

- A reverse DNS lookup is a DNS query for the domain name associated with a given IP address. This accomplishes the opposite of the more commonly used forward DNS lookup, in which the DNS system is queried to return an IP address.
- Standards from the **Internet Engineering Task Force** (IETF) suggest that **every domain should be capable of reverse DNS lookup**.
- Reverse DNS lookups query DNS servers for a PTR (pointer) record; if the server does not have a PTR record, it cannot resolve a reverse lookup.

# Reverse DNS

## rDNS-Based Blocking

- Blocking traffic from IPs without rDNS can be a good practice for enhancing security.
- However, it should be implemented thoughtfully, considering the potential for false positives and access issues.
- Combining rDNS checks with other security measures and maintaining flexibility in your approach will help balance security and usability.





# Module 3

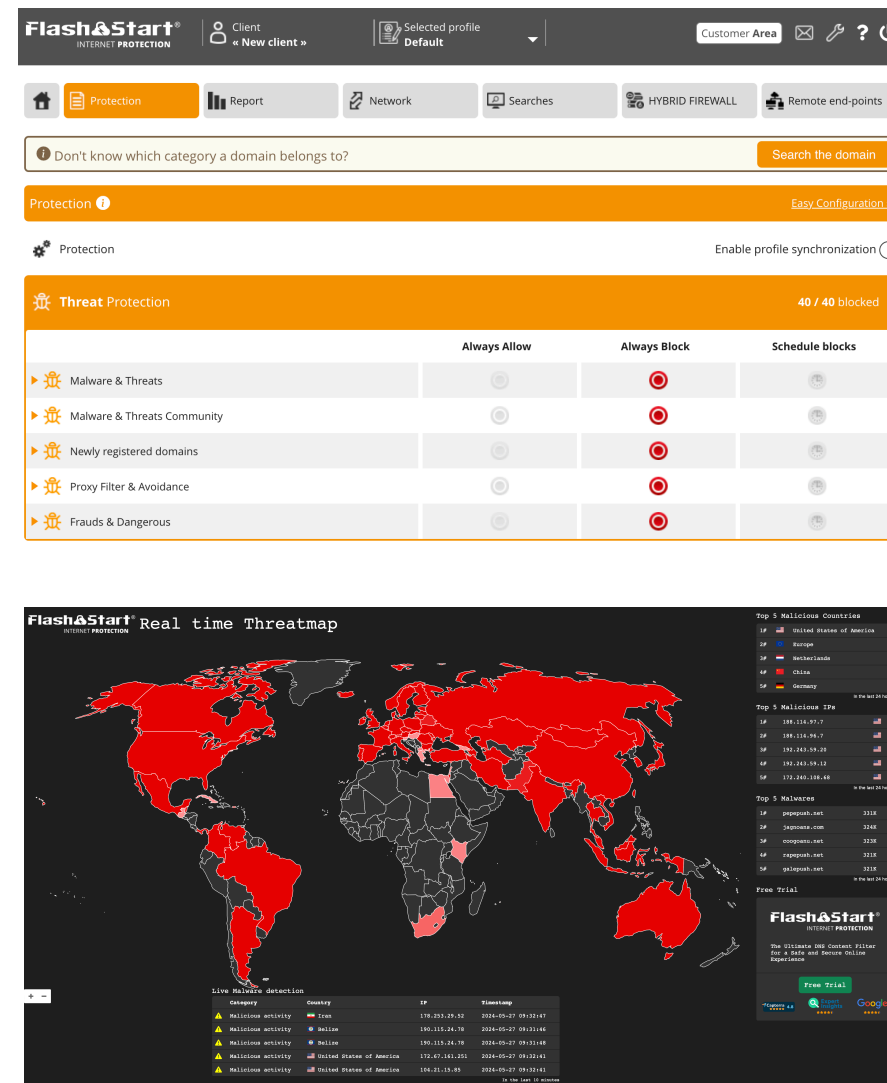
## FlashStart Threat Protection



# Threat Protection

## FlashStart DNS

- FlashStart employs real-time threat detection to identify and block malicious content as it emerges, ensuring up-to-date protection.
- Advanced AI and machine learning algorithms analyze patterns and behaviors to detect and mitigate threats effectively, even those that are new and previously unknown.



