An NCC Group Publication

Drive-by Enumeration of Web Filtering Solutions

Prepared by:
Ben Williams

Contents

1 List of Figures and Tables

2 Introduction

This paper follows on from the previous work *Automated Enumeration of Email Filtering Solutions*, and summarises research undertaken in 2014 to develop offensive reconnaissance techniques for automated and external enumeration of web filtering solutions. We show how a methodology, simple web application, and test download files can be used to enumerate web filtering solutions quickly and to a high level of detail and accuracy. Enumeration described here is performed without requiring any exploits but using product and service features which are there by design.

Details which can be enumerated by an external attacker include:

- The web filtering managed service, software, or appliance products in use, with version information, hostnames, internal IP addresses, and proxy ports.
- A detailed picture of the filtering policy in place, including identification of policy or configuration loopholes.
- The capability of the products and services in use, and their ability to handle identification of hidden threats in more challenging formats (such as embedded within various documents, archives or other specially chosen attachment format-types, or specially encoded formats).
- Whether any inspection or blocking is being done for HTTPS (via HTTPS MitM).
- Detection of desktop antivirus products via their browser plugin, with measurement of capability and sometimes version information.

These techniques require a minimal level of interaction with internal users (getting a user to click on a link for example, though the enumeration may not be noticeable to users in most situations). Information on filtering weaknesses identified using these techniques could be gathered by attackers in a reconnaissance phase, to help tailor targeted phishing, malware, or client-side attacks, by providing information on effective ways to bypass the specific filtering in place.

This paper builds on previous work by the author in identifying vulnerabilities in a variety of security appliances[1] and gateways[2], and shows how the discovery and further enumeration of vulnerable web security solutions can be achieved. This may be of special interest as internal web security proxies have previously been difficult to enumerate as they do not normally have any services externally exposed.

In most situations, this type of enumeration can be fast and accurate, with a relatively simple web application. We also take some time to explore how test file download sets can be constructed in an automated way, to help measure the limit of product capabilities, and identify which file types and encodings effectively evade most filters.

The techniques described in this paper were developed by testing the techniques in a variety of situations including penetration testing engagements, product capability assessments, and sampling tests on site visitors.

[1] Hacking Appliances: Ironic exploits in security products
https://media.blackhat.com/eu-13/briefings/B_Williams/bh-eu-13-hacking-appliances-bwilliams-wp.pdf

[2] They ought to know better: Exploiting Security Gateways via their Web Interfaces
https://www.nccgroup.com/media/18475/exploiting_security_gateways_via_their_web_interfaces.pdf

3 An Overview of Common Web Filtering Solutions

Here we give a brief overview of typical technologies, topologies, options, and best practices for web filtering.

3.1 Common Web Filtering Solutions and Topology

There are a variety of products and services available, with offerings from many vendors. A brief overview of typical web filtering systems and characteristics is provided in this section, to aid understanding of subsequent sections. Examples of the following topologies were seen during NCC Group's research and tests on a variety of organisations.

3.1.1 Onsite Web Filtering Appliances, UTM Gateways, and Software Filtering Solutions

Onsite web security proxies were among the most common solutions observed. Products seen included appliances, multi-function gateways, firewalls and software solutions. In this configuration, internal browsers are configured to proxy via the web filtering solution, and direct unfiltered Internet access is blocked with access control restrictions at boundary firewalls.

As well as offering good protection for URL filtering, known malware, and other code execution threats, some appliances can be configured with complex customised policies and can detect embedded threats. By using onsite solutions, the organisation retains confidentiality, as web browsing traffic is not routed and processed via a third party.

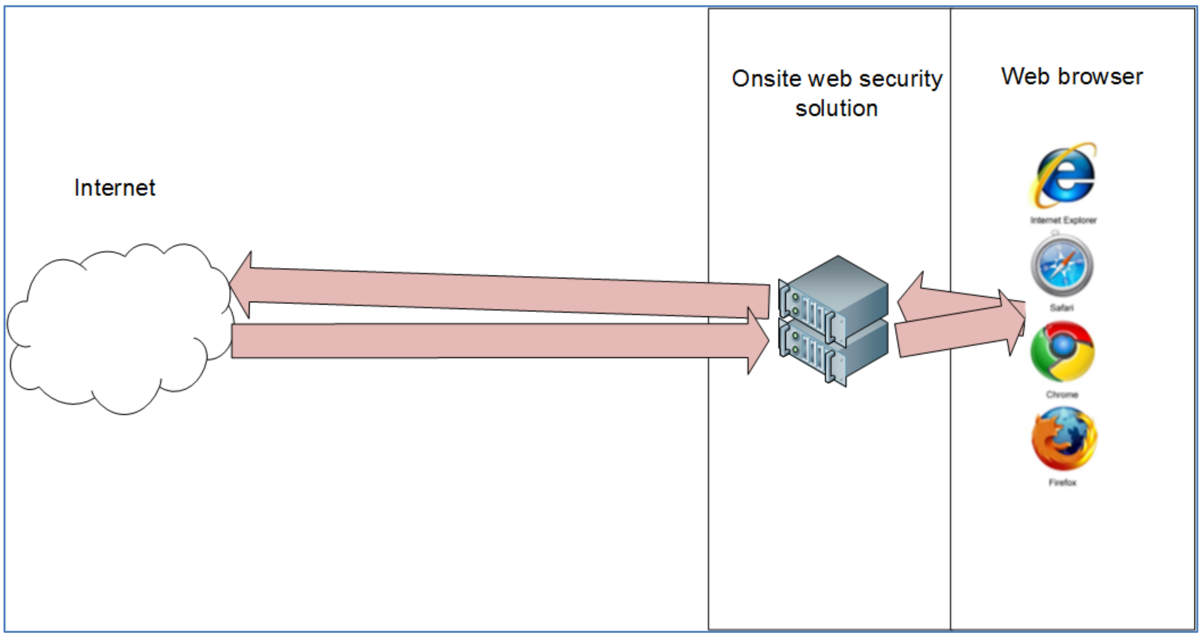


Figure 1 Onsite web security appliance

For internal web proxies as described above, clients and firewalls should be configured such that it is not possible for web browsers to directly access the Internet without traffic being routed via the secure web proxies. This ensures that all HTTP browser traffic is processed by the web security proxy.

Transparent proxies could be used where client configuration is challenging, though typically this is not preferable, as full proxies configured via the browser provide more functionality (such as the ability to modify requests and responses). In enterprises, browsers are typically configured using Proxy Auto Configuration (PAC)[3] scripts, which makes deployment more manageable.

3.1.2 External Web Security Managed Services

A less common alternative to using an onsite web filtering solution is to use an external web managed service. In this case, browsers are configured to proxy via the external service (much like onsite appliances). Web traffic is processed and filtered externally and forwarded to and from the target organisation's caching proxy (or directly from users' browsers).

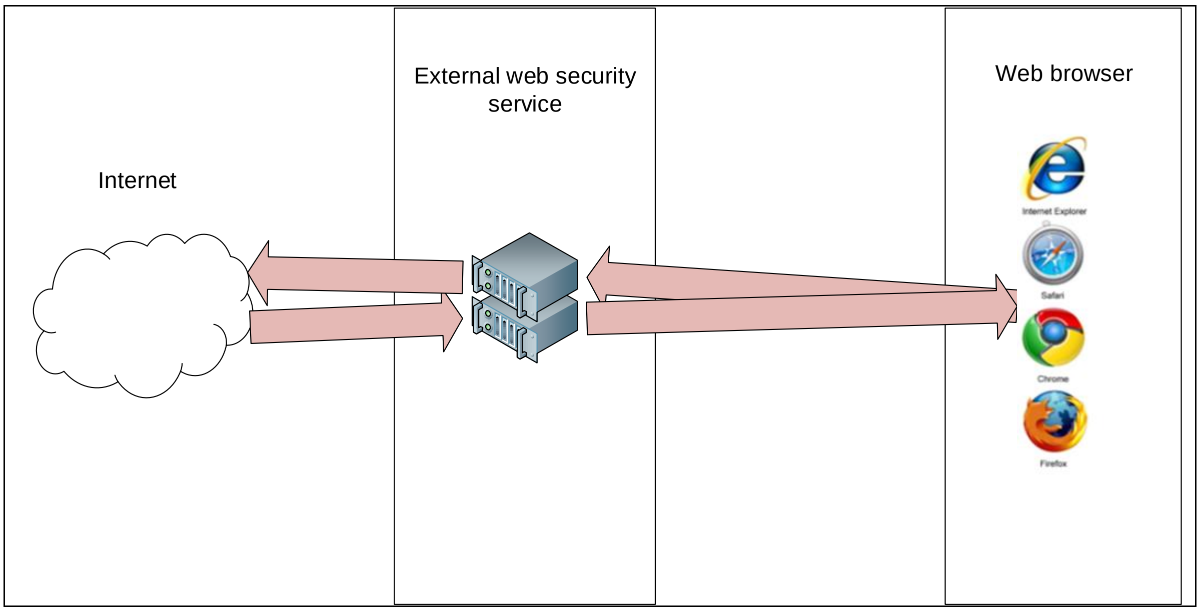


Figure 2 External web security managed service

Managed web filtering services offer the benefit of ease of deployment and outsourcing of the service. These services process very large amounts of URL requests for many sites, and have a high visibility of site data, collating statistical data across many systems and tracking site classifications based on the data seen.

Web security managed services can give a good level of protection in terms of classifying popular site content and blocking known malware. However, these services are typically limited in terms of their ability to perform deep content analysis (finding embedded executable code in document and archive attachments for example) and define complex granular policy rules for groups or individual users. Similar to email managed services, confidentiality issues are a concern, though the ability for

[3] Proxy Auto Configuration
http://en.wikipedia.org/wiki/Proxy_auto-config

web security managed services to perform inspection of HTTPS content was not observed during this research.

3.1.3 Additional Defence Layers

Multiple layers of web proxies are sometimes seen, though these extra proxy layers are almost always deployed to provide improved performance through localised and shared caching of static content, rather than to implement multiple security boundaries or layered defences.

Desktop antivirus often provides a browser plugin feature which can provide additional protection within the browser for some known malware and generic exploit vectors. URL filtering can also be implemented by browser plugins, by restricting DNS resolution to undesirable domains, or by restricting site access by IP address.

Some additional web security features are provided by browsers themselves (for example Microsoft Internet Explorer's "SmartScreen Filter", Google Chrome's "Safe Browsing" and Mozilla Firefox's "Phishing and Malware Protection"). These browser features implement a malware detection and warning system for downloadable executable code and scripts, to reduce the risk of users running them directly by clicking on a link. These features can help supplement the identification of known malware or unusual executable code, though these features are typically easily bypassed and should be supplemented with web security proxies, and desktop antivirus products.

One advantage of browser plugins is that, for HTTPS traffic, they offer the benefit of checking content before requests are encrypted and after responses are decrypted, effectively providing some layer of defence for HTTPS traffic. However, though browser features and plugins can offer an additional defensive layer, this can typically be easily bypassed (for example by embedding known threats in other format types, such as documents or archives).

3.2 Presenting a Consistent Defence

Various features are typically required to protect organisations by presenting a consistent defence.

3.2.1 The Problems of Remote Workers Accessing the Internet

For filtering to be effective, it should not be possible to trivially bypass web filtering by directly accessing the Internet from users' laptops or desktops. Ideally, these systems should always access internet resources via an organisation's secure web proxy.

This is challenging in some situations, especially with remote workers needing to access the Internet. For secure environments, remote workers should always connect to an organisation's VPN and access the Internet via the organisational secure web proxies. If this does not happen then their system could potentially be compromised. A system compromised in this way can then introduce threats into the wider organisational network when the remote workers later return to the office or connect via a VPN.

