Combatting Incident Responders with Apache mod_rewrite

By Jeff Dimmock

Published: 2016-04-12 · Archived: 2026-04-05 13:56:53 UTC

Any phishing campaign involving an active incident response element usually requires some evasive steps to prolong its longevity. This often includes being stealthier, performing anti-forensics actions, or avoiding certain tradecraft altogether. Phishing is no different, and is often the most ‘vulnerable’ part of a campaign from an active IR perspective. Using a distributed infrastructure built with independent components helps reduce the risk of the overall architecture being blocked, but individual phishing campaigns are likely to be caught and blocked throughout the duration. The longer we can stretch out the usability of each of those campaigns, the better our chances of gaining access.

Using Apache mod_rewrite rules, we can rewrite potential incident responder or security appliance requests to an innocuous website or the target’s real website. While the methods discussed below won’t stave off a concerted investigation, it will hopefully make the malicious website pass the ‘sniff test’ with recipients and lower level help desk or incident responders.

It’s important to note that these techniques could prevent valid phishing victims from reaching your malicious website. You will need to weigh the risks of losing out on potential clicks against the risks posed by the target’s incident responders.

Block Common IR User Agents

Blocking common incident responder or security appliance user agents is an easy way to reject a swath of traffic hitting your malicious website. I watch my web server’s Apache access logs like a hawk while a phishing campaign is active. It usually doesn’t take long before web crawlers and security appliances start making GET requests. That traffic is normal for web servers, but we don’t really want our malicious website being crawled, classified, and archived when we’re trying to be stealthy.

Add the following ruleset to the top of any ruleset already in place, below the `RewriteEngine On` line, to redirect any blank or provided user agents to an alternate location. If you identify the security products your target uses and think it is performing heuristic analysis on the page for filtering purposes, you can additionally add a separate copy of these rules to match the product(s) and proxy the request to the alternate site, rather than redirect.

```
RewriteCond %{HTTP_USER_AGENT} "wget|curl|HTTrack|crawl|google|bot|b\-o\-t|spider|baidu" [NC,OR]
RewriteCond %{HTTP_USER_AGENT} =""
RewriteRule ^.*$ http://COMPANYDOMAIN.com/404.html/? [L,R=302]
```

Line by line explanation:

If the request's user agent matches any of the provided keywords, ignoring case. OR
If the request's user agent is blank

Change the entire request to http://COMPANYDOMAIN.com/404.html/ and drop any query_strings from original request. Do not evaluate further rules and redirect the user, changing their address bar.

As mentioned in my first post about mod_rewrite, placing this ruleset in a *.htaccess* file allows changes to be made on the fly without needing to restart the Apache service. We can make adjustments as the campaign progresses if we notice more user agents that should be filtered. For a lengthy list of common user agents, check out useragentstring.com.

IP Filtering

IP filtering provides the ability to selectively proxy requests or redirect the user based upon the originating IP address. We can monitor the Apache logs and modify filtering rules throughout the phishing campaign's execution.

Whitelisting

IP whitelisting is an option if you can determine the IP addresses or ranges from which the targets will originate. For straight whitelisting, use the following ruleset:

```
RewriteEngine On
RewriteCond %{REMOTE_ADDR} ^100\.0\.0\. [OR]
RewriteCond %{REMOTE_ADDR} ^100\.0\.1\.
RewriteRule ^.*$ http://TEAMSERVER-IP%{REQUEST_URI} [P]
RewriteRule ^.*$ http://COMPANYDOMAIN.com/404.html/? [L,R=302]
```

Line by line explanation:

Enable the rewrite engine
If the requestor's IP address starts with 100.0.0, ; OR
If the requestor's IP address starts with 100.0.1,
Change the entire request to serve the original request path from the teamserver's IP, and keep the user's address bar the same (obscure the teamserver's IP).
If the above conditions are not met, change the entire request to http://COMPANYDOMAIN.com/404.html and drop any query strings from the original request. Do not evaluate further rules and redirect the user, changing their address bar.