The screenshot shows the FlashStart Internet Protection dashboard. At the top, there's a navigation bar with 'FlashStart INTERNET PROTECTION', a client profile dropdown (set to 'New client'), a selected profile dropdown (set to 'Default'), and a 'Customer Area' button. Below this is a secondary navigation bar with tabs for 'Protection', 'Report', 'Network', 'Searches', 'HYBRID FIREWALL', and 'Remote end-points'. A search bar is present with the placeholder text 'Don't know which category a domain belongs to?' and a 'Search the domain' button. The main content area is titled 'Protection' and includes a toggle for 'Enable profile synchronization'. Below this is a 'Threat Protection' section showing a table of threat categories and their status.

	Always Allow	Always Block	Schedule blocks
Malware & Threats	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
Malware & Threats Community	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
Newly registered domains	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
Proxy Filter & Avoidance	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
Frauds & Dangerous	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>

Below the table, there's a 'Real time Threatmap' section showing a world map with red areas indicating threat activity. To the right of the map, there are lists for 'Top 5 Malicious Countries', 'Top 5 Malicious IPs', and 'Top 5 Malware'. At the bottom, there's a 'Live Malware detection' table and a 'Free Trial' button.

**Top 5 Malicious Countries**

IP	Country
198.124.97.7	United States of America
198.124.97.7	Russia
198.124.97.7	Germany
198.124.97.7	China
198.124.97.7	Germany

**Top 5 Malicious IPs**

IP	Country
198.124.97.7	United States of America
198.124.97.7	Russia
198.124.97.7	Germany
198.124.97.7	China
198.124.97.7	Germany

**Top 5 Malware**

IP	Country
198.124.97.7	United States of America
198.124.97.7	Russia
198.124.97.7	Germany
198.124.97.7	China
198.124.97.7	Germany

**Live Malware detection**

Category	Country	IP	Timestamp
Malware activity	Iran	198.124.97.7	2024-05-27 09:13:17
Malware activity	Brazil	198.124.97.7	2024-05-27 09:13:17
Malware activity	Brazil	198.124.97.7	2024-05-27 09:13:17
Malware activity	United States of America	198.124.97.7	2024-05-27 09:13:17
Malware activity	United States of America	198.124.97.7	2024-05-27 09:13:17