Another option for remote workforces is to provide split tunnelling, which enables remote workers to access the Internet directly while connected to an organisational VPN. This means that there is relatively unfiltered access to the Internet at the same time that laptops are connected to the corporate network via the VPN, which presents a much increased risk. NCC Group regularly assess laptop and VPN configurations for remote worker scenarios, and providing access, speed, and functionality for remote workers, as well as effective internet filtering, is a recurring problem.

3.3 Filtering Policy Best Practices

For organisations, filtering the web for unwanted URL categories and removing known malware are primary goals for secure web proxies. Additional threats such as unknown executable code, applets, thick client apps, scripts, and documents containing macros should be addressed to protect end users. It is also important to manage encrypted attachments and HTTPS resources appropriately, as these could contain any of the threats listed above. Threats should be detected, and action taken, which usually takes the form of editing content to remove the offending items or blocking the requests or responses (replacing them with block-pages for example).

3.3.1 Deep Content Inspection

As with email filtering solutions, web filtering solutions should identify executable code and scripts by MIME-type, file extension and file signature. They should also unpack encoded attachments, including popular types of documents, archives, and compound files, and identify threats contained within them, and this unpacking should be performed in a multi-layered approach. Web security proxies can typically perform much more in-depth analysis than browser features.

There are three main ways of blocking unwanted file attachments:

- By MIME type (partially effective)
- By file extension (partially effective)
- By file content
    - Usually signature based, and is more accurate than the above methods
    - Not applicable to some file types which have no defined structure (such as vbs or bat scripts)

Where possible, all three of the above should be implemented. Although blocking by file extension is not reliable (as files can be renamed) it is important to implement a policy for file extensions and MIME-types as scripts effectively have no file signature, so the extension is used to determine the execution context. High-risk file extensions can execute arbitrary code when run, so a list of high-risk file extensions can be blocked (in addition to blocking by content-type).

However, it should also be noted that with HTML5 a bypass which renames the file extension at the client is possible, so deep content inspection is important:

```
<a href="innocent.txt" download="nasty.bat">Download this file</a>
```

For secure environments, it may also be desirable to block Java applets, Flash, Silverlight or other client-side code.

3.3.2 The HTTPS problem

HTTPS traffic presents a direct barrier to traffic analysis required in order to perform malware analysis, effective URL filtering, and blocking of unwanted executable code.

A limited degree of site category filtering, similar to URL filtering, can be achieved for HTTPS traffic without decryption. This is done by classifying resources by target IP address, and performing access control blocking by category without inspecting the content of requests or responses. This filtering is limited, and is not as granular or reliable as full URL filtering. Global content delivery and caching services can also prevent this feature from working effectively for a wide variety of Internet content.

To perform effective monitoring and blocking of malware and executable code, the HTTPS barrier can be overcome in an organisational scenario using technical methods. The most common way to inspect HTTPS traffic is to force the client browsers of an organisation to trust a Certificate Authority (CA) certificate used by the proxy. The proxy then decrypts and re-encrypts traffic for the client, and in this way is able to intercept, monitor, and modify SSL and TLS traffic. This solution requires that all clients trust the proxy’s CA certificate, and though this can be challenging to deploy in large enterprises (due to the diversity of SSL services required to be supported), it is both achievable and desirable.

Monitoring and blocking of threats in HTTPS traffic is technically possible, and improves user security, but many organisations are prevented from doing this because of concerns around the non-technical confidentiality and legal issues. There is a fundamental conflict of interest with inspecting HTTPS traffic, between security of the organisation and its resources, and privacy of the individual for secure personal transactions (such as personal email and Internet banking). It is important for organisations to tackle this issue and find an appropriate balance.

3.3.3 Limiting Information Disclosure

Limiting information exposure regarding which security products are in use is desirable to help defend against targeted attacks against the filtering products in use. Preventing external attackers from enumerating the filtering solution and policy is also important. In any filtering solution, there will be filtering weaknesses which may be exploited if known, to deliver threats to internal users. The remainder of this paper discusses enumeration issues in detail.

4 Enumeration Techniques

The following section describes techniques for enumeration of web filtering solutions, gives examples of information disclosed and shows some statistics collated with a proxy enumeration web application, the Web Filter External Enumeration Tool (WebFEET), used in a variety of scenarios.

Tests were mainly performed in three scenarios:

- Targeting specific products in a controlled test environment, downloading hundreds of payloads to identify product capabilities and weaknesses.
- Detailed analysis during customer engagements, including phishing and client-side attacks, and firewall or web filter policy reviews.
    - These enumeration techniques have been used by NCC Group prior to active attacks against real users
- Limited payloads and tests on site visitors.
    - These tests were conducted on a low-traffic site, with a limited number of test downloads
    - Executable but inert proof-of-concept payloads were used.
    - This testing was mainly used for collecting data for producing discovery signatures, and for gathering statistics and data on typically implemented policies.

4.1 Managed Service Enumeration

An easy form of enumeration can be used to determine if a web filtering managed service is in use. This can be done by the attacker's web server, simply by checking the IP address of connecting clients with "whois", and matching these against a known signature list of companies providing web security services.

For example, for the connecting IP "208.81.64.248":

```
# whois 208.81.64.248| grep OrgName
OrgName:        MX Logic, Inc.
```

This can be automated to quickly and accurately determine known managed services that are used by connecting IP addresses. There is not usually a way to obscure the use of a web filtering managed service, as the IP addresses of systems providing the secure proxy service will be directly connecting to resources on the Internet. This does not present an issue in itself, unless an attacker is aware of specific filtering weaknesses associated with a particular managed service.

In tests, the use of managed service web filtering was found to be rare compared to the use of onsite web security solutions, and browser-based web security features.

4.2 Web Security Product Identification With Port-Scanning

For secure web proxies and browser defences, normally no services should be externally exposed, so it should not possible to identify these web security products externally using typical port-scanning and vulnerability-scanning techniques.

4.3 Product Enumeration With HTTP Header modifications

Secure web proxies often make modifications to HTTP headers as they process responses. One of the simplest enumeration techniques is to capture modifications to HTTP headers. This can be done using a simple web application to request a resource, using a JavaScript “XMLHttpRequest” to make a request within the application’s own domain. Headers are collected from the responses, and then sent back to the server in a POST request for logging.

```
var req = new XMLHttpRequest();
req.open('GET', 'test.txt', false); // Get a simple text test file
req.send(null);

// Extract added header elements which can be passed back to the server or parsed
with regular expressions and used with the client JavaScript
var headers = req.getAllResponseHeaders();
var via = req.getResponseHeader("Via");
var cache = req.getResponseHeader("X-Cache");
var lookup = req.getResponseHeader("X-Cache-Lookup");
```

Figure 3 Proof of Concept JavaScript to collect header modifications

In tests, NCC Group observed that it is standard practice for most web security proxies to add some HTTP headers which contain information which could useful to an attacker. Information disclosed in this way included product vendors and versions, internal IP addresses and hostnames, and proxy ports. Where multiple proxies were configured in a chain, multiple internal hosts were revealed (the following extracts have been redacted to remove some specific confidential information).

```
X-Cache-Lookup: MISS from wp-xxxxxxx.xxx.xx.xx:3128
```

```
X-Cache: MISS from 10.56.106.47
```

Figure 4 Examples of internal IP addresses and hostnames in cache headers

```
Via: 1.0 10.5.222.20 (McAfee Web Gateway 7.2.0.1.0.13253)
```

```
Via: 1.0 barracuda.xxxxxxxxxxxxx.xx:8080 (http_scan/4.0.2.6.19)
```

```
Via: 1.1 xxxxproxy02.xx.xxxxxx.com:3128 (Cisco-IronPort-WSA/7.5.2-118)
```

```
Via: 1.1 backup.xxxx.xxx.xx:3128 (squid/2.7.STABLE9)
```

Figure 5 Examples of product versions, disclosed in “Via” headers

```
X-Cache-Lookup: MISS from xxxxxx:53128, MISS from pfsense:3128
```

```
Via: 1.1 xxxxx-3:80, 1.0 proxy-xxxx (squid/3.1.19), 1.0 xxxxxxxxxxxxxx:3128
(squid/2.6.STABLE9)
```

Figure 6 Multiple proxies chained together can be revealed in either “Via” or cache headers

```
X-Antivirus: avast! 4
X-Antivirus-Status: Clean
```

```
X-WebMarshal-RequestID: 098018AD-177F-4783-AE5F-FF2B8F58CB95
```

Figure 7 Examples of custom X-headers, from the Avast and WebMarshal products

While this level of information disclosure is intended for troubleshooting purposes, and is normally considered a low-severity risk (unlikely to cause problems in isolation), it should be noted that this information is accessible to external attackers. It is best practice to minimise the disclosure, especially where there is disclosure of exact product versions and internal IP addresses.

4.4 URL Category Filter Enumeration

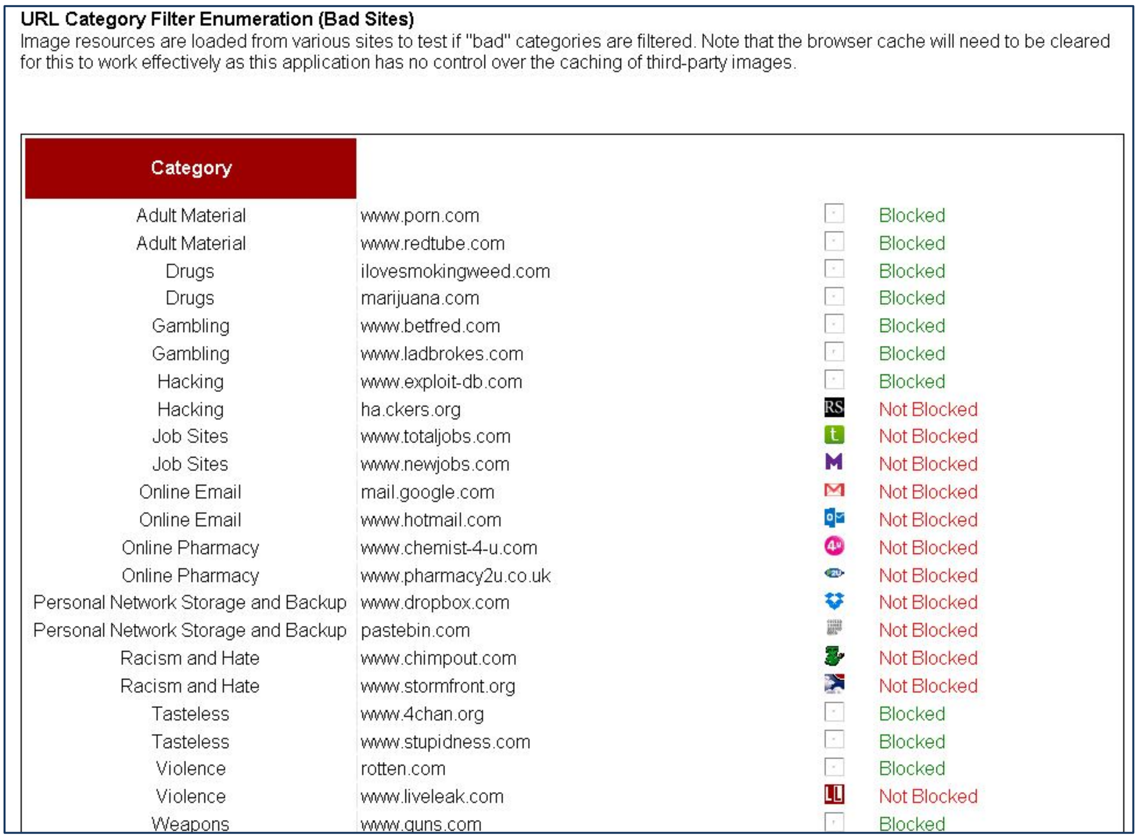
Web filtering products usually implement URL category filtering. The presence of this feature is relatively easy for an attacker to enumerate, by making a simple web application which uses HTML to load resources into the web application DOM from sites in various categories.

