Environment Variables and Security

Server-Side Web Languages

Uta Priss
School of Computing
Napier University, Edinburgh, UK

Outline

Environment Variables

- Environment variables are a means for server-side web languages to exchange information between the server and the client.
- They control the information which is disclosed by either server or client or which refers to a specific HTTP request.
- Not all environment variables are always available.

Client-Side Information

Environment variables which contain client-side information:

HTTP_USER_AGENT	type and version of the client's browser client
HTTP_ACCEPT	accepted MIME types
REMOTE_ADDR	IP address of the client

Sever-Side Information

Environment variables which contain server-side information:

SERVER_SOFTWARE	software used for webserver (e.g. Apache)
SERVER_NAME	name of server
SERVER_PROTOCOL	protocol used by server (e.g. HTTP/1.1)

Request-Specific Information

Environment variables which contain information about a specific HTTP request:

SCRIPT_NAME	the URL of the script
REQUEST_METHOD	usually either GET or POST
QUERY_STRING	form parameters as part of URL (only for GET)
CONTENT_LENGTH	(only for POST)
HTTP_COOKIE	content of a cookie
REMOTE_USER	username - if authentication is used
HTTP_REFERER	URL of previous page

Can environment variables be trusted?

Can environment variables be trusted?

- The value of environment variables cannot be trusted.
- For example: clients can lie about which type of browser they use.
- HTTP_REFERER should never be used for security because it can be modified by a client.
- For highly secure applications, the content of cookies should be encrypted because otherwise it can be modified by a client.

Why can environment variables not be trusted?

Why can environment variables not be trusted?

Because they are usually transmitted as plain text via the HTTP protocol. A client can use a scripting language (such as Perl) to write a mini-browser. Using this mini-browser, all environment variables can be read and modified.

What are the security problems associated with server-side programs?

What are the security problems associated with server-side programs?

- In contrast to stand-alone applications, which often run within a single process, server-side applications require at least two processes: one on the client and at least one on the server.

What are the security problems associated with server-side programs?

- In contrast to stand-alone applications, which often run within a single process, server-side applications require at least two processes: one on the client and at least one on the server.
- The information exchange between these two processes occurs as plain text over the web (unless SSL is used and the communication is encrypted).

What are the security problems associated with server-side programs?

- In contrast to stand-alone applications, which often run within a single process, server-side applications require at least two processes: one on the client and at least one on the server.
- The information exchange between these two processes occurs as plain text over the web (unless SSL is used and the communication is encrypted).
- Without encryption, neither the client nor the server can trust the other one.

...

What are the security problems associated with server-side programs? - continued

- Client security depends on the browser. A client should make sure that the browser is up-to-date and patched.

What are the security problems associated with server-side programs? - continued

- Client security depends on the browser. A client should make sure that the browser is up-to-date and patched.
- The server needs to conduct security checks not just once per session but in fact **every time a new request is received, i.e. every time the client submits a page.**

What are the security problems associated with server-side programs? - continued

- Client security depends on the browser. A client should make sure that the browser is up-to-date and patched.
- The server needs to conduct security checks not just once per session but in fact **every time a new request is received, i.e. every time the client submits a page.**
- These security problems apply equally to all scripting languages (Perl, PhP, ASP, etc) but some languages are more open about the problems while others keep more details hidden from the programmers.

Specific Security Issues

- A client could embed HTML code including Javascript within formdata. A possible exploit is defacing, i.e. the display of a page on the server could be changed. **Solution:** remove all HTML tags and all unnecessary special characters from formdata before doing anything else with the data.

Specific Security Issues

- A client could embed HTML code including Javascript within formdata. A possible exploit is defacing, i.e. the display of a page on the server could be changed. **Solution:** remove all HTML tags and all unnecessary special characters from formdata before doing anything else with the data.
- A client can change variable names and values independently of how these names and values appear on a webform. **Solution:** all formdata must be carefully checked. It should never be assumed that a correct form was used.

Specific Security Issues

- A client could embed HTML code including Javascript within formdata. A possible exploit is defacing, i.e. the display of a page on the server could be changed. **Solution:** remove all HTML tags and all unnecessary special characters from formdata before doing anything else with the data.
- A client can change variable names and values independently of how these names and values appear on a webform. **Solution:** all formdata must be carefully checked. It should never be assumed that a correct form was used.
- The worst security risk is a client obtaining access to information or processes outside the HTML environment on the server. **Solution:** Interaction between client data and other server processes (such as databases, email, reading and writing files) should be avoided or at least carefully monitored.

...

Specific Security Issues - continued

- Malicious code could be installed on a server (as a Trojan horse). **Solution:** All software that is installed on a server and runs under the webserver (such as CGI scripts) should be carefully checked. It would be dangerous to just download some code from somewhere on the web and install it on the server.

Specific Security Issues - continued

- Malicious code could be installed on a server (as a Trojan horse). **Solution:** All software that is installed on a server and runs under the webserver (such as CGI scripts) should be carefully checked. It would be dangerous to just download some code from somewhere on the web and install it on the server.
- A script that sends email back to clients could be abused for creating spam. **Solution:** Automatic email sending should be avoided. A distinction between registered and unregistered users might be helpful. Emails from unregistered users should be forwarded to the server and checked by a human.

...

Specific Security Issues - continued

- Denial of Service and other attacks that attempt to use up a server's resources. **Solution:** limits should be set for how much CPU time and memory can be used by a server-side script. Ideally these limits should be set by the webserver, but some checking can also be performed by the script itself.

Specific Security Issues - continued

- Denial of Service and other attacks that attempt to use up a server's resources. **Solution:** limits should be set for how much CPU time and memory can be used by a server-side script. Ideally these limits should be set by the webserver, but some checking can also be performed by the script itself.
- And finally: all server-software should be kept up-to-date. Programmers should read up on security issues that pertain to server-side languages and applications.