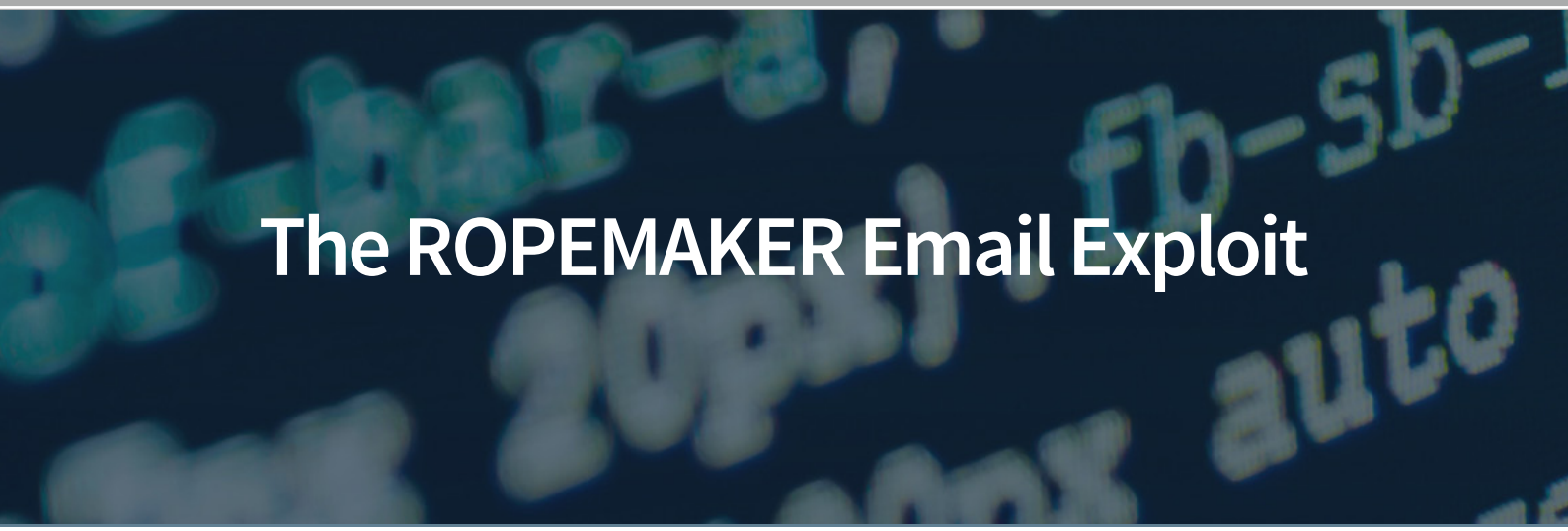
mimecast®

SECURITY ADVISORY



The ROPEMAKER Email Exploit

Do you think email is immutable once delivered? Do you think of email as being like a physical letter, that once dropped in a postal box cannot be changed? If you think these things, then you need to learn about an email-based attack technique that Mimecast has dubbed ROPEMAKER. What if a malicious actor, whenever they chose, could remotely change the content that you see in your email? Or even worse, what if a malicious actor could swap a benign URL with a malicious one in your delivered email or turn simple text into a malicious URL, without needing direct access to your PC or your email application? Do you want malicious outsiders to know when you read an email, what IP address you are on or what email client you are using? Furthermore, if you are using emails as business records, what if they could be changed post-delivery? If any of this concerns you, read on.

What is ROPEMAKER?

What is ROPEMAKER? The ROPEMAKER acronym itself stands for Remotely Originated Post-delivery Email Manipulation Attacks Keeping Email Risky[1]. What ROPEMAKER really is, is a type of email attack - discovered by Francisco Ribeiro (@blackthorne) at Mimecast - that could enable an attacker to remotely change the (perceived) content of an email, anytime, post-delivery. Does ROPEMAKER represent a vulnerability – which needs to be patched – or the exploit or misuse of an application which needs to be defended against using other means? Our hope is that this paper will help drive this discussion!

The origin of ROPEMAKER lies at the intersection of email and Web technologies, such as HTML, Cascading Style Sheets (CSS), and hypertext. While the use of these Web technologies has made email more visually attractive and dynamic relative to its purely text based predecessor, this has also introduced an exploitable attack vector for email. People commonly expect the content of Web pages to be dynamic - able to change moment-to-moment - but do not expect their email to do so as well. Email in many cases is treated more like a snail mail letter – once sent never changing - whereas Web pages are understood to be more like TV stations with a continuously changing flow of visual, audio, and text content. The techniques behind ROPEMAKER are thus another potential email-based attack vector that we expect attackers to leverage as they continually evolve from one technique to the next.