Though the resources loaded are not on the same site, Same Origin Policy (SOP) bypass is not required to confirm that the site resources are accessible. JavaScript “onload” or “onerror” directives can be used to update the application DOM for images that successfully load. In this way, small site images can be loaded from reliable locations on a wide variety of sites, such as by using website favicons, to assess whether the sites are accessible to the browsing user. Once the test images are all loaded, either the updated DOM or an array of results can be posted back to the enumerating application server.

```
<tr>
  <td>Adult Material</td>
  <td>www.porn.com</td>
  <td><img src='http://www.porn.com/favicon.ico' height = 16 width = 16
onload="document.getElementById('www.porn.com').innerHTML = '<font
color=red>Yes</font>'"></td>
  <td><span id = "www.porn.com"><font color=green>No</font></span></td>
</tr>
```

Figure 8 An example of loading favicon image resources to detect if sites are allowed or blocked

Using this method a wide variety of sites can be quickly tested, by loading single images from many sites, to identify if URL category filtering is enabled, which categories are filtered, and confirm whether the filtering is effective and configured according to best practice guidelines.



URL Category Filter Enumeration (Bad Sites)

Image resources are loaded from various sites to test if "bad" categories are filtered. Note that the browser cache will need to be cleared for this to work effectively as this application has no control over the caching of third-party images.

Category			
Adult Material	www.porn.com		Blocked
Adult Material	www.redtube.com		Blocked
Drugs	ilovesmokingweed.com		Blocked
Drugs	marijuana.com		Blocked
Gambling	www.betfred.com		Blocked
Gambling	www.ladbrokes.com		Blocked
Hacking	www.exploit-db.com		Blocked
Hacking	ha.ckers.org	RS	Not Blocked
Job Sites	www.totaljobs.com	t	Not Blocked
Job Sites	www.newjobs.com	M	Not Blocked
Online Email	mail.google.com	M	Not Blocked
Online Email	www.hotmail.com		Not Blocked
Online Pharmacy	www.chemist-4-u.com	4u	Not Blocked
Online Pharmacy	www.pharmacy2u.co.uk		Not Blocked
Personal Network Storage and Backup	www.dropbox.com		Not Blocked
Personal Network Storage and Backup	pastebin.com		Not Blocked
Racism and Hate	www.chimpout.com		Not Blocked
Racism and Hate	www.stormfront.org		Not Blocked
Tasteless	www.4chan.org		Blocked
Tasteless	www.stupidness.com		Blocked
Violence	rotten.com		Blocked
Violence	www.liveleak.com	LL	Not Blocked
Weapons	www.guns.com		Blocked

Figure 9 WebFEET output; here we see URL category filtering implemented, but not effectively

This particular type of enumeration is more useful for an IT security audit than it is for an external attacker focusing on a targeted attack, but it is still useful for an attacker to some degree because it can provide a quick confirmation that a web security proxy or other URL filter is in place.

The above URL category filter enumeration can work well, but it is important to note a couple of specific limitations.

1) URL category filtering may only work accurately the first time it is run, because image cache control directives are supplied by third-party categorised sites rather than the enumeration application.
2) Not all images load equally on all browsers; results can vary between browser vendors and versions.

If results are inconsistent for sites in the same category, this may be because of cross-browser support issues with a particular image, or the fact that a particular image had already been cached in the browser when the test was run, or that the site categorisation is not reliable.

4.5 Malware and File-type Policy Enumeration

File-type restriction policies can be enumerated by making a series of download and upload requests of various types. Differences in response can be analysed to enumerate the implemented policy automatically.

4.5.1 Detecting Policy Enforcement for File Downloads

When a variety of file download requests are made via a web security proxy, there are various potential outcomes to consider. Here are some examples:

1) The requested file is downloaded.
    a. This means that the file download was not filtered by the proxy.
2) No data is returned, the request was received at the webserver but the response was dropped by the proxy.
    a. This results in a browser timeout.
    b. Normally one should assume that the response has been blocked.
3) A request is made by the client but no request is received at the webserver.
    a. The request is blocked – perhaps by file extension.
4) Unexpected data is returned in the response. This could be:
    a. A policy block page – meaning the download was blocked. This often has additional confidential information which may be useful to an attacker.
    b. The original file is truncated or modified – meaning the file is effectively blocked.

Some of the possible scenarios are shown in the diagrams below:

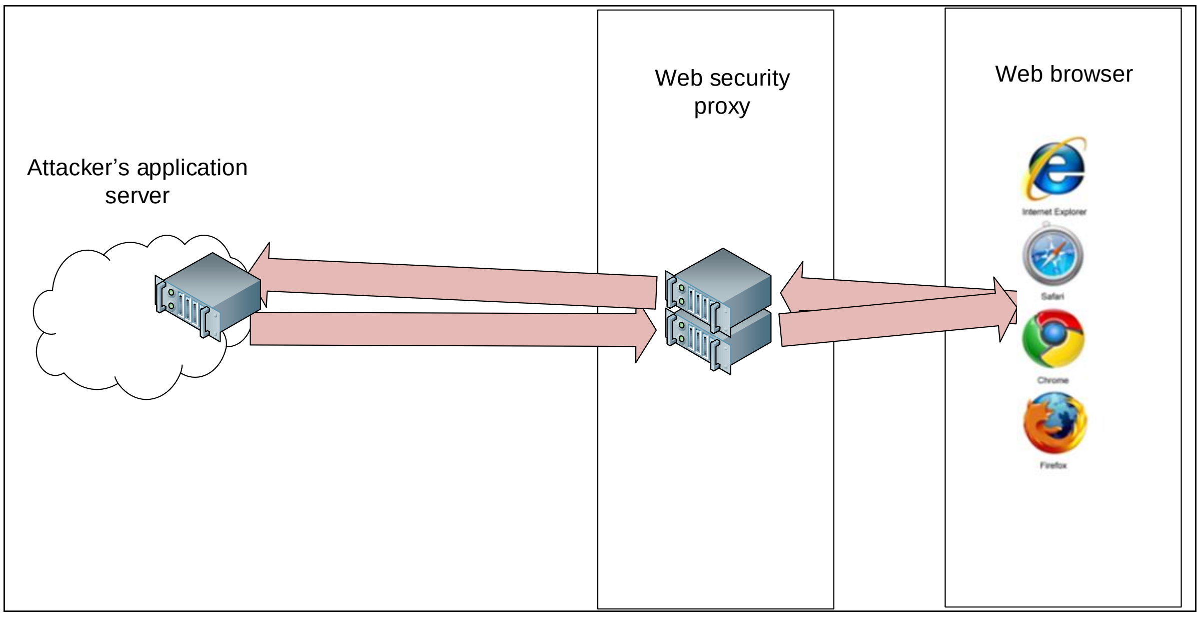


Figure 10 A series of requests are made to the attacker's application server

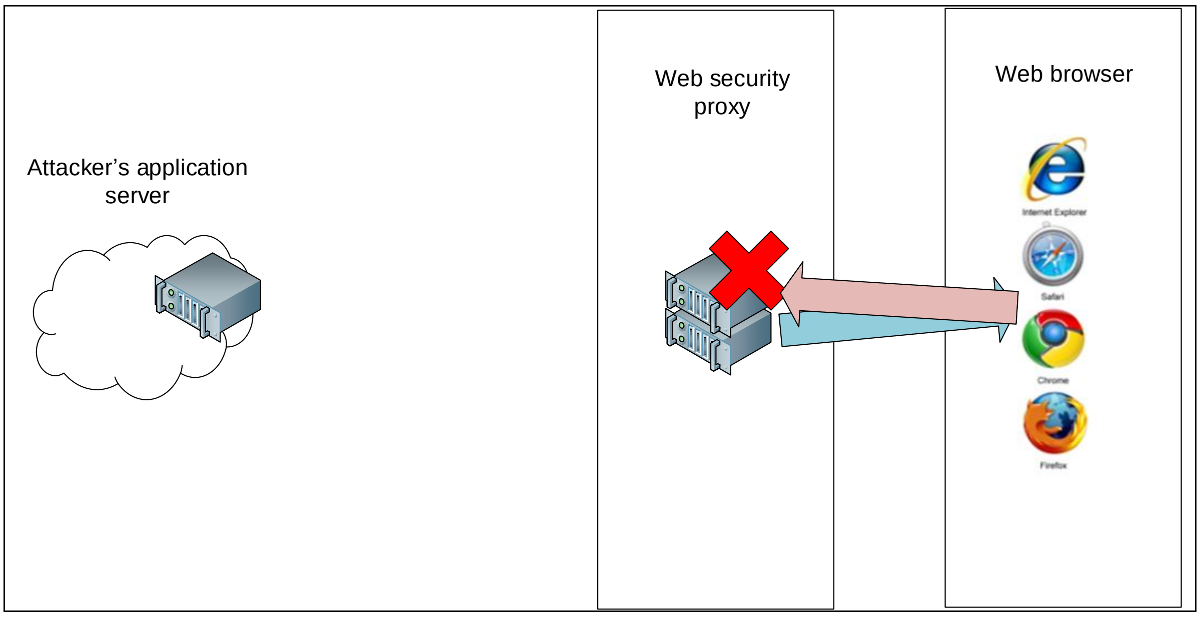


Figure 11 The request is blocked and block-page returned

The enumeration application is designed to request a series of file downloads and check the responses to see if the original file is downloaded, or if there is another outcome. Secure web proxies can block requests, or responses, but typically send a block-page response to the browser when a threat is detected. Examples of typical block-pages are shown later in this paper.

