Zeus Panda Webinjects: Don’t trust your eyes

cyber.wtf/2017/03/13/zeus-panda-webinjects-dont-trust-your-eyes/

March 13, 2017

In our last blog article Zeus Panda Webinjects: a case study, we described the functionality of current Zeus Panda webinject stages and gave some insight into the corresponding administration panel. As we only scratched the surface of the target specific second webinject attack stage (in the following we reference this as 2nd attack stage), we would like to share more details about this part.

Basically, the 2nd attack stage already includes the complete code needed for the attack. The different code branches are triggered by setting status variables, especially the *branch* variable already introduced in our previous article on that topic. Last time we also introduced the *send()* function, which is used to exfiltrate data. *send()* isn’t entirely unidirectional: the HTTP response of this request includes further code that is evaluated as JavaScript. Thereby the backend is able to set the different status variables to trigger the existing code branches of the 2nd attack stage. Let’s dive into the details of this communication protocol:

Communication protocol and status variables

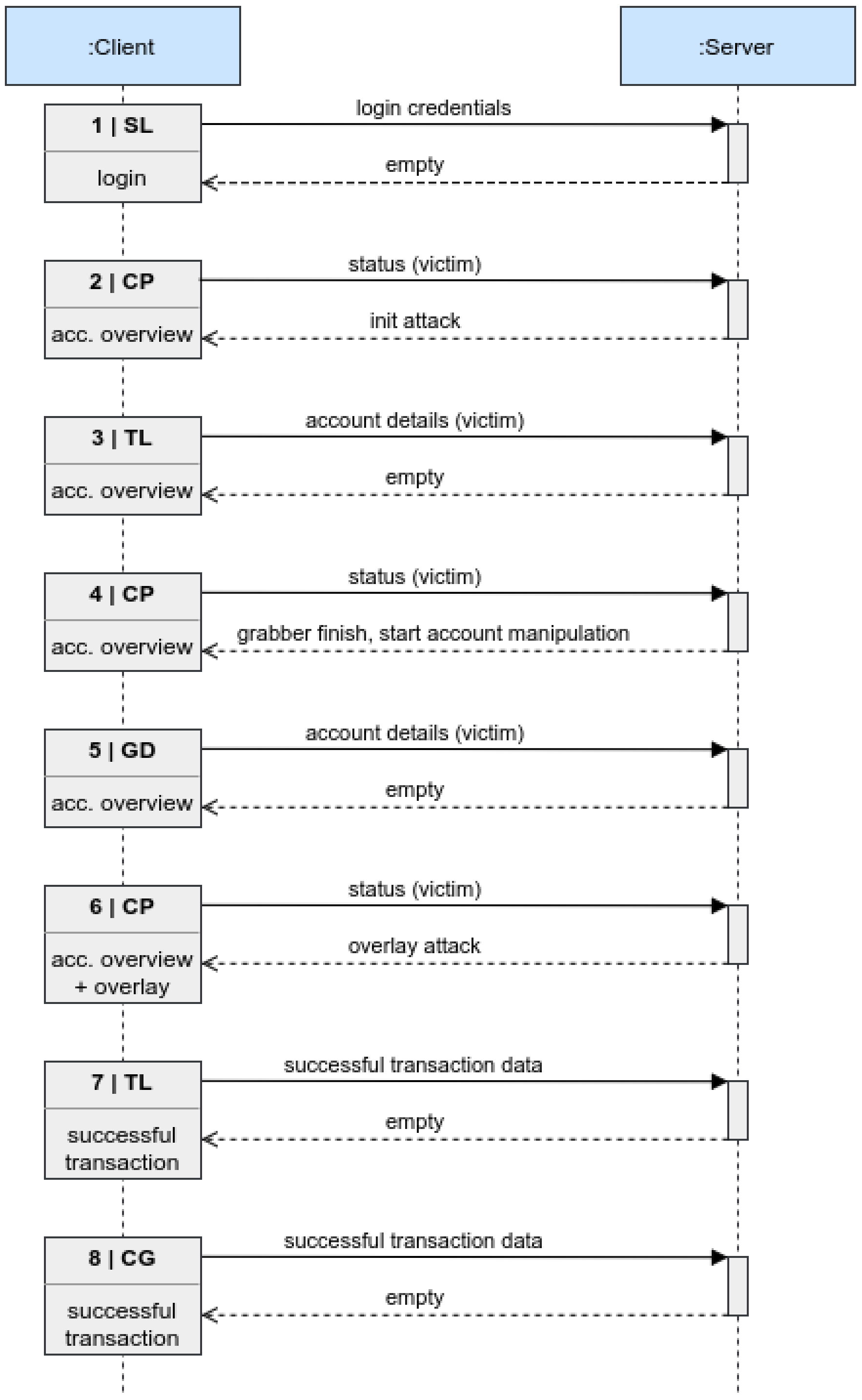


Figure 1: Communication

protocol

Figure 1 illustrates the communication protocol between the 2nd attack stage and the backend server. We see the different steps of the communication, the branches triggered, and the website on which the step occurs. Before going into details, the concept behind the communication is the following:

