Prepared web pages and links — version 2022/I

Egg, February 7, 2022: *A short email, then a phone call. In October 2021, a company approaches Archivista GmbH. They had carried out a security audit for a customer. WebDMS and WebAdmin were vulnerable to attack via prepared websites or links. The customer could not be named, but the attack was documented and the document could be accessed via a link (with password). What follows is a longer odyssey. This blog is about that Blog.*



What is cross-site scripting all about?

All modern web applications work with two layers. Firstly, there is the presentation of the content in the web browser on the end devices. On the other side are the servers (e.g. the ArchivistaBox) on which the data is processed. JavaScript is usually used for the presentation of the content. This means that information can be displayed and changed very flexibly (quasi on-the-fly) in the user interface (e.g. WebDMS), without the need for all data to be retransmitted between the client (WebDMS) and the server (ArchivistaBox) with each call.
The disadvantage of JavaScript in terms of security is that the corresponding programs are processed directly in the web browser, because this delegates sovereignty to an external partner (web browser). And even if JavaScript programs are displayed illegibly, the sources are open source. The code can be viewed in the web browser without much effort and (much more “unpleasant”) can also be changed. The called server cannot determine whether its external partner works in good or bad mission, i.e. with changed sources.
The only control the server has over its client (web browser) is that the sources are loaded from the server when they are called. Whether the delivered program is processed correctly is not decided by the server (ArchivistaBox), but by the respective web browser. With so-called cross-site scripting, also known as XSS attacks, this is exactly what happens. The JavaScript code is compromised in the web browser.

Reflected attacks

A subtype of such attacks is reflected attacks. In this case, the “attack” takes place via an external link that is delivered, for example, via prepared PDF documents or e-mails. If unsuspecting users click on it and enter their user name and password without thinking, the external server “intercepts” the login data and can then access the desired web application without any problems using the data obtained in this way.
It is important to know that the login data is entered on an external computer. If,

when entering data, you check whether the navigation bar of your web browser contains the address of the desired server, you cannot be “duped” in this way. A subtype of the reflected attack consists in the fact that the desired website (e.g. ArchivistaBox) is called, but JavaScript code is transmitted directly in the process. Attached is an example:

http://192.168.0.99/perl/avclient/index.pl?target=_top&host=localhost"><script>alert("Hacking 1")</script>&db="><script>alert("Hacking 2")</script>&uid="><script>alert("Hacking 3")</script>&uid=Admin&db=archivista&pwd=xxxx

To be able to execute the example, the IP address of an ArchivistaBox must be entered for the IP address 192.168.0.99. Apart from the fact that the above link is relatively long, it is not primarily “Hacking X” that should make you suspicious (a “malicious” hacker would not greet you with ‘Hacking X’, but the fragment ‘<script>…</script>’, since it is intended to deliberately “infiltrate” external code.

In principle, every server should check whether such code fragments are transmitted to it. Correctly parsed, the fragment ‘<script>…</script>’ would then simply be output as text in the corresponding field of the user, i.e. the “infiltrated” code is processed as text and not as script. Unfortunately, this control was poorly done in earlier versions of WebDMS. Although it was not possible to drop such code fragments when working in WebDMS after logging in, unfortunately no such control was performed for the login form.



Is my ArchivistaBox affected by this?

Basically all previous ArchivistaBoxes (versions before 2022/I) are affected by this, as the fragment ‘<script>…</script>’ was not necessarily converted to text in the login form. Even if such attacks are not possible without the assistance of unsuspecting third parties, the machine (in this case ArchivistaBox) should of course still carry out all possible checks so that such attacks can be avoided as far as possible. By ‘avoided as far as possible’ it is meant that only that which is known can be avoided.

What is to be done?

The answer is to “order” the update and apply it via WebConfig. “Ordering” means that customers with a maintenance contract can receive an updated version free of charge at any time, if they wish (notification by phone, mail or letter is sufficient).

At this point the question may be raised as to why all ArchivistaBox systems are not simply updated automatically. The answer is that this is not provided for in the ArchivistaBox concept, as such updates would require automated access to the ArchivistaBox. As practical as such updates may be, systems which can be updated automatically at any time are “vulnerable” to a far greater extent than when updates are “triggered” by humans.
As already mentioned above, reflected cross-site scripting attacks always require a (more or less) naive cooperation of the user. Anyone who carelessly clicks on links or does not care who is entrusted with this information when entering username and password does not have to be surprised when a takeover occurs. After all, the system is only taken over when the login data is “revealed” on a third-party site. The same situation occurs when the same user data is used in all possible and impossible situations.
Even the best password is of little use if it is used thousands of times on all systems. Likewise, standard passwords should not be used. In this sense, reflected cross-site scripting attacks are a rather minor problem. Nevertheless, it should and may be talked about.