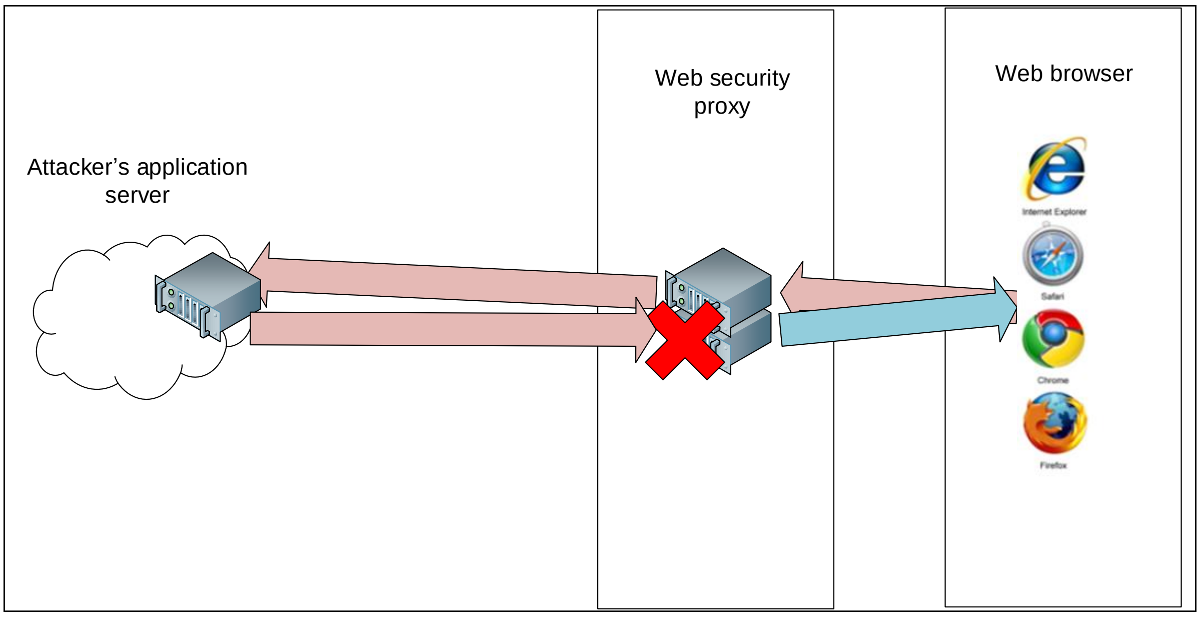


Figure 12 The response is blocked and block-page returned

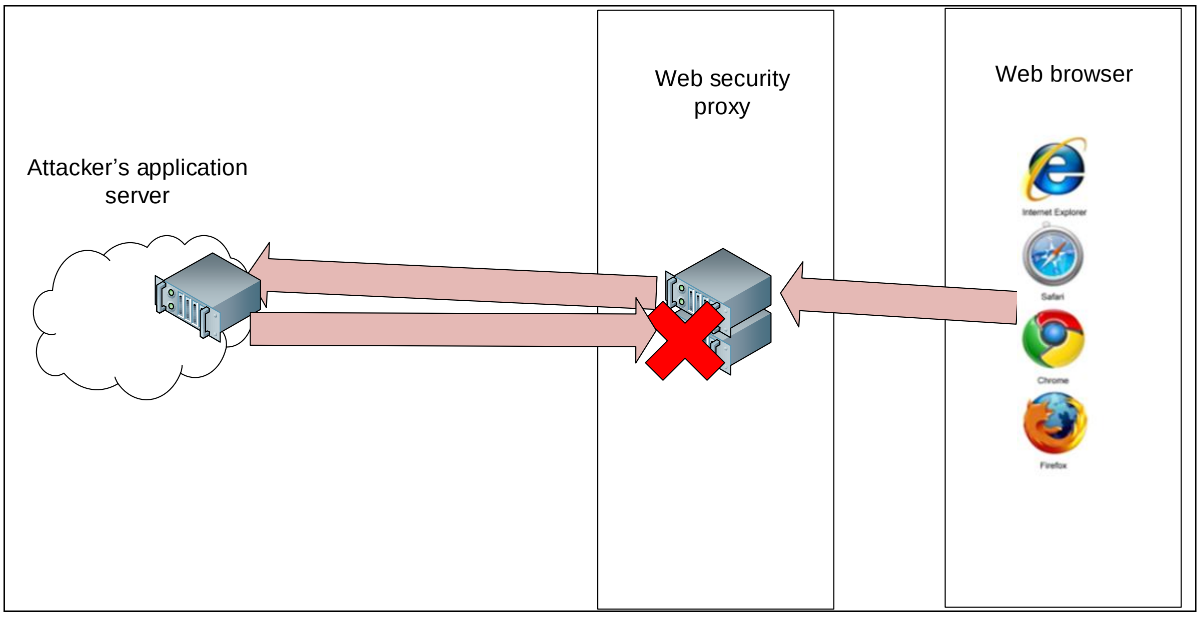
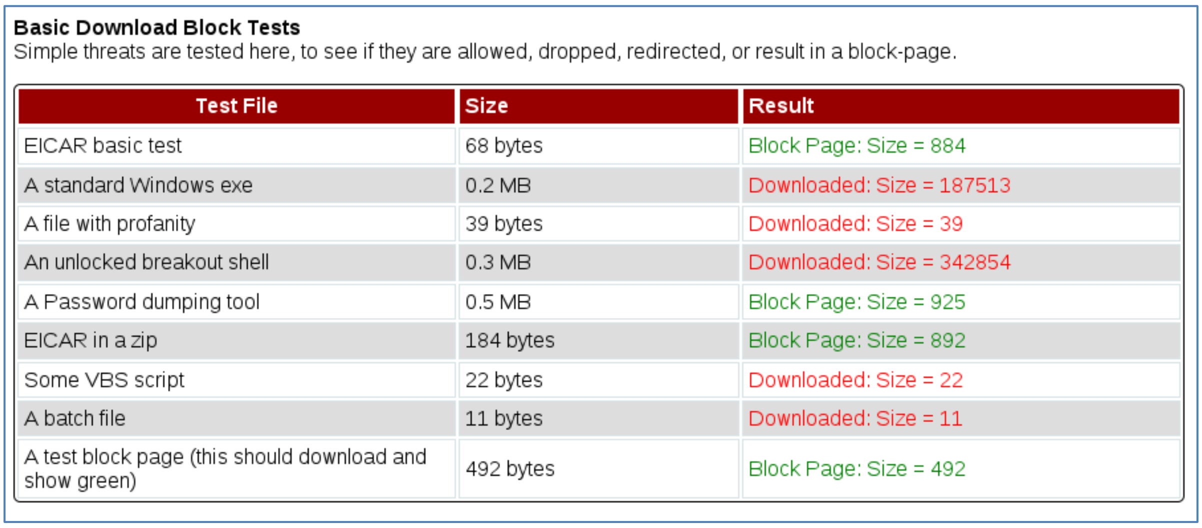


Figure 13 The response is dropped, which results in browser timeout

Other combinations are possible, but the key point to consider is that a variety of requests can be made by the client-side JavaScript, and the results recorded and sent back to the enumerating application. An example of the results of a small number of requests is shown below:



Basic Download Block Tests
Simple threats are tested here, to see if they are allowed, dropped, redirected, or result in a block-page.

Test File	Size	Result
EICAR basic test	68 bytes	Block Page: Size = 884
A standard Windows exe	0.2 MB	Downloaded: Size = 187513
A file with profanity	39 bytes	Downloaded: Size = 39
An unlocked breakout shell	0.3 MB	Downloaded: Size = 342854
A Password dumping tool	0.5 MB	Block Page: Size = 925
EICAR in a zip	184 bytes	Block Page: Size = 892
Some VBS script	22 bytes	Downloaded: Size = 22
A batch file	11 bytes	Downloaded: Size = 11
A test block page (this should download and show green)	492 bytes	Block Page: Size = 492

Figure 14 Example results for a small number of file download tests

4.5.2 Block-Page Policy and Product Enumeration

Usually when a request or response is blocked by policy, an informational block-page is returned to the browser. This has three purposes; to block the threat, inform the user, and prevent the browser from hanging, waiting for a timeout.

Block-pages are delivered to the client with two main types of response, which have a different impact in respect to product enumeration. These are:

1) A response with a block-page inline.
2) A response with a redirect to a block-page.

If the response is modified with the block-page inline (usually returning a HTTP response code 200). This results in the block-page being within the same origin (SOP) as the enumeration application, which means that the block-page is accessible to the enumeration application's client-side JavaScript. The block-page can be parsed client-side, or passed back to the enumeration application server and logged.

If the response is modified to cause a HTTP response code 301 or 302 to redirect the browser to a block-page on a web server on the proxy, this results in the JavaScript seeing no data (due to SOP). It is clear that the original file was not received and what the policy is, but there is less direct product disclosure as the block-pages are not directly accessible to an external attacker. However, this does usually mean that block-pages are served on a webserver on a specific port on the proxy (which means that it may be possible to load resources from that web server to check which product it is).

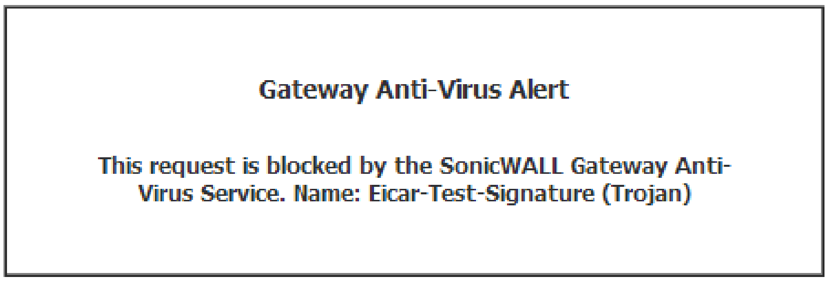
Here is some example JavaScript which attempts to download a small set of files and sends back any resulting block-pages to the enumerating application. JQuery is used here to simplify browser compatibility. The file set can be extended to hundreds of test files, to enumerate differences in detection capability for file-type, file extensions, MIME-type and embedded threats:

```
<script>
var files = ['exe-in-rtf.rtf','exe-popup.exe','eicar.com'];

// Multi-file Downloader
$(files).each(function() {
  var filedownload = this;
  $.get( "payloads/"+filedownload, function( data ) {

    var jsSrcRegex = /<html(.+?)/igm; //Regex to find HTML block pages
    jsSrcRegex.compile(jsSrcRegex);

    if (jsSrcRegex.exec(data)) {
    document.getElementById(filedownload).innerHTML = "Block Page: Size = " +
data.length;
    // Post back the blockpage to the enumerating application
    $.post( 'upload.php', {  block : data, name : filedownload } );
    } else if (data === "") {
       document.getElementById(filedownload).innerHTML = "Blocked: Size = " +
data.length;
    } else {
       document.getElementById(filedownload).innerHTML = "Downloaded: Size = " +
data.length;
    }
  });
});
</script>
```

Figure 15 A series of trigger files are downloaded and any resulting block-pages uploaded back to the enumerating web application

During testing, policy block-pages were found to be returned to the enumerating web application in around 90% of tests where secure web proxies were present. These pages often revealed sensitive information about the web filtering products used, the policy, and other data such as client information, including internal IP addresses, hostnames, usernames, domain names, and client software versions.

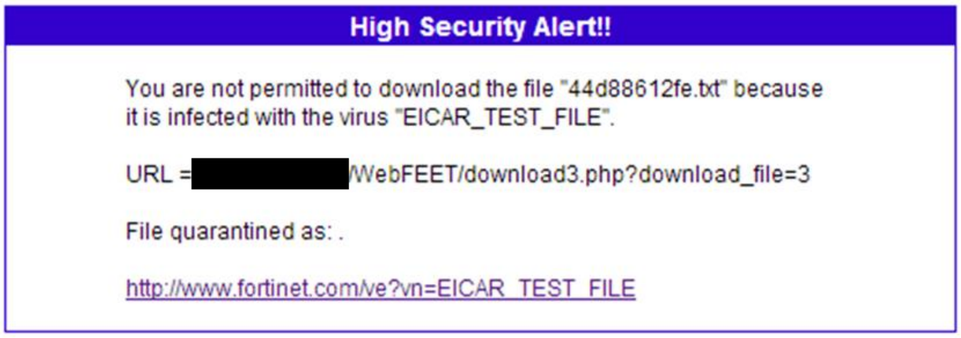
Here are some examples of collected policy block-pages and some of the information they revealed. It may be surprising to note that these were all collected externally using the block-page collection techniques described above, and that these are a small fraction of those collected in a short period on a low-traffic website.



Gateway Anti-Virus Alert

This request is blocked by the SonicWALL Gateway Anti-Virus Service. Name: Eicar-Test-Signature (Trojan)

Figure 16 Some products give limited detail, such as this SonicWALL Gateway block-page



High Security Alert!!