1. The current attack state is sent from the client to the backend server.
2. The backend checks for the current attack state and sets the right response parameters to initiate the next attack stage.

3. The backend response contains variables to notify the 2nd attack stage (client), which attack branch should be executed next.
4. The 2nd attack stage evaluates the response variables and triggers the next branch.
5. This procedure is repeated until the final state of the protocol is reached.

Time to branch

Let's take a detailed look into the different branches now.

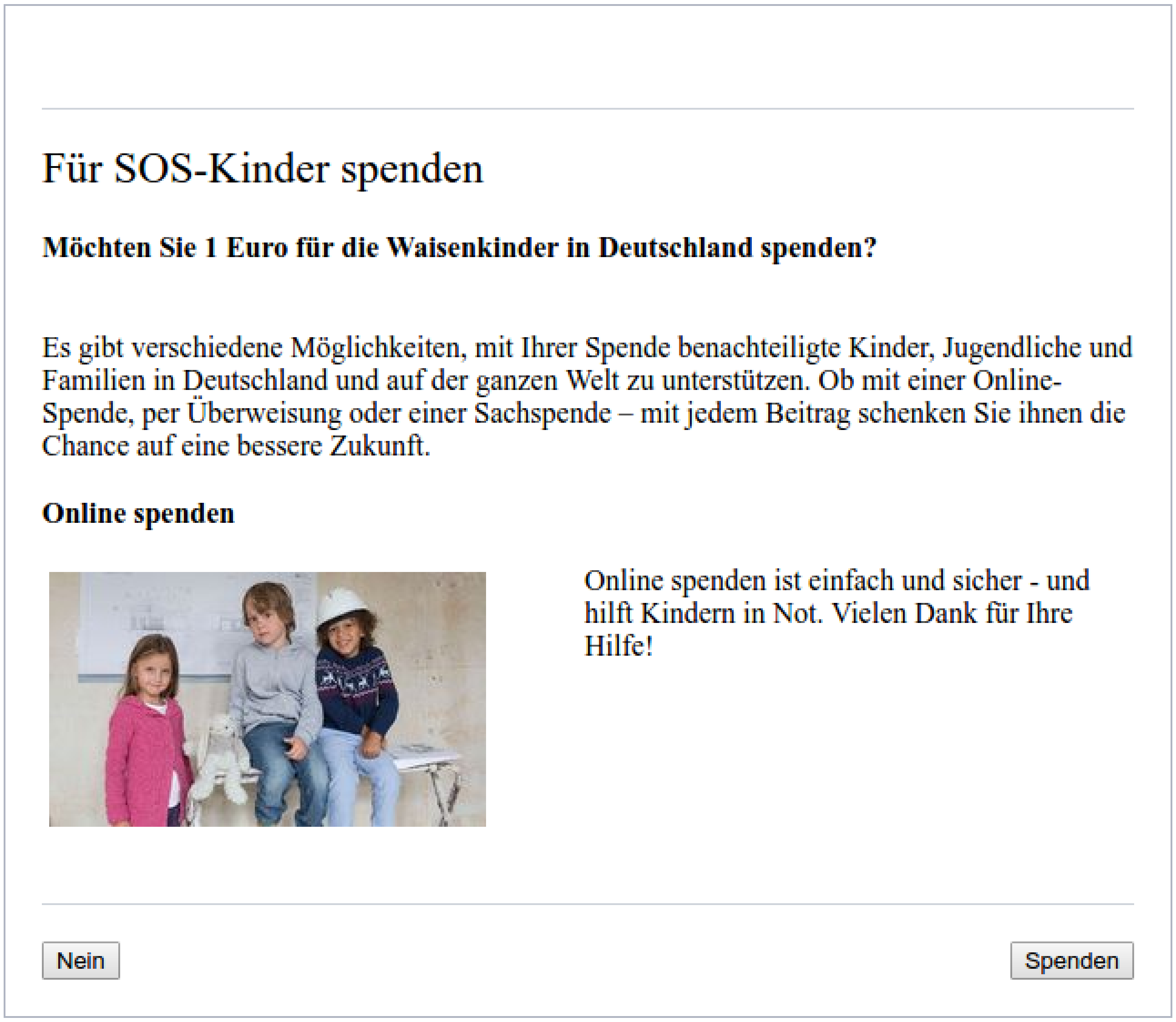
Step	Action
1	The *SL* branch is triggered at the beginning of the attack, when an infected victim accesses the login page of the targeted online banking, inserts the login credentials and clicks on the submit button. (NOTE: The low level Trojan functions need to trigger an the initial webinject (generic loader) on that website and therefore the URL of the online banking website has to be listed in the trojan config file). The submitted login credentials are intercepted, exfiltrated to the backend (see previous blog post), then the 2nd stage code calls the original login function of the banking or payment website. The backend now registers the new victim, identified by the botid. It returns an empty response to the webinject.
2	At this point, the victim has successfully logged in and has been redirected to the account balance overview page. This triggers the 2nd branch: *CP*. The *CP* branch is called multiple times during the attack and transmits general status information of the victim to the attacker. The response of the backend contains status flags to trigger the next step of the attack. At this point here, the backend signals to initiate the attack.
3	The attack signal triggers the 3rd step shown in Figure 1: The *TL* branch. This branch is used to collect details from all available accounts by using the grabber module. Furthermore, a flag is set to indicate a page reload after the response of the send function has been received. The collected data is then exfiltrated again. The botid is used to correlate transmitted data to existing victim entries in the backend and therefore works as unique identifier for the victim. The server response is empty, but the previously set reload flag now triggers the *CP* branch again.
4	The *CP* branch now sends the some information to the backend as described in Step 2. As the backend has stored a different state for the botid already, the response is different now. It signals the 2nd attack stage that the grabber module has finished and the ats module should start now. This module is used to manipulate account details like the account balance or transaction details. Also some status flags are set to trigger the next branch.
5	The *GD* branch: This branch is used to collect and exfiltrate account details of the victim. As already described in step 3, the reload flag is used to trigger the *CP* branch again.