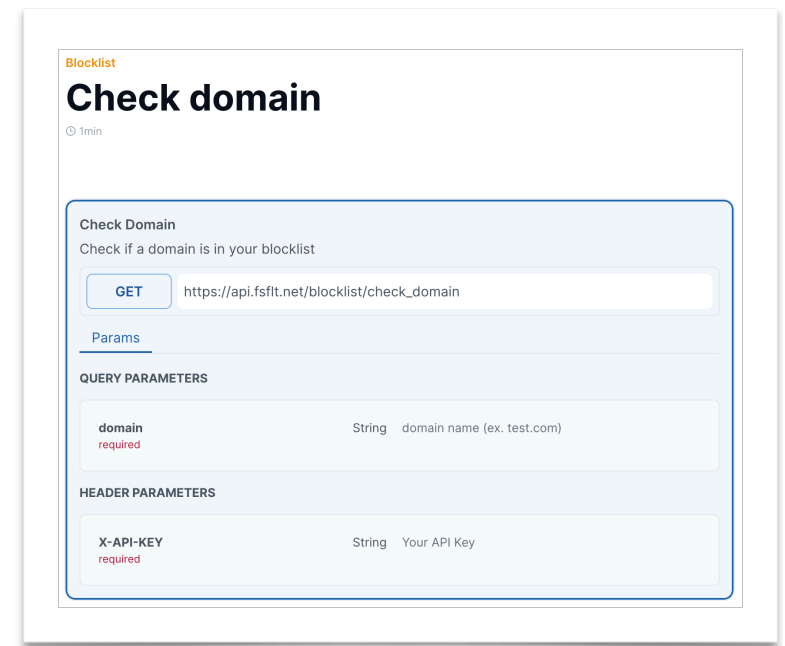
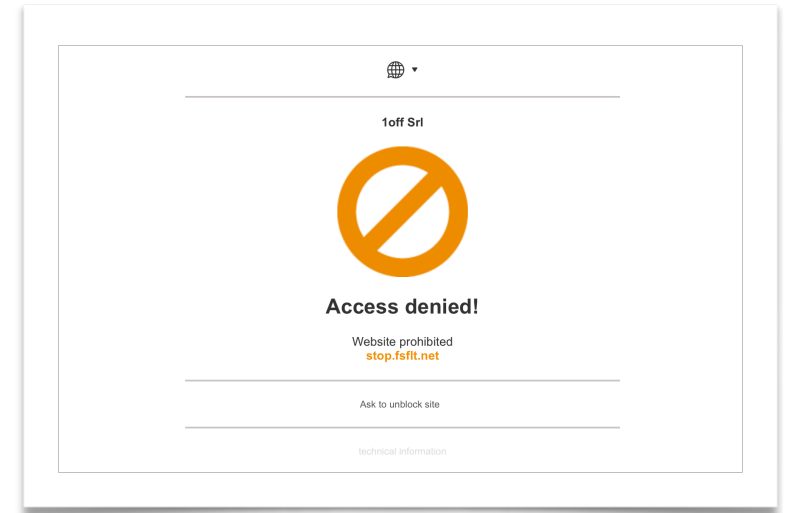
**Free Trial**

FlashStart®  
INTERNET PROTECTION  
The Ultimate DNS Content Filter  
for a FREE and Secure Online  
Experience  
[Free Trial](#)

# Threat Protection

## FlashStart Blocking page and API

- Every DNS request for a domain on the **Threat List** is redirected to the IP address of the block page **stop.fstflt.net**.
- It is also possible to query the FlashStart database via API to check if a domain is on a blocked list





# Module 4

## Preparing the Trap



# MikroTik Firewall

## Stateful mode

- First, we need to make the firewall stateful, meaning it should be capable of tracking and accepting traffic related to connections that it has initiated.
- We also need to create a rule that accepts traffic from a list of authorized addresses, which we will call the whitelist.

```
/ip firewall filter
add action=accept chain=input connection-state=established,related
add action=accept chain=input src-address-list=whitelist
/ip firewall address-list
add address=home.1off.it list=whitelist
```



# MikroTik Firewall

## Capture the input traffic

- Let's create a firewall rule that identifies all incoming traffic directed to the router's IP from the WAN, which does not belong to the list of addresses testedIP that contains already checked IPs.
- We will add these IPs to the toBeTested list.

```
/ip firewall filter  
add action=add-src-to-address-list address-list=toBeTested address-list-timeout=none-static chain=input \  
    dst-address-type=local in-interface=ether1 src-address-list=!testedIPs
```

# MikroTik Firewall

## Configuring the Raw table to drop traffic

- We will now create firewall/raw rules that will only accept traffic from addresses in the whitelist and discard traffic coming from three different categories of senders: **blacklist**, **quarantine**, and **suspicious**.
- We create these rules in the raw table to minimize CPU load.

```
/ip firewall raw  
add action=accept chain=prerouting src-address-list=whitelist  
add action=drop chain=prerouting src-address-list=blacklist  
add action=drop chain=prerouting src-address-list=quarantine  
add action=drop chain=prerouting src-address-list=suspicious
```



# Module 5

## Functions and Scripts



# MikroTik Functions

## Create a function and pass arguments

- To create a function, we declare a global variable and assign a code block to that variable, we can use the **:return** command to return a value.
- When creating a function, the commands to be executed within the function are contained in a **do={}** block
- When calling a function, it is possible to pass arguments. This are the data that the function should operate on when executing its code block.

```
:global myFunc do={:put "This is the value of argument a: $a"}  
$myFunc a="this is argument a value"
```