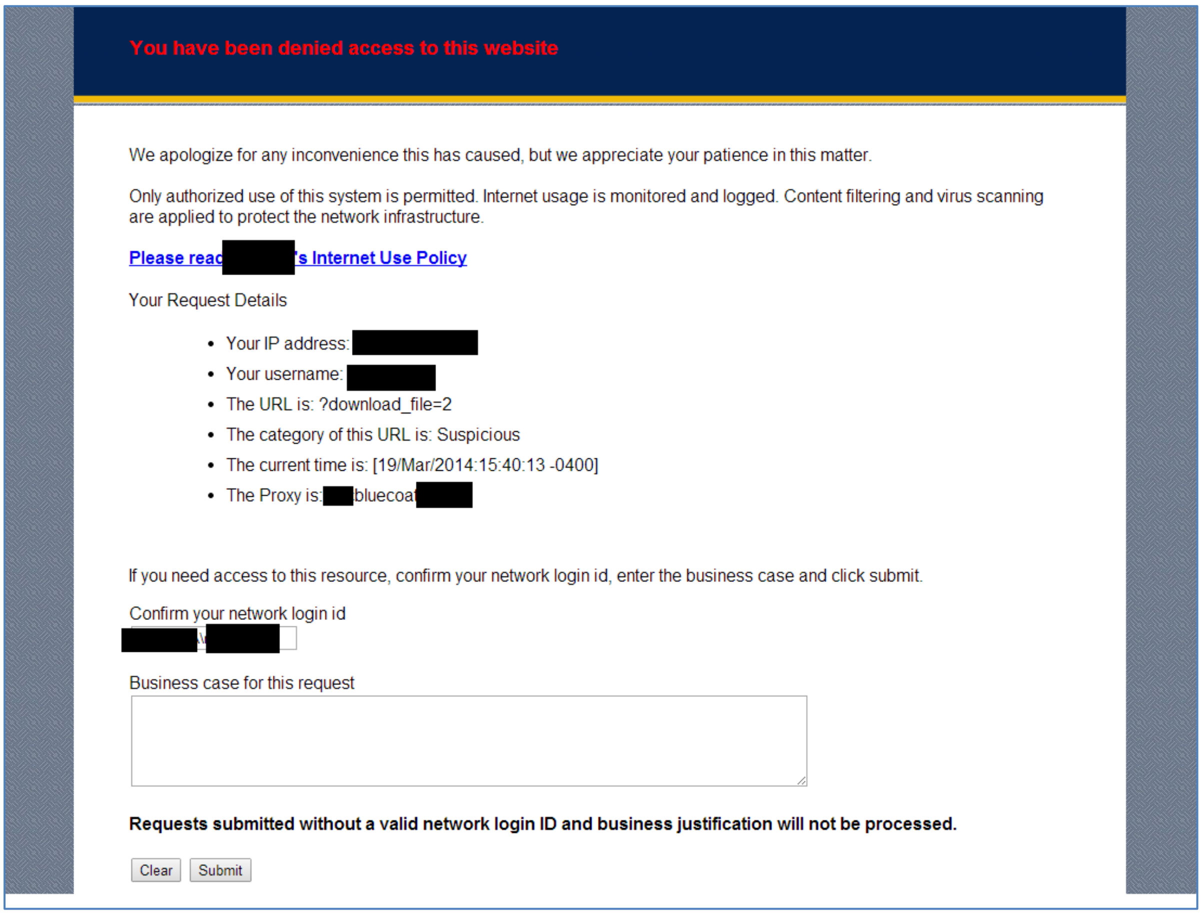
You are not permitted to download the file "44d88612fe.txt" because it is infected with the virus "EICAR_TEST_FILE".

URL = [redacted]/WebFEET/download3.php?download_file=3

File quarantined as: .

http://www.fortinet.com/ve?vn=EICAR_TEST_FILE

Figure 17 A similar example from the Fortinet product



You have been denied access to this website

We apologize for any inconvenience this has caused, but we appreciate your patience in this matter.

Only authorized use of this system is permitted. Internet usage is monitored and logged. Content filtering and virus scanning are applied to protect the network infrastructure.

Please read [redacted]'s Internet Use Policy

Your Request Details

- Your IP address: [redacted]
- Your username: [redacted]
- The URL is: ?download_file=2
- The category of this URL is: Suspicious
- The current time is: [19/Mar/2014:15:40:13 -0400]
- The Proxy is: [redacted]bluecoat[redacted]

If you need access to this resource, confirm your network login id, enter the business case and click submit.

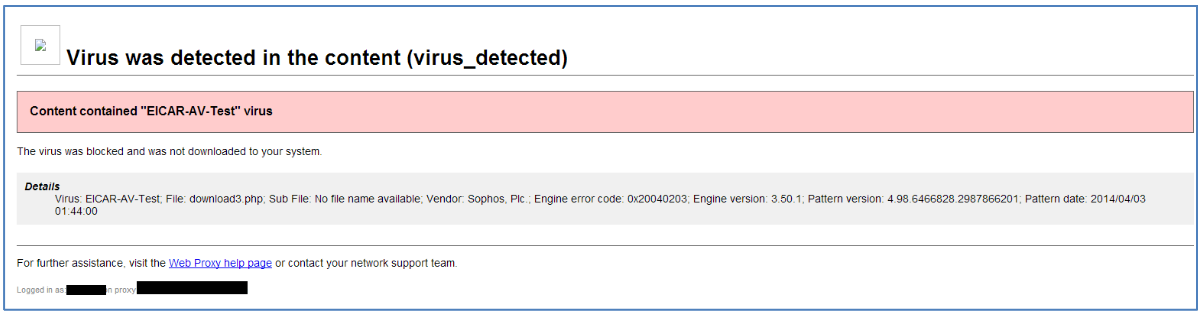
Confirm your network login id

[redacted] [redacted]

Business case for this request

Requests submitted without a valid network login ID and business justification will not be processed.

Figure 18 A block-page from a bluecoat proxy reveals lots of information



Virus was detected in the content (virus_detected)

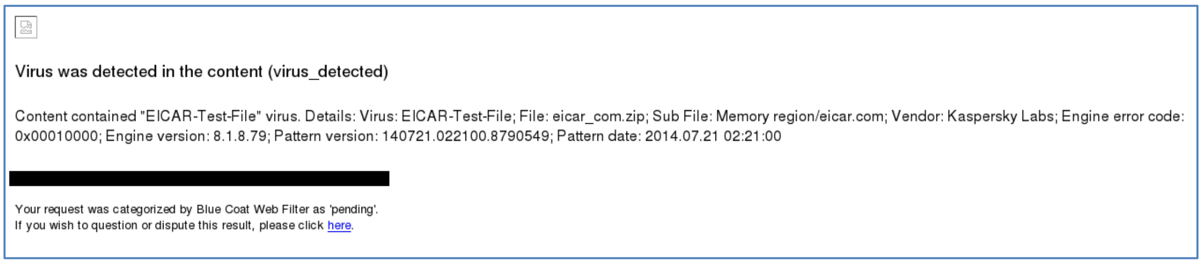
Content contained "EICAR-AV-Test" virus

The virus was blocked and was not downloaded to your system.

Details
Virus: EICAR-AV-Test; File: download3.php; Sub File: No file name available; Vendor: Sophos, Plc.; Engine error code: 0x20040203; Engine version: 3.50.1; Pattern version: 4.98.6466828.2987866201; Pattern date: 2014/04/03 01:44:00

For further assistance, visit the Web Proxy help page or contact your network support team.

Logged in as [redacted]n proxy [redacted]



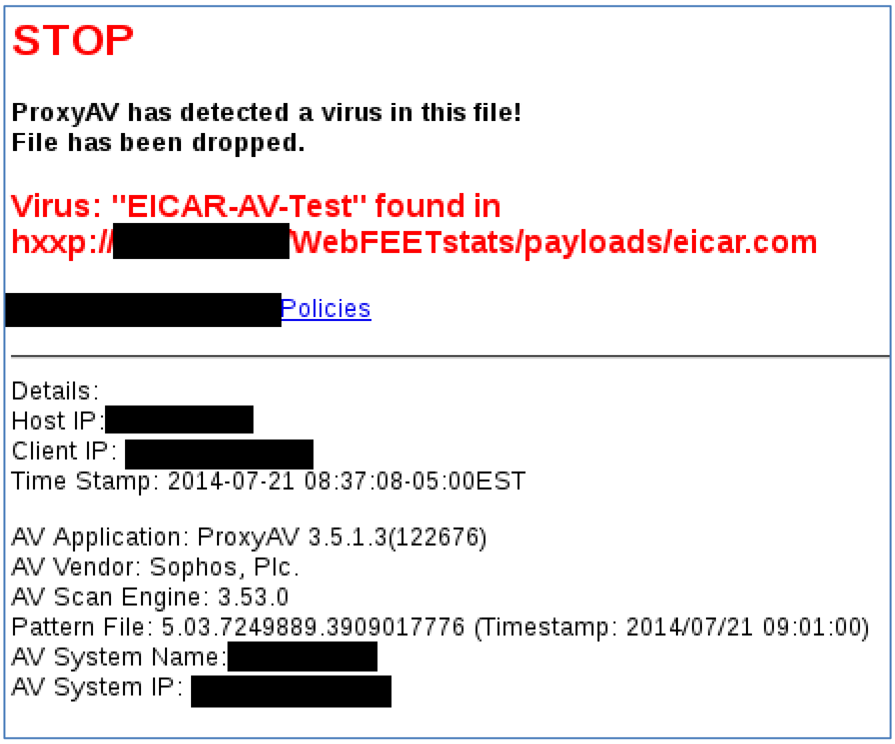
Virus was detected in the content (virus_detected)

Content contained "EICAR-Test-File" virus. Details: Virus: EICAR-Test-File; File: eicar_com.zip; Sub File: Memory region/eicar.com; Vendor: Kaspersky Labs; Engine error code: 0x00010000; Engine version: 8.1.8.79; Pattern version: 140721.022100.8790549; Pattern date: 2014.07.21 02:21:00

[redacted]

Your request was categorized by Blue Coat Web Filter as 'pending'.
If you wish to question or dispute this result, please click here.

Figure 19 Two very similar block-pages show a proxy with multiple antivirus plugin options



STOP

ProxyAV has detected a virus in this file!
File has been dropped.

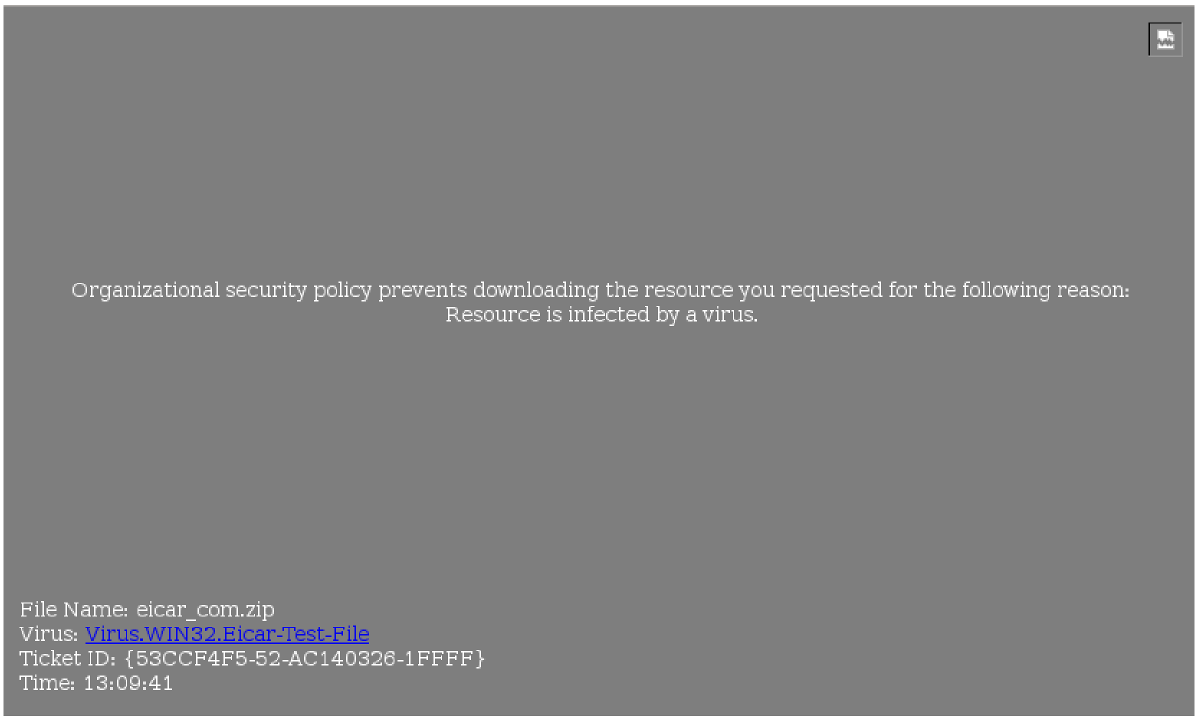
Virus: "EICAR-AV-Test" found in
hxxp://[redacted]WebFEETstats/payloads/eicar.com