When an unknown third party invites a download

Back to the question of why there is talk of an odyssey at the beginning. The first call was made in October 2021. A person pretends to be a security researcher and refers to a company website based in Germany. With a certain tone, the caller wants the company Archivista to download a document (password protected) from a website. The security vulnerability concerns a customer, an audit has revealed.
Since neither the name of the customer is mentioned, since there is no business relationship with the caller, since the caller is calling from an unknown and unverifiable mobile phone number from a third country, since a look at the stated company homepage does not suggest a website in the narrow sense of a company with hundreds of employees (as mentioned in the phone call), the caller is invited to send the “alleged” gap by mail. This, in turn, the caller does not want at all, as this would be insecure.
The caller is not convinced by the objection that security problems can and should be discussed publicly in open source common sense. He insists that the gap can only be communicated confidentially. Even the suggestion to send the document in letter form (with/without USB stick) finds no mercy. At some point, the Archivista company refers the caller to the federal cyberattack reporting center. The caller emphasizes that should the document not be downloaded, such a report would be made and that the security report would then be published publicly.



National reporting center only accessible via web form

At the beginning of February, a call is received from a Mr. Knoepfel, who works at the national center for cyber security. He asks for a callback. Unfortunately, it is not possible to reach Mr. Knoepfel. Furthermore, it is relatively difficult to check whether Mr. Knoepfel is really an employee of the said federal office, because on the homepage of the **Swiss Federal Cyberattack Reporting Center** there is only a web form to enter reports. If you look for a telephone number, you will not find one.

Those who want to know who works at the office in question would have to register via Linkedin.com, the link **https://www.linkedin.com/company/ncsc-ch** is located at the very bottom of the corresponding NCSC homepage. Since Linkedin.com had attracted attention a few years ago by the fact that more than 100 million passwords had been cribbed, the opening of a account is waived. Mr. Knoepfel gets in touch by mail a few days later. A security vulnerability had been submitted to NCSC, and the researchers would probably publish the vulnerability in the near future.

In the mail there is a reference to another mail in which it is claimed that the company Archivista received the security vulnerability on December 17, 2021. Since this cannot be verified with the logs or the spam folder, another call is made. Again the answering machine invites to leave a message.

NSCS phone number “only” at help.ch

A web search reveals the phone number of the NCSC at **help.ch.** When calling this number, a Ms. Minder answers. She is the communications officer of the Finance Department. When asked whether Mr. Knoepfel works at the NSCS, she answers that there is no Mr. Knoepfel known. Only after the objection that if there is no Mr. Knoepfel, it would require a report to the NCSC that someone is posing as an NSCS employee with the name Knoepfel, Ms. Minder asks how Mr. Knoepfel is spelled. Under ‘Knöpfel’ there is no one, but under Knoepfel, yes, he exists.

The number given by telephone matches the number in the e-mail. Again, an attempt is made to contact Mr. Knoepfel by telephone. Since this is unsuccessful, a longer e-mail is written describing the situation. Before the mail is sent, the phone rings. Mr. Knoepfel calls back and says that I have called several times.

The situation is described, according to which the company Archivista has not yet received a security report, although this has been requested several times

by e-mail. Mr. Knoepfel confirms that a notification has been received by the NCSC. Upon request, Mr. Knoepfel sent the security report by e-mail.
It is an eight-page PDF document that logs a reflected cross-siting script vulnerability (as described above) in the login screen. Also described quite well is how to avoid the execution of JavaScript via manipulated links, or to verify in our application that values are not processed correctly when logging in.

The “ominous” customer with version 5.2

A screenshot in the document in question shows Archivista WebClient with version 5.2. Version 5.2 dates from 2005 to 2007, so it is clear that the “mysterious” customer is not running an ArchivistaBox ever purchased from Archivista. Rather, an open source version is being used, which by now has somewhere between 15 and 20 years under its belt.
Of course, the GPLv2 license includes the right to use any software in any form. However, it is quite astonishing that a company can afford a security audit (such audits are not quite “cheap”) in order to have an approximately 15 to 20 year old software (in this case Archivista WebClient, currently WebDMS) audited. Now the company in question, which has obviously been using the ArchivistaBox without maintenance and support for some time, could tackle the problem themselves – after all, the sources are available. However, this did not happen. Instead, the audit company was commissioned to document the gap with classification “confidential”.
This led to the fact that the audit firm did not want to receive the document by mail or letter at any price. Nevertheless, after many detours via the **Swiss Federal Cyberattack Reporting Office,** the company Archivista GmbH finally gained insight into the problem — and was finally able to act.

The moral of the story…

Attacks with reflected cross-siting scripting are ultimately as effective as the users click on any links unthinkingly or enter their passwords unabashedly. Such gaps should and may be discussed. But they should not be overrated. Regardless of this, open source means openness based on trust and fairness. Where the sources are open, “secrecy” is of little or no use. Any vulnerabilities are publicly visible in the code anyway. And that’s why everyone is welcome to communicate any vulnerabilities at any time. In the spirit of openness, however, such reports will also be accepted in the future preferably by **mail.**