Output:  
This is the value of argument a: this is argument a value

# MikroTik Functions

## testDomain

Use FlashStart API to check if a domain is blocked

- **Arguments**  
domain
- **Returns**  
true/false
- **Syntax**  
`$testDomain domain="google.com"`

```
:global testDomain do={  
  
    :local url "https://api.fsflt.net/blocklist/check_domain"  
    :log warning ("https://api.fsflt.net/blocklist/check_domain?domain=".$domain)  
  
    # Call FlashStart API  
    :local result [/tool fetch url=($url."?domain=".$domain) http-method=get  
mode=https http-header-field=""accept: application/json',"Content-  
Type:application/json\',"X-API-  
KEY:6f244333-2863-4a13-9b0a-454edde95d9c\''" output=user as-value]  
  
    # Deserialize the downloaded json and check if the domain is blocked  
    :if ($result->"status" = "finished") do={  
        :local deserResult [:deserialize ($result->"data") from=json]  
        :local isBlocked ($deserResult->"content"->"blocked")  
        :return $isBlocked  
    } else { :log error "Error connecting to FlashStart API"  
    }  
}
```



# MikroTik Functions

## getDomain

Find the top-level domain

- **Arguments**  
domain
- **Returns**  
cropped domain  
("www.1off.it" -> "1off.it")
- **Syntax**  
\$getDomain domain="www.1off.it"

```
:global getDomain do={  
  
:local dotPos [:len $domain];  
:local lastDotPos 0;  
:local secondLastDotPos 0;  
:local cropDomain "";  
  
# Find last dot position  
:while (($dotPos > 0) && ([:pick $domain ($dotPos - 1) $dotPos] != ".")) do={  
    :set dotPos ($dotPos - 1);  
}  
  
:set lastDotPos $dotPos;  
  
# Find second dot position  
:set dotPos ($lastDotPos - 1);  
:while (($dotPos > 0) && ([:pick $domain ($dotPos - 1) $dotPos] != ".")) do={  
    :set dotPos ($dotPos - 1);  
}  
  
:set secondLastDotPos $dotPos;  
  
# Extract the domain if the length of the segment between the penultimate and  
the last point is appropriate  
:if (($lastDotPos - $secondLastDotPos - 1) > 3) do={  
    :set cropDomain [:pick $domain ($secondLastDotPos) $lastDotPos];  
} else={  
    # Find the next point to the left if the block is too short  
    :set dotPos ($secondLastDotPos - 1);  
    :while (($dotPos > 0) && ([:pick $domain ($dotPos - 1) $dotPos] != ".")) do={  
        :set dotPos ($dotPos - 1);  
    }  
    :set cropDomain [:pick $domain ($dotPos) $lastDotPos];  
}  
  
:return ("${cropDomain}[:pick $domain $lastDotPos [:len $domain]]");  
}
```

# testInputIP script

## Catalog sender with rDNS

- Resolve the reverse DNS of the IP with Google DNS Server
- Resolve the DNS of the reverse DNS with FlashStart DNS Server
- If the IP address is blocked by FlashStart, add to **blacklist**
- If the reverse DNS is consistent, add to **suspicious** list
- If reverse DNS cannot be found, add to **quarantine** list