[redacted]Policies

Details:
Host IP:[redacted]
Client IP: [redacted]
Time Stamp: 2014-07-21 08:37:08-05:00EST

AV Application: ProxyAV 3.5.1.3(122676)
AV Vendor: Sophos, Plc.
AV Scan Engine: 3.53.0
Pattern File: 5.03.7249889.3909017776 (Timestamp: 2014/07/21 09:01:00)
AV System Name:[redacted]
AV System IP: [redacted]

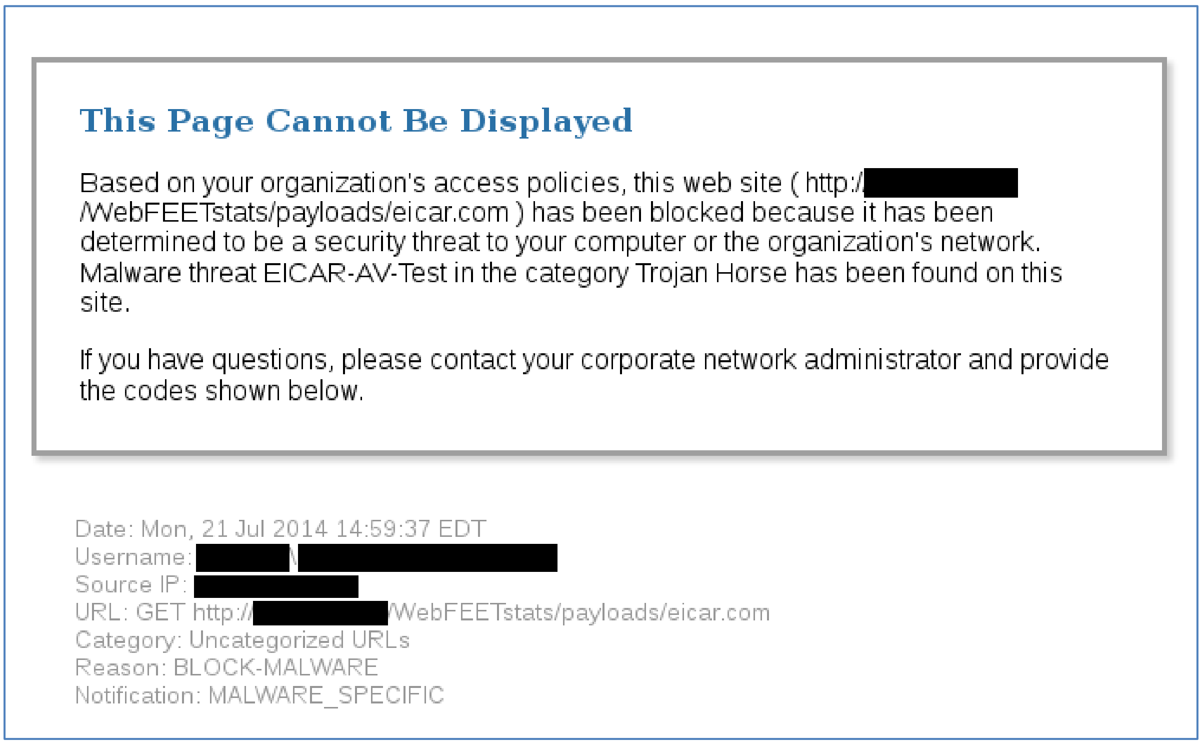
Figure 20 ProxyAV with the Sophos plugin



Organizational security policy prevents downloading the resource you requested for the following reason:
Resource is infected by a virus.

File Name: eicar_com.zip
Virus: Virus.WIN32.Eicar-Test-File
Ticket ID: {53CCF4F5-52-AC140326-1FFFF}
Time: 13:09:41

Figure 21 Sometimes it is not always obvious in the rendered HTML, but in the HTML source it could be seen that this was a block-page from a Checkpoint firewall



This Page Cannot Be Displayed

Based on your organization's access policies, this web site (http:/[redacted]/WebFEETstats/payloads/eicar.com) has been blocked because it has been determined to be a security threat to your computer or the organization's network. Malware threat EICAR-AV-Test in the category Trojan Horse has been found on this site.

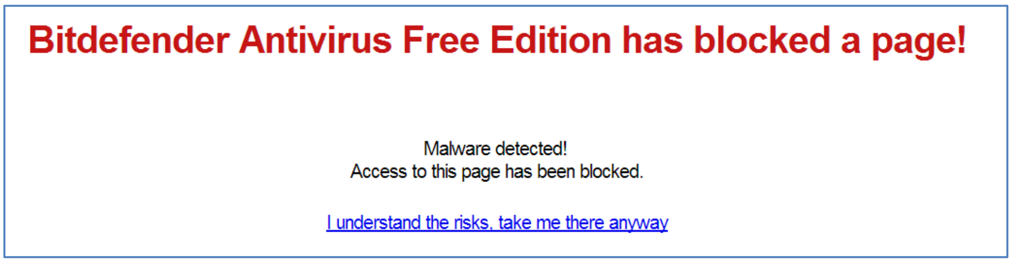
If you have questions, please contact your corporate network administrator and provide the codes shown below.

```
Date: Mon, 21 Jul 2014 14:59:37 EDT
Username: [redacted]\[redacted]
Source IP: [redacted]
URL: GET http://[redacted]/WebFEETstats/payloads/eicar.com
Category: Uncategorized URLs
Reason: BLOCK-MALWARE
Notification: MALWARE_SPECIFIC
```

Figure 22 A Cisco Ironport Web Security appliance discloses a variety of information

4.5.3 Enumeration of Desktop Antivirus Browser Plugins Via Block-Pages

As an unexpected side effect of the proxy block-page enumeration technique, NCC Group found that desktop antivirus plugins often produce similar block-pages which are also collected with the same technique. This presents an interesting way to enumerate the version of desktop antivirus in use and the effectiveness of its browser plugin.

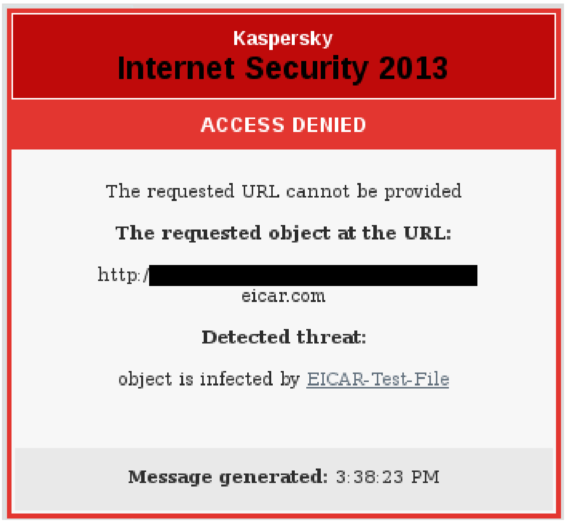


Bitdefender Antivirus Free Edition has blocked a page!

Malware detected!
Access to this page has been blocked.

I understand the risks, take me there anyway

Figure 23 A Bitdefender antivirus browser plugin block page



Kaspersky
Internet Security 2013

ACCESS DENIED

The requested URL cannot be provided

The requested object at the URL:

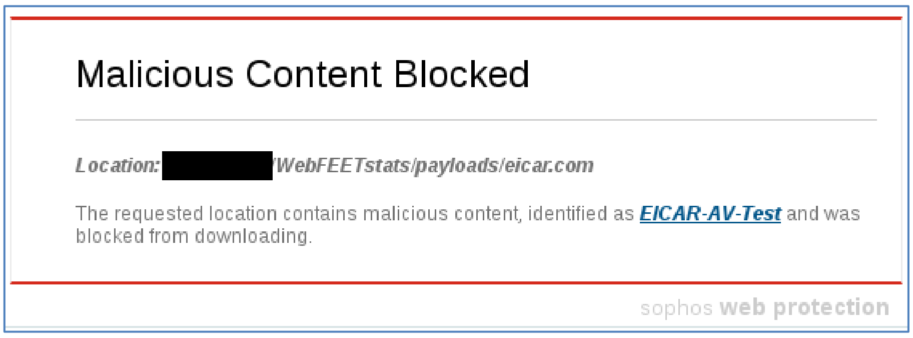
http:/[redacted]
eicar.com

Detected threat:

object is infected by EICAR-Test-File

Message generated: 3:38:23 PM

Figure 24 A Kaspersky antivirus browser plugin block page



Malicious Content Blocked

Location: [redacted]*/WebFEETstats/payloads/eicar.com*

The requested location contains malicious content, identified as ***EICAR-AV-Test*** and was blocked from downloading.

sophos **web protection**

Figure 25 A Sophos antivirus browser plugin block page

There are a variety of ways of detecting installed software via a browser in a drive-by way (which are actively exploited by attackers in the wild[4]), but this technique represents an additional means by which desktop antivirus browser plugins could be detected.

4.5.4 Multilayer Enumeration With Block Pages

Where both a web security proxy and an antivirus browser plugin are implemented, the effective policy is the sum of the two blocking policies. Interestingly, the two different layers of block-pages can be enumerated by testing a wide variety of threats, ensuring that some threats get through the proxy layer and trigger the browser plugin.

If no HTTPS inspection is in place then there will be a clear difference in block pages returned for threats requested over HTTPS (which we discuss later).

4.6 Product Type and Version Enumeration With Targeted Resource Loading

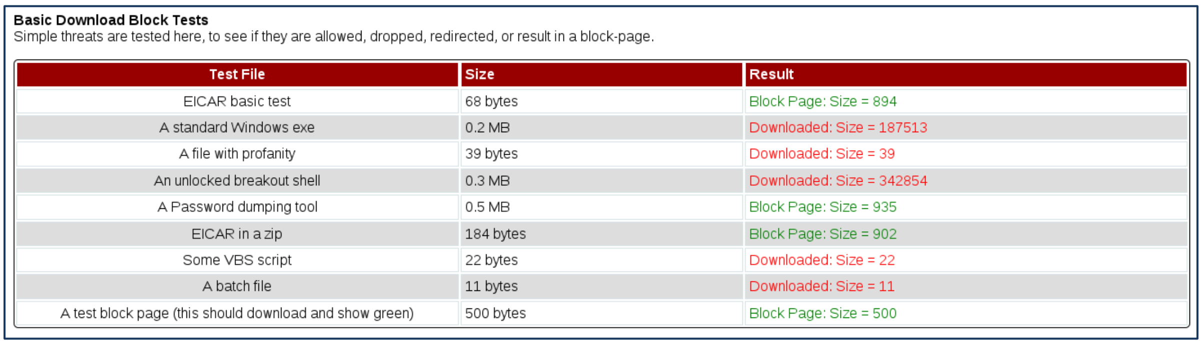
In some cases where block-pages are not accessible, due either to a block-page redirect or the request being dropped, it may not be possible to enumerate the proxy type and version in this way. However, using a similar method to the URL category filter enumeration techniques it is sometimes possible to use the disclosure of the proxy address, to try to load images or other resources from a web server on the proxy itself, to assist in detecting the product in use.

Examples of web services which would be running on the proxy include administrative UIs, and web services hosting block-pages, help pages, or other content intended for end users. By creating signatures for the various web filtering products, detailing known image locations on accessible web services, it was shown to be possible to enumerate a variety of specific web security products which may otherwise have been hidden.

