NativeSPSpoofChecking

- Background
- General Settings
- Platform Notes
  - IIS / ISAPI
  - Sun / NSAPI
  - Apache
  - FastCGI

Background

SP releases starting with 1.3.1 have included a feature in some of the filters/modules called "spoof checking". The purpose of this feature is to actively scan the HTTP headers in each client request and attempt to detect if any of them map to a header variable under the SP's "control". The SP maintains a list of all the possible header names that it might be expected to create in response to a user session, including all of the mappings based on attributes. If the client attempts to supply its own value for one of these headers, this feature tries to detect that and block the request with an error message (and log the attempt).

In general, and this is noted below, you should **always** favor environment variables to request headers if the server platform supports that option. Environment variables cannot be influenced by the client and are much safer. This feature is ignored in such cases.

This feature has been enabled by default for a while, but has frequently required disablement because of "false alarms" caused by internal server redirects and rewrites. The code saw the headers that the SP itself added to the original request and assumes they're from the client, tripping the error.

Starting with version 2.2 (and 1.3.2), this feature should be left enabled, and should be safe to use, as well as significantly more robust.

The following provides more detail on how to ensure the feature is active and how to avoid false alarms without compromising the feature.

Note that this feature is **NOT** the only measure taken by the SP to prevent spoofing. The SP still clears or removes any headers that it controls using a variety of algorithms. This feature is an additional measure of protection, because those algorithms are complex and have a history of not proving to be foolproof.

General Settings

There are two settings that control the operation of this feature, both found in the `<InProcess>` configuration element (or the `<Local>` element in the case of 1.3.x).

The `checkSpoofing` property is a flag that is enabled when omitted, and is set only to explicitly disable the feature. While this was occasionally necessary in the past, you should ensure that this setting is `true` or remove it entirely going forward.

The other property is called `spoofKey`, and is now supported across all the relevant module implementations as a way of preventing false alarms triggered by server-side behavior. There are specific notes in the platform-specific sections below, but in general, all the implementations support the use of an explicit value in this setting.

The `spoofKey` value is intended to be a long, random string of alphanumerics that is hidden from clients. The SP uses this value to distinguish between requests from a client and requests to which the SP has already added headers. It explicitly creates an extra header containing this key, and the theory is that if the client can't guess it, it can't fool the SP into bypassing detection. The SP assumes if the header and value is present, the request has already passed the detection step.

For obvious reasons, you **MUST** prevent the client from accessing any server-side scripts that might expose the `spoofKey` value through a dump of arbitrary (or all) request headers.

Scripts like this are often used in debugging problems by "dumping" the request variables available to applications. Note that blocking or removing such scripts is a standard server-hardening measure that should not be unusual or unfamiliar.

Platform Notes

IIS / ISAPI

The ISAPI filter API does not appear to support the creation of environment variables, so request headers are used out of necessity as a portable communication channel to applications. As a result, the detection feature should be enabled in all cases. To facilitate this, the ISAPI filter module will automatically generate a random `spoofKey` value if one is not set for it, and if it can't do so it will refuse to run.

Note that on older versions of Windows, generally Windows 2000 or NT, the 2.x filter will refuse to load because the API used to generate secure random data is not available. You will need to explicitly create the `spoofKey` setting there (see NativeSPInProcess).

Certain configurations of Windows 2000, possibly involving recent patches from Microsoft, appear to alter the behavior of the SP such that prior to version 2.2.1, the filter will cause IIS to crash instead of simply failing with a Windows log event noting that the `spoofKey` can't be generated. If you're crashing on Windows 2000, make sure you create the setting yourself.

If you experience issues or false alarms with the random key value, you can explicitly set one that will be shared by all IIS server processes and should prevent those problems (at the cost of less randomness, obviously).

Sun / NSAPI

The information described for IIS applies to the NSAPI module as well.

Apache

The Apache modules support request headers for backward compatibility, but environment variables are used by default in 2.x and can be turned on in 1.3.x.

Under **no** circumstances should you rely on the request header option other than as a temporary measure while adjusting applications to use the environment option. There are no known scenarios in which environment variables can't be used, including with Java containers, though sometimes extra effort or Apache settings may be needed. Do **NOT** take shortcuts with this. Do the work and use them.

If for some inexplicable reason you choose not to do this, then you may need to manually add a random `spoofKey` setting to the configuration yourself to avoid false alarms from the spoof detection feature. Because Apache is a multi-process web server, automatically generating a key to use isn't currently supported. Ideally, I suggest running without it for a while and only adding the setting if you have problems.

FastCGI

The FastCGI mechanism for adding information to the request is somewhat convoluted, but it relies on environment variables, so this feature does not apply.