# testInputIP script (part 1)

```
# Include functions
:global getDomain
:global testDomain

# For each address in the toBeTested address-list do:
:foreach i in=[/ip/firewall/address-list/find where list=toBeTested] do={
    :do {
        # Resolve the reverse DNS of the IP with Google DNS Server and remove the final "." character
        :local tempDNS [:resolve [/ip/firewall/address-list get $i address] server=8.8.8.8]
        :if ([[:pick $tempDNS ([:len $tempDNS]-1) ([:len $tempDNS])]] = ".") do={
            :set $tempDNS [:pick $tempDNS 0 ([:len $tempDNS]-1)]
        }
        # Resolve the DNS of the reverse DNS with FlashStart DNS Server
        :local tempReverse [:resolve [$tempDNS] server=185.236.104.104]
        :log warning ("Input traffic received from: ".$tempDNS)
        # If the IP address resolved with FlashStart DNS is not equal to the original IP address, check if the domain is blocked with FlashStart API and add to blacklist
        :if ([/ip/firewall/address-list get $i address] != $tempReverse) do={
            :local redDom [$getDomain domain=($tempDNS)]
            :local ar1 [$testDomain domain=($redDom)]
            :local ar2 [$testDomain domain=($tempDNS)]
            :if ($ar1 || $ar2) do={
                :log error ("Domain blocked by FlashStart API: ".$tempDNS.". BLACKLISTED!!!")
                /ip/firewall/address-list/add list=blacklist address=[/ip/firewall/address-list get $i address] comment=("API: ".$tempDNS)
            } else={
                # Add to blacklist with Reverse Resolution comment
                :log error ("Resolution for ".$tempReverse." is not consistent: ".$tempReverse.". BLACKLISTED!!!")
                /ip/firewall/address-list/add list=blacklist address=[/ip/firewall/address-list get $i address] comment=("Inconsistent domain ".$tempDNS)
            }
        }
        # Add IP in testedIP list
        /ip/firewall/address-list/add list=testedIPs address=[/ip/firewall/address-list get $i address] comment="blacklist"
        # Remove IP from toBeTested list
        /ip/firewall/address-list/remove [/ip/firewall/address-list find address=[/ip/firewall/address-list get $i address] list=toBeTested]
```

## testInputIP script (part 2)

```
    } else={
        :log warning ("Resolution consistent. Reverse DNS for: "[/ip/firewall/address-list get $i address]." is: ".$tempReverse.". Added to SUSPICIOUS")
        # Add IP in testedIP list for 10 minutes
        /ip/firewall/address-list/add list=testedIPs address=[/ip/firewall/address-list get $i address] comment="suspicious" timeout=10m
        # If the Reverse DNS is consistent, add to suspicious list for 10 minutes
        /ip/firewall/address-list/add list=suspicious address=[/ip/firewall/address-list get $i address] comment=("Suspicious domain ".$tempDNS) timeout=10m
        # Remove IP from toBeTested list
        /ip/firewall/address-list/remove [/ip/firewall/address-list find address=[/ip/firewall/address-list get $i address] list=toBeTested]
    }
    } on-error={:log warning ("No reverse DNS found for IP: "[/ip/firewall/address-list get $i address])
    # Add IP in testedIP list for 10 minutes
    /ip/firewall/address-list/add list=testedIPs address=[/ip/firewall/address-list get $i address] comment="quarantine" timeout=10m
    # If Reverse DNS cannot be found, add to quarantine list for 10 minutes
    /ip/firewall/address-list/add list=quarantine address=[/ip/firewall/address-list get $i address] comment=("No reverse DNS found") timeout=10m
    # Remove IP from toBeTested list
    /ip/firewall/address-list/remove [/ip/firewall/address-list find address=[/ip/firewall/address-list get $i address] list=toBeTested]
    }
}
```

# Schedule

## Initialize the router and run the script periodically

- It is essential to initialize the functions we have created. To do this, we need to run the scripts that define them at startup.
- We then run the control script every 30 seconds for example.

```
/system scheduler  
add interval=30s name=testInputIp on-event=testInputIp policy=\  
    ftp,reboot,read,write,policy,test,password,sniff,sensitive,romon start-time=startup  
add name=Include on-event="testDomain;testInputIp" policy=\  
    ftp,reboot,read,write,policy,test,password,sniff,sensitive,romon start-time=startup
```



# LAB 1

## Populating Address Lists

Let's see how the routers configured in this way perform in a real lab setting.



# Module 6

## Share informations



# MikroTik API

## Using REST API

- The term REST API generally refers to an API accessed via HTTP protocol at a predefined set of resource-oriented URLs.
- Starting from **RouterOS v7**, it is implemented as a **JSON wrapper** interface of the console API. It allows to create, read, update and delete resources and call arbitrary console commands.
- To start using REST API, the **www-ssl** service (HTTPS access) must be configured and running.
- We also need a valid certificate that can be obtained via **Let's Encrypt**.