Step	Action
6	The *CP* branch again submits status information, and the backend now triggers the next step of the attack. Besides some status flags, details about the target account and some fake data is provided. The data is used by the *CP* branch to display a fake overlay with a message and/or images, to trick the victim into starting a transaction. To that end, the fake overlay is used like in a normal phishing attack. We could observe different kinds of messages, which could be categorized into different modi operandi. (see below). If the victim fell for the scam, the previously provided data is used to pre-fill the transaction form. Naturally, this data contains a target account for the transaction. This account will be controlled by the attacker somehow, i.e., it most likely belongs to a money-mule. Additionally, the response from the backend contains fake information to be displayed. Depending on the modus operandi, this information is used to display different transaction details to the victim, then the ones used for the transaction in background.
7	Now the victim is redirected to the overview page for a successful transaction. In combination with the current flag state, this page visit triggers the *TL* branch of the 2nd stage code. The *TL* branch is used to collect details from the transaction overview page and exfiltrates them to the backend. This indicates a successful transaction to the attacker. The backend response is empty. The webinject transits into the next state, without the need for further communication with the backend.
8	The last triggered branch is called *CG*. It creates a copy of the complete DOM of the successful transaction overview page and exfiltrates it to the backend. There is no indication that this data is displayed in the admin panel, thus we assume it is transmitted for debug purposes only.

Modi Operandi

In the following we detail two different exemplary modi operandi, which we could observe during our analysis. The real visible appearance is different, as the webinject makes heavy use of the style-sheets provided by the target website. This is a very straight-forward way to properly brand fraudulent content to match the corporate design of target banks or payment providers. We focus on the content shipped to banking customers.

Charity Fraud: SOS-Kinder



Für SOS-Kinder spenden

Möchten Sie 1 Euro für die Waisenkinder in Deutschland spenden?

Es gibt verschiedene Möglichkeiten, mit Ihrer Spende benachteiligte Kinder, Jugendliche und Familien in Deutschland und auf der ganzen Welt zu unterstützen. Ob mit einer Online-Spende, per Überweisung oder einer Sachspende – mit jedem Beitrag schenken Sie ihnen die Chance auf eine bessere Zukunft.