[4] Attackers abusing Internet Explorer to enumerate software and detect security products
http://www.alienvault.com/open-threat-exchange/blog/attackers-abusing-internet-explorer-to-enumerate-software-and-detect-securi

4.7 Enumeration of HTTPS Inspection Capabilities

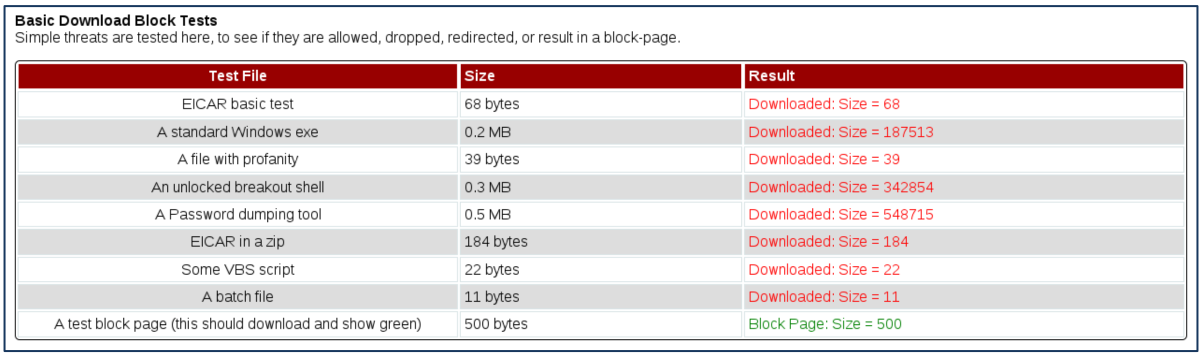
For detecting the capability of HTTPS inspection the file download tests can be performed for HTTP and again using HTTPS (this simply requires a valid SSL certificate on the attacker's server to work transparently). If no HTTPS inspection is taking place, then there will be a marked difference between the detection capabilities, with no file-download-based threats being detected by the proxy when requested via HTTPS.



Basic Download Block Tests
Simple threats are tested here, to see if they are allowed, dropped, redirected, or result in a block-page.

Test File	Size	Result
EICAR basic test	68 bytes	Block Page: Size = 894
A standard Windows exe	0.2 MB	Downloaded: Size = 187513
A file with profanity	39 bytes	Downloaded: Size = 39
An unlocked breakout shell	0.3 MB	Downloaded: Size = 342854
A Password dumping tool	0.5 MB	Block Page: Size = 935
EICAR in a zip	184 bytes	Block Page: Size = 902
Some VBS script	22 bytes	Downloaded: Size = 22
A batch file	11 bytes	Downloaded: Size = 11
A test block page (this should download and show green)	500 bytes	Block Page: Size = 500

Figure 26 WebFEET output: a small number of tests show some threats are blocked



Basic Download Block Tests
Simple threats are tested here, to see if they are allowed, dropped, redirected, or result in a block-page.

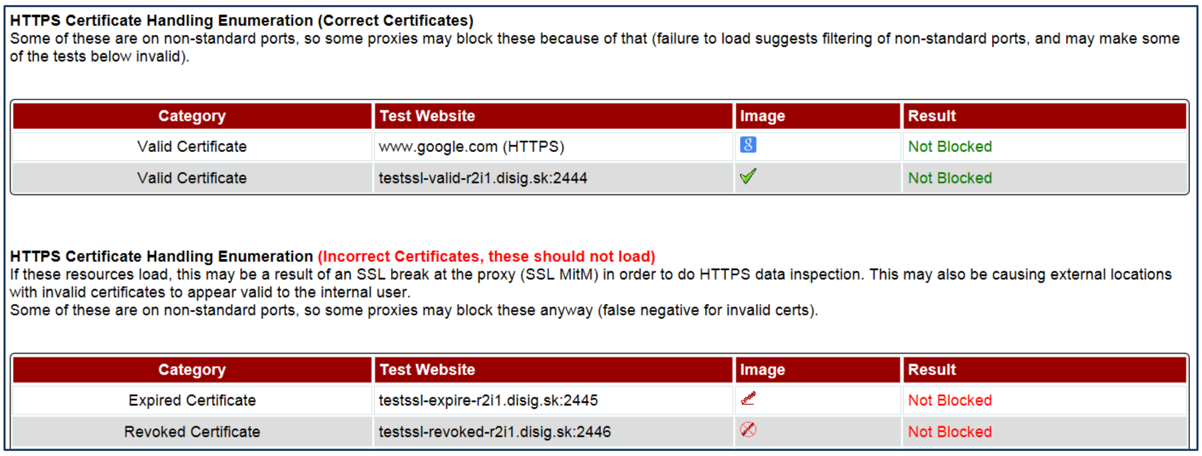
Test File	Size	Result
EICAR basic test	68 bytes	Downloaded: Size = 68
A standard Windows exe	0.2 MB	Downloaded: Size = 187513
A file with profanity	39 bytes	Downloaded: Size = 39
An unlocked breakout shell	0.3 MB	Downloaded: Size = 342854
A Password dumping tool	0.5 MB	Downloaded: Size = 548715
EICAR in a zip	184 bytes	Downloaded: Size = 184
Some VBS script	22 bytes	Downloaded: Size = 22
A batch file	11 bytes	Downloaded: Size = 11
A test block page (this should download and show green)	500 bytes	Block Page: Size = 500

Figure 27 The same tests run over HTTPS show no inspection or blocking in this case

If no filtering is in place for HTTPS, this gives an attacker a great advantage in terms of sending attack payloads, and exfiltration of data and remote shells.

An additional way to detect HTTPS inspection capabilities is to try to load resources from sites which have known invalid certificates. If the resources load, then this indicates that the browser trusts the certificate it received, which may indicate that the certificate has been replaced by one signed by the proxy which is valid and trusted by the client[5] (which is a way that some secure web proxies can perform HTTPS inspection, with MitM for the SSL/TLS traffic).

[5] Jeff Jarmoc – SSL/TLS Interception Proxies and Transitive Trust http://www.secureworks.com/cyber-threat-intelligence/threats/transitive-trust/



HTTPS Certificate Handling Enumeration (Correct Certificates)
Some of these are on non-standard ports, so some proxies may block these because of that (failure to load suggests filtering of non-standard ports, and may make some of the tests below invalid).

Category	Test Website	Image	Result
Valid Certificate	www.google.com (HTTPS)	g	Not Blocked
Valid Certificate	testssl-valid-r2i1.disig.sk:2444	✓	Not Blocked

HTTPS Certificate Handling Enumeration (Incorrect Certificates, these should not load)
If these resources load, this may be a result of an SSL break at the proxy (SSL MitM) in order to do HTTPS data inspection. This may also be causing external locations with invalid certificates to appear valid to the internal user.
Some of these are on non-standard ports, so some proxies may block these anyway (false negative for invalid certs).

Category	Test Website	Image	Result
Expired Certificate	testssl-expire-r2i1.disig.sk:2445		Not Blocked
Revoked Certificate	testssl-revoked-r2i1.disig.sk:2446	⊘	Not Blocked

Figure 28 With HTTPS inspection, invalid certificates are sometimes replaced by valid certificates at the proxy, resulting in resources which would not be accessible loading without errors

4.8 What Typically Gets Through?

In a variety of tests, focus was given to try to identify the simplest way to enable victims to download and run arbitrary executable code or scripts, which allowed the transmission of attacker's code, while being able to execute on the widest variety of corporate desktop environments and bypass the majority of filters.

During tests, it was found to be common for URL filtering and known-malware detection to be implemented on secure web proxies, but relatively rare for unknown executable code and scripts to be effectively filtered at the proxy. In cases where unknown executable code was being filtered, the filtering was bypassed in most of the test cases where executables or scripts were embedded in common office documents or archives (which most office workers could open and run). Filtering of encrypted attachments such as password-protected office documents was not seen at all in tests, and no policies were observed which effectively prevented code execution attacks.

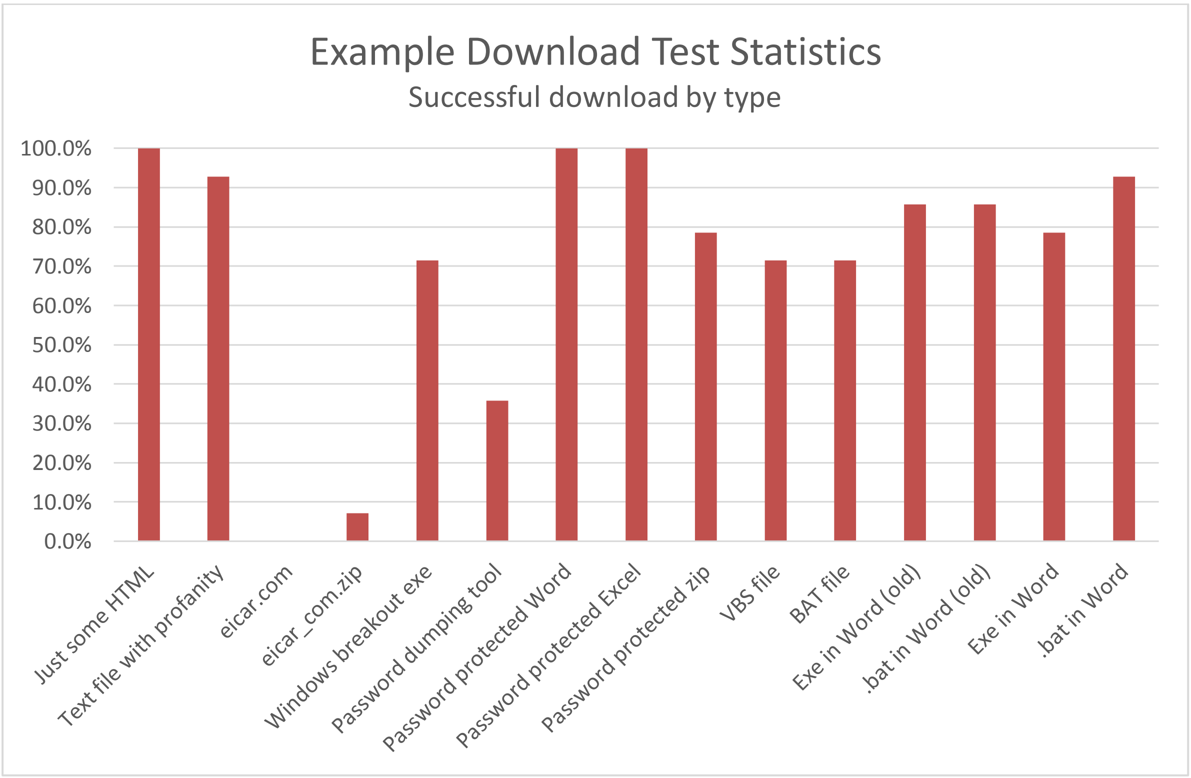
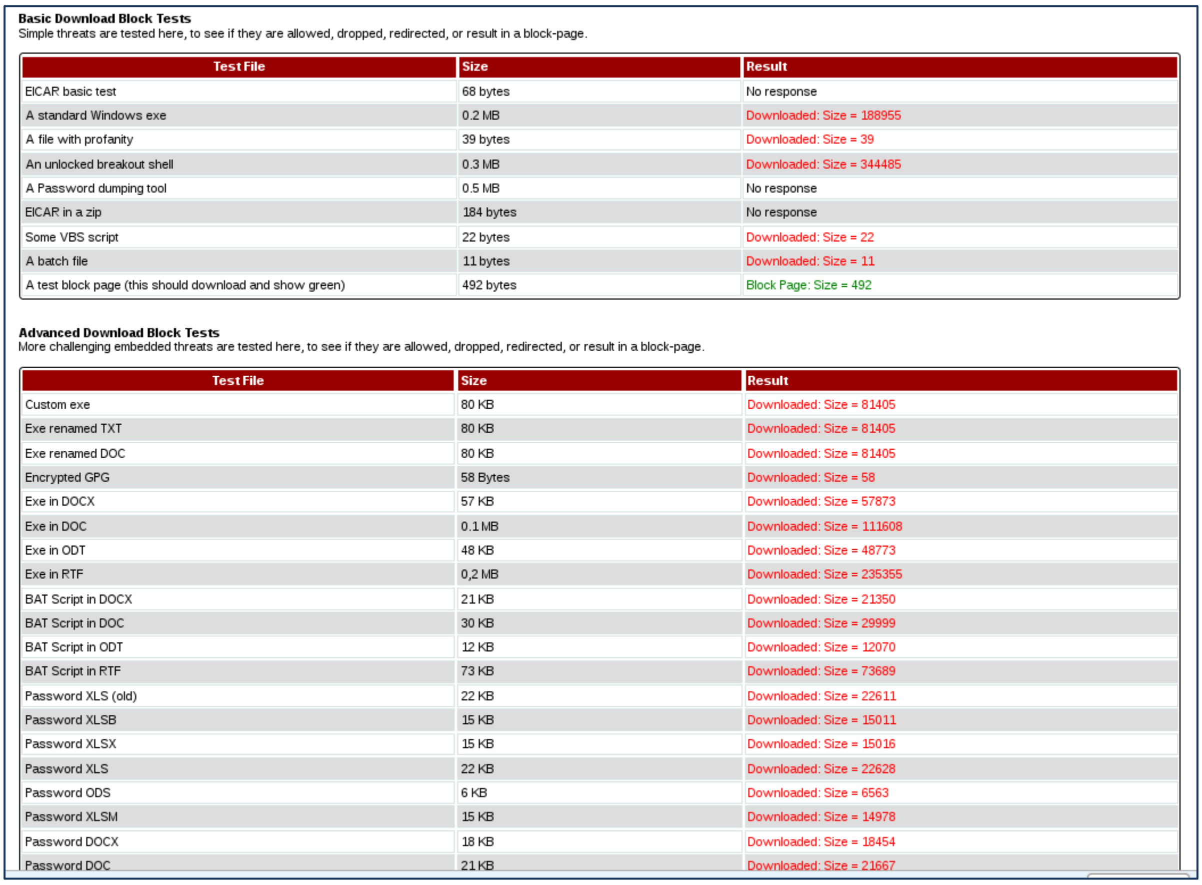