In this example, any user visiting from 100.0.0.0/24 and 100.0.1.0/24 will have the original URI request proxied to the teamserver. Visitors from other IPs will be redirected to *companydomain.com/404.html*.

Doing a simple whitelist will also block any users trying to load the page from a non-corporate IP, such as coffee shop, home, or mobile device. To slightly alleviate this risk, we can add a *RewriteCond* line to match requests from mobile user agents and allow those requests to access our site:

```
RewriteEngine On
RewriteCond %{HTTP_USER_AGENT} "android|blackberry|googlebot-mobile|iemobile|ipad|iphone|ipod|opera mobile|palmos|webos" [NC,OR]
```

```
RewriteCond %{REMOTE_ADDR} ^100\.0\.0\. [OR]
RewriteCond %{REMOTE_ADDR} ^100\.0\.1\.
RewriteRule ^.*$ http://TEAMSERVER-IP%{REQUEST_URI} [P]
RewriteRule ^.*$ http://REDIRECTION-URL.com/? [L,R=302]
```

Blacklisting

Blacklisting is inherently a reactive control, but can be useful on later phish campaigns after IR and control product's IPs have been identified. The ruleset is just flipping the last two lines so that the RewriteCond matches are redirected instead of proxied to the teamserver:

```
RewriteEngine On
RewriteCond %{REMOTE_ADDR} ^100\.0\.0\. [OR]
RewriteCond %{REMOTE_ADDR} ^100\.0\.1\.
RewriteRule ^.*$ http://REDIRECTION-URL.com/? [L,R=302]
RewriteRule ^.*$ http://TEAMSERVER-IP%{REQUEST_URI} [P]
```

Time-Based Filtering

To reduce the risk of off-hours IR loading our malicious site, we can use the time-based server variables built into mod_write. For example, if we wanted to limit users to only load our site during business hours:

```
RewriteEngine On
RewriteCond %{TIME_HOUR}%{TIME_MIN} >0600
RewriteCond %{TIME_HOUR}%{TIME_MIN} <2000
RewriteRule ^.*$ http://TEAMSERVER-IP%{REQUEST_URI} [P]
RewriteRule ^.*$ http://REDIRECTION-URL.com/? [L,R=302]
```

Line by line explanation:

Enable the rewrite engine
If the request time (hour and minute) is greater than 0600 server local time, AND
If the request time (hour and minute) is less than 2000 server local time,
Change the entire request to serve the original request path from the teamserver's IP, and keep the user's address bar the same (obscure the teamserver's IP).
If the above conditions are not met, change the entire request to http://REDIRECTION-URL.com/ and drop any query strings from the original request. Do not evaluate further rules and redirect the user, changing their address bar.

Any requests received between the hours of 6am and 8pm are proxied to our malicious site. Any requests received outside that time range are redirected. Time calculation is based upon the server's locale settings. Use the `date` command to verify the time zone before configuring this ruleset.

We can also filter based on the day of the week with the *%{TIME_WDAY}* server variable. The regex uses 0 - 6 to represent Sunday through Saturday. To match days, use <, >, or = in the regex portion of the *RewriteCond* line.

For example, the following *RewriteCond* rule matches Monday through Friday:

```
RewriteCond %{TIME_WDAY} >0
RewriteCond %{TIME_WDAY} <6
```

Summary

Apache mod_rewrite provides a wealth of powerful features we can use to make a phishing campaign more resilient to detection. We can filter requests based upon a number of properties, including user agent, source IP, and time. By redirecting or proxying those requests to an innocuous website we increase the chances that the investigator or security appliance will disregard the phish and continue allowing access.

Strengthen Your Phishing with Apache mod_rewrite Posts

- Strengthen Your Phishing with Apache mod_rewrite and Mobile User Redirection
- Invalid URI Redirection
- Operating System Based Redirection with Apache mod_rewrite
- Combatting Incident Responders with Apache mod_rewrite
- Expire Phishing Links with Apache RewriteMap

Resources

- mod-rewrite-cheatsheet.com
- Official Apache 2.4 mod_rewrite Documentation
    - Apache mod_rewrite Introduction
- An In-Depth Guide to mod_rewrite for Apache