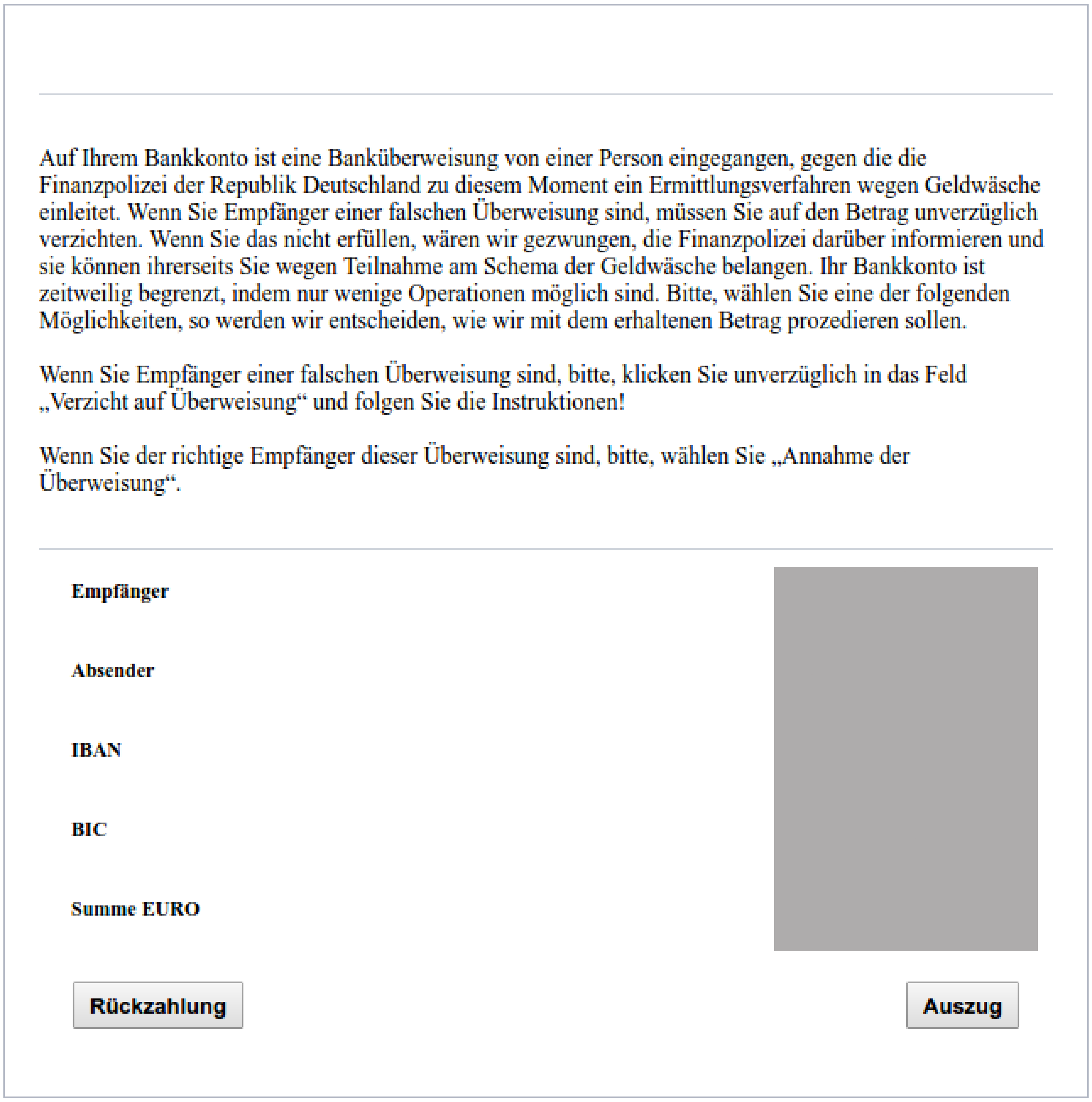
Online spenden

Online spenden ist einfach und sicher - und hilft Kindern in Not. Vielen Dank für Ihre Hilfe!

The victim is asked to donate 1€ to an non-profit organization, in this case for SOS children. This mimics the well know internationally active “SOS-Kinderdorf” organization. The German text is well written and does not contain the obvious indications for phishing that we all love and know from the occasional phishing mail, like contorted grammar and a more than flowery vocabulary. No Google Translate in sight, here. To leverage this scam vector, the webinject makes use of the data provided by the backed in Step 6 as detailed above. Using an overlay, the victim is made believe he/she is transferring 1€, but under the hood the amount is change to a much higher value.