# MikroTik API

## Create a dedicated User Group and User

- It is recommended to create a dedicated user group and a dedicated user for the APIs.

```
/user group  
add name=api policy="rest-api,!local,!telnet,!ssh,!ftp,!reboot,read,!write,!policy,!test,!winbox,!password\  
    ,web,!sniff,!sensitive,api,!romon"  
/user/add name=api group=api password=[strong password]
```



# Importing Address Lists using API

## Ready to share the Address Lists

- It is now possible to retrieve the address lists created on the Honeypot router via API.
- To do this, we will create a procedure for acquiring a list and then call this procedure for the three lists.

# MikroTik Functions

## getAddressList

Fetch Address List from honeypot and insert in Address List

- **Arguments**  
addList
- **Syntax**  
\$getAddressList addList="blacklist"

```
:global getAddressList do={  
  
# Call honeypot API  
:local url ("https://honeypot.1off.it/rest/ip/firewall/address-list?list=".$addList)  
:log warning ("Update list: ".$addList)  
:local result [/tool/fetch user=api password=a4joEQs14lGEsaf mode=https  
url=$url as-value output=user]  
  
# Convert result to array  
:local resultData [toarray ($result->"data")]  
:local counter 0  
  
:foreach record in=[ $resultData ] do={  
:if ($counter != 0) do={  
# Convert each record (except first empty) to array  
:local tempRec [toarray $record]  
# Check if the IP is still in Address-List  
:local ifExist [/ip/firewall/address-list find where address=[:pick $tempRec 5]  
list=[:pick $tempRec 20]]  
:if ([:len $ifExist] = 0) do={  
# Check if the entry has timeout  
:local timeOut [:pick $tempRec 23]  
:if ([:len $timeOut] != 0) do={  
# Add IP to Address-List with timeout  
/ip/firewall/address-list/add list=[:pick $tempRec 20] address=[:pick  
$tempRec 5] comment=[:pick $tempRec 8] timeout=[:pick $tempRec 23]  
} else={  
# Add IP to Address-List without timeout  
/ip/firewall/address-list/add list=[:pick $tempRec 20] address=[:pick  
$tempRec 5] comment=[:pick $tempRec 8]  
}  
}  
}  
:set $counter ($counter +1)  
}  
}
```

# getAddressList

## Obtain blacklist, quarantine and suspicious address lists

- Let's create the script for this and schedule the update

```
:global getAddressList
$getAddressList addList="blacklist"
:delay 2
$getAddressList addList="suspicious"
:delay 2
$getAddressList addList="quarantine"

/system scheduler
add interval=30s name=updateLists on-event=updateLists policy=ftp,reboot,read,write,policy,test \
password,sniff,sensitive,romon start-time=startup
add name=include on-event=getAddressList policy=ftp,reboot,read,write,policy,test \
password,sniff,sensitive,romon start-time=startup
```

# Using the Address Lists

## Create the firewall rules

- Once the address lists are imported into the router, firewall rules can be created.
- For example, we could discard all input and forward traffic for the blacklist, discard input traffic for the suspicious and quarantine lists, and evaluate whether to filter forwarded traffic for the latter.





# LAB 2

## Sharing informations

Let's see how the routers configured in this way perform in a real lab setting.



# Module 7

## Conclusions



# Disclaimer

## Don't be misled by all this honey

- This guide is designed to demonstrate the **effectiveness of FlashStart's DNS** filters and to showcase the incredible management capabilities of MikroTik routers through advanced scripting.
- It should not be taken as a best practice or as a configuration guide for a production firewall.



# Conclusion

## Positive results

- **FlashStart** is extremely effective at identifying malicious senders.
- The use of DNS is certainly a useful and fast method for analyzing traffic sources.
- MikroTik offers such advanced scripting tools that they enable the creation of extremely effective analysis and control systems.

1off.it Training Material

# Thank you

Alessandro Campanella