Figure 29 Typically, basic threats are filtered but embedded threats or executable code are not

Clearly many organisations are choosing to allow documents with macros, unknown executable code and script downloads, and password-protected files for business reasons. HTTPS interception and filtering was not encountered during these tests. Unfiltered HTTPS traffic, which could contain any of the previously mentioned threats, is usually allowed in most organisations, often to protect the confidentiality of personal Internet browsing, such as banking websites and other confidential services. In short, web traffic was found to be much less filtered than email traffic (assessed during previous research by NCC Group).



Basic Download Block Tests
Simple threats are tested here, to see if they are allowed, dropped, redirected, or result in a block-page.

Test File	Size	Result
EICAR basic test	68 bytes	No response
A standard Windows exe	0.2 MB	Downloaded: Size = 188955
A file with profanity	39 bytes	Downloaded: Size = 39
An unlocked breakout shell	0.3 MB	Downloaded: Size = 344485
A Password dumping tool	0.5 MB	No response
EICAR in a zip	184 bytes	No response
Some VBS script	22 bytes	Downloaded: Size = 22
A batch file	11 bytes	Downloaded: Size = 11
A test block page (this should download and show green)	492 bytes	Block Page: Size = 492

Advanced Download Block Tests
More challenging embedded threats are tested here, to see if they are allowed, dropped, redirected, or result in a block-page.

Test File	Size	Result
Custom exe	80 KB	Downloaded: Size = 81405
Exe renamed TXT	80 KB	Downloaded: Size = 81405
Exe renamed DOC	80 KB	Downloaded: Size = 81405
Encrypted GPG	58 Bytes	Downloaded: Size = 58
Exe in DOCX	57 KB	Downloaded: Size = 57873
Exe in DOC	0.1 MB	Downloaded: Size = 111608
Exe in ODT	48 KB	Downloaded: Size = 48773
Exe in RTF	0,2 MB	Downloaded: Size = 235355
BAT Script in DOCX	21 KB	Downloaded: Size = 21350
BAT Script in DOC	30 KB	Downloaded: Size = 29999
BAT Script in ODT	12 KB	Downloaded: Size = 12070
BAT Script in RTF	73 KB	Downloaded: Size = 73689
Password XLS (old)	22 KB	Downloaded: Size = 22611
Password XLSB	15 KB	Downloaded: Size = 15011
Password XLSX	15 KB	Downloaded: Size = 15016
Password XLS	22 KB	Downloaded: Size = 22628
Password ODS	6 KB	Downloaded: Size = 6563
Password XLSM	15 KB	Downloaded: Size = 14978
Password DOCX	18 KB	Downloaded: Size = 18454
Password DOC	21 KB	Downloaded: Size = 21667

Figure 30 It is common to see very little filtering of custom executable code and encrypted files

5 Further Research and Findings

5.1 Extending Capability Detection

A key difference between the enumeration of email filtering (discussed in a separate paper) and the enumeration of web filtering is that in the techniques described here to enumerate web filtering the attacker has much more control. This is because client-side code can be used to make arbitrary bidirectional tests, and send results back to the enumerating application. For this reason, as long as the enumeration JavaScript runs, it should be possible to perform a wide variety of additional tests, such as testing for content modifications, for features which remove content such as Java applets or other thick-client code.

It would be possible to extend the WebFEET tool to detect policy for different types of requests and responses, and to test for outbound policy via upload enumeration by encoding or otherwise hiding test files, and then using the client-side JavaScript to attempt to upload them back to the webserver.

5.2 Injection Vulnerabilities Within Block-Pages

Though the focus of this research was to identify enumeration techniques, rather than find new vulnerabilities in web security products, several serious issues were identified. One issue found in multiple products was XSS vulnerabilities in block-pages.

Often block-pages contain the URL that was blocked. If this is reflected with the parameters included, and the parameters are not sanitised (with input validation and output encoding) then this presents an interesting attack-vector for proxy-based XSS. For example, if a request is made as follows:

```
http://www.example.org/uploads/eicar.txt?dummy=<script>alert("XSS
everywhere!")</script>
```

This can trigger XSS which would affect the www.example.org site and its session-cookies. Sometimes this issue can affect a wide variety of sites because:

- Web security block-pages that are inline responses rather than redirects are in the SOP of the request or response being blocked. In other words the block-page is SOP with the site where the threat is detected.
- Some web security appliances have XSS in their block-pages, which means that if you can induce a block-page for any given site, you can potentially trigger a generic XSS for that site; for example:
    - If an organisation has a policy to block executable code, and the web security proxy block-page has XSS, then this introduces XSS into the SOP of any site with downloadable executables or scripts.
    - Any site which has EICAR, or to which it is possible to upload eicar.txt (for example) or induce it to be reflected, potentially has the same problem.
- Some web security products block files for generic reasons, for example any files with executables embedded in Word documents.
    - So again, any site to which an attacker can upload these files could be vulnerable to reflective XSS when visited through a web security proxy affected by this issue.
- This issue potentially gets worse when the organisation's policy is stricter, as when more file types are restricted more sites can be affected.

6 Conclusion

6.1 Successful Enumeration Was Achieved

In this paper we have demonstrated how an external attacker could enumerate products, services, and policies used for web filtering, and shown how policy bypass misconfigurations or product capability shortfalls can be identified. We have also discussed the results from tests showing common weakness in implemented web security policy.

The root causes of most of the policy and product discovery issues would typically be considered low severity information disclosure issues, but in combination the disclosure issues can give an external attacker the ability to build a clear picture of a target organisation's web filtering solution in advance of an attack. Some of the issues seen during this research appear to be widespread and systemic, and as the responsibility for addressing issues lies with a wide variety of product vendors, service providers, and consumer organisations, these issues are likely to persist for some time.

In regard to the policies found to be implemented, analysis was based on a limited data-set from a relatively small number of tests. Despite this shortcoming, the results for the general weaknesses of policy were surprising. When these techniques are combined with NCC Group's recent research on email filtering enumeration and policy weaknesses, they can place an attacker in a much stronger position prior to a targeted attack.

More enumeration functionality may be possible with thick client code such as Flash, Java or Silverlight, but these technologies have additional dependencies, and all enumeration goals were achieved by using simple JavaScript.

6.2 Recommendations for Web Security Implementers

This research shows that reducing information disclosure and improving both policy and capability can greatly reduce exposure to potential threats from targeted enumeration and client-side attacks. Various defensive configurations can be implemented to reduce the likelihood and impact of enumeration.

6.2.1 Minimise Information Disclosure

The main enumeration method employed during this research relied on client-side JavaScript to request a variety of resources, interpret responses, and post details back to the enumerating application. It would be challenging to defend against this type of enumeration attack, but perhaps a defensive measure may be to temporarily block further requests from a client which has requested a blocked resource. Product features which may help include features which prevent further enumeration by closing a tab, or stopping JavaScript execution within the browser at the client-side, or providing a time-based restriction of additional requests from a client at the web security proxy (once a single request has been blocked).

Other defensive configurations include:

- Deploying script filtering features on clients such as the no-script plugin (though this does have many disadvantages with the modern internet being largely dependent on JavaScript).
- Run web security proxy services such as web servers for admin UIs, and block-pages on non-standard ports where possible.

- Check what headers are added to responses, and limit disclosure where possible:
    - Use generic hostnames rather than IP addresses (as this reduces the information disclosed regarding configured IP address ranges used by the target organisation).
    - Try not to put the product names or similar references in the hostnames.
    - Ensure that product versions are not disclosed.
- For block-pages:
    - Consider that any content presented on a policy block-page is potentially accessible to external attackers.
    - Block-pages can be a source of proxy-based XSS so ensure that parameters are stripped, or effective input validation and output encoding is performed (if not, do not display the URL).
    - If it is possible to redirect, to a generic block-page, which is not on the proxy, then this may be worth considering.

6.2.2 Tackling Arbitrary Executable Code and Scripts

In addition to filtering URLs and known malware, it is also important to address the risk from unknown executable code and scripts. This is because it is relatively easy for attackers with development skill to generate malicious code which is not detected by antivirus. There are also various frameworks for encoding known threats in such a way so they are not detected by antivirus.

Recent exploits should ideally be identified especially where proof-of-concept code is in the public domain. As an additional form of mitigation, sensitive organisations may wish to additionally filter high risk data types such as Flash, Java applets, PDFs and macro-enabled office documents. It may be difficult for an organisation to apply this policy to all internal users (due to the negative effect on business processes) though it may be possible to apply stricter policies to some groups of users to reduce overall risk.

6.2.3 Tackling Encrypted Connections and Content

A lack of interception and filtering for HTTPS traffic presents a considerable attack vector for most organisations. It would be achievable for many organisations to inspect HTTPS traffic to and from user systems, but there is a reluctance for organisations to take this step, due to confidentiality issues, for example with personal banking sites used by employees.

One option for sensitive sites is to split the policy so that traffic to especially sensitive URL categories such as personal banking sites is not inspected while other categories, and uncategorised sites, are subjected to HTTPS inspection. This can be challenging, but a balance needs to be found between data confidentiality and protecting users and systems.

In addition to encrypted HTTPS traffic, encrypted documents and archives can present a significant risk to organisations. As filtering solutions typically cannot be configured to decrypt password protected attachments, policies are often configured to let these attachments pass unprocessed. Similarly to HTTPS this introduces a loophole in web filtering which attackers can easily exploit. Therefore, uncontrolled use of encrypted attachments should be strictly limited by policy where possible.

Organisations should try to reduce risk by limiting access to encrypted content and traffic, and this can be done by applying appropriate policies to groups of users and sites. If this is not done attackers will always have the advantage of being able to deliver malicious files and other content to end users.