The attackers follow a very classic social engineering approach for our part of the world and appeal to the victims helpfulness: Who doesn’t want to help children in need by spending 1€? We refer to this kind of attack as charity fraud.

Refund Fraud: Finanzpolizei



Auf Ihrem Bankkonto ist eine Banküberweisung von einer Person eingegangen, gegen die die Finanzpolizei der Republik Deutschland zu diesem Moment ein Ermittlungsverfahren wegen Geldwäsche einleitet. Wenn Sie Empfänger einer falschen Überweisung sind, müssen Sie auf den Betrag unverzüglich verzichten. Wenn Sie das nicht erfüllen, wären wir gezwungen, die Finanzpolizei darüber informieren und sie können ihrerseits Sie wegen Teilnahme am Schema der Geldwäsche belangen. Ihr Bankkonto ist zeitweilig begrenzt, indem nur wenige Operationen möglich sind. Bitte, wählen Sie eine der folgenden Möglichkeiten, so werden wir entscheiden, wie wir mit dem erhaltenen Betrag prozedieren sollen.

Wenn Sie Empfänger einer falschen Überweisung sind, bitte, klicken Sie unverzüglich in das Feld „Verzicht auf Überweisung" und folgen Sie die Instruktionen!

Wenn Sie der richtige Empfänger dieser Überweisung sind, bitte, wählen Sie „Annahme der Überweisung".

Empfänger

Absender

IBAN

BIC

Summe EURO

[redacted]

The overlay presents a message to the victim, indicating a transaction has been made to their account. As the victim sees a manipulated version of his account balance, he really believes the transaction has had happen. Furthermore the text indicates a preliminary investigation by the "Finanzpolizei" against the initiator of the transaction. If the victim is not transferring the money back, the text threatens with prosecution by law enforcement for participating in a money laundering scheme.

Finally, all the Google Translate and contextual cluelessness we came to love in the scams out there! Regrettably for the attacker, not all German-speaking countries are actually Germany. (We tried that once, partially, and it was a horrible idea.) An institution called "Finanzpolizei" does indeed exist — but not in Germany. The valid target audience for this scam is thus supposedly to be found in Austria, however, the scam is also actively used in Germany. The German text includes some mistakes and is not as well written as the first modus operandi we have shown above.

In the case at hand, the attackers try to make the victim follow through with a classic refund scam, by threatening legal consequences. As the story works without the need to manipulate the transferred amount under the hood, the fake data needed in the first described modus operandi is not used in this kind of attack. Nevertheless the attack is kind enough to prefill the transaction form with the correct details to ease the transaction for the victim.

Return of the victim

Now let’s assume the victim has been tricked into initiating a transaction by themselves to send their money to the attacker. What happens, if the victim takes a look into his online banking account some time later? As expected, the 2nd attack stage is also prepared for that case: The user is presented the “temporarily unavailable” notification (see Figures 1 and 2 from our previous post) and the login function of the target website is disabled. As long as the status variables are set to the finale state of the described communication protocol, the victim is thus unable to access their account again as long as the backend server is reachable. Even when disabling this blocking functionality, account information like transaction details and total balance are still manipulated. As this manipulations use the originally provided style cheets (CSS) from the target institute, a victim has no way to visibility distinguish between a fake entry and an original one.

Conclusion

Nowadays almost all financial institutes make use of two-factor authentication to protect their users from fraud. The modi operandi used by current banking trojan attacks successfully circumvent this by using social engineering techniques. The victim is tricked into initiating the transaction willingly and happily provides all information needed to confirm the transaction. This is achieved by visible modifications of the website that are indistinguishable from the original website content. The success rate of these attacks is still quite high.

By using a multi-layered attack, it’s also cumbersome for analysts to get an complete insight into the technical details. As soon as the backend server is not available anymore, only the 1st stage of a webinject is accessible on an infected machine. Without the backend server, most of the attack code is not available and therefore some pieces of the puzzle are missing.

These kind of multi-layered attacks have become more and more complex and sophisticated. However, beyond the visual appearance, the code of the original website is modified heavily to make this attacks work and these modifications necessarily leave a footprint. In our fraud detection solutions, we provide our customers with instant visibility into these modification symptoms so they can fare better at protecting their customers’ assets.

Authors: Manuel Körber-Bilgard and Karsten Tellmann