[1]Ropemaker Street in London "coincidently" also happens to be the street on which Mimecast has our European headquarters and where most our threat research team is based.

Bringing the dynamic capabilities of the Web to email

Fundamentally ROPEMAKER exists because Web technologies can and often do interoperate over a network, typically the Internet. To be more precise, two resources that are housed remotely from one another, but are linked via a network can interoperate; one affecting the execution of the other. In the Web content model, remotely based and controlled resources can be fetched or referenced without the direct control of the local user, it happens automatically, and in most cases is expected and desired functionality when it relates to web sites. A perfect example of this is the use of remote Cascading Style Sheets (CSS).

Adapted from the Wikipedia definition - A CSS is a style sheet language used to describe the presentation of a document written in a markup language, such as HTML. A CSS is a cornerstone technology used by most websites to create visually engaging webpages. ROPEMAKER takes advantage of the fact that a CSS enables the separation of presentation and content, including aspects such as the layout, colors, and fonts. Importantly, if supported by the presenting application such as the many email clients, a CSS file can be used locally with the markup language file or accessed remotely across the network (generally the Internet). And of course, the key of this exploit is from a security point of view, is that part of the system is controlled in an untrusted zone. And instead of controlling just the style of the email, as will be shown below, the remote CSS can actually control the content of the email.

Now imagine for a moment that you are a malicious actor or a cybercriminal with evil intent and you send an HTML-based email to your intended victim using a remote CSS that you host. But now imagine that a CSS can do more than just direct the presentation style of an HTML document, but can change the content that is presented to the user after the email has been delivered. ROPEMAKER works as long as the email client automatically connects to the remote CSS to retrieve the desired “style” for the email. This is at the core of the ROPEMAKER exploit.

How could ROPEMAKER be used in a cyberattack?

For example, an attacker could switch the display of an email from using a “good” URL to presenting an “evil” URL, just by changing the remote CSS that they control. Or the attacker could turn the presentation of regular email text into an “evil” URL for the user to click or copy into a browser directly. Or a malicious actor could change the “content” (the presentation of this content) of a delivered email thus impacting the integrity of a business record – changing “yes” to “no” or “$1” to “$1M”.

Two examples of ROPEMAKER being exploited using a remote CSS

Switch Exploit Using a Remote CSS

In the first example, which we call the "Switch", an email with a good URL later has its presentation switched by the malicious actor to the same message, but this time with a bad URL.

First let's look at the good email. Everything looks fine in Figure 1. But what if an attacker could change this email by editing the remote CSS from which the email gets its display "style" - Figure2.

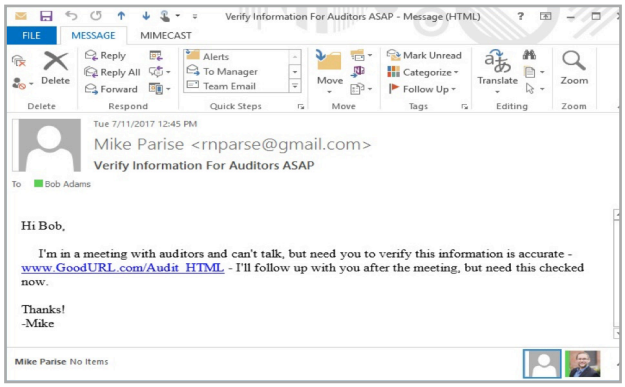


Figure 1 – Switch Exploit Email with a Good Link

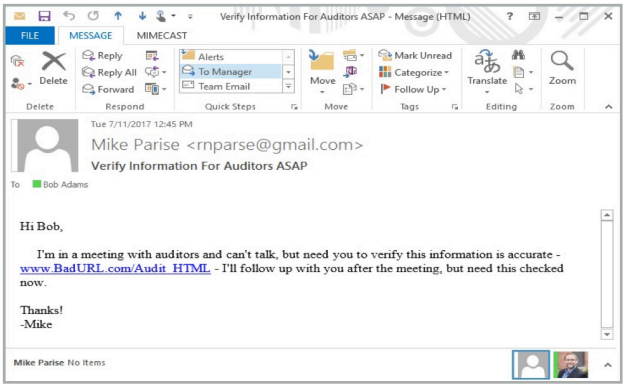


Figure 2 – Switch Exploit Email with a Bad Link

Of course, an attacker is unlikely to use a URL with "BadURL" text in it! Wouldn't that be nice!

The remote CSS code that switches the display to the evil URL for this example is in Figure 3.
This enables the attacker to toggle the display between the #safe and the #evil portions of the email.

And finally, the underlying HTML of the above emails with the associated call to the remote CSS is seen in Figure 4.

```
1  #evil { display: inline; }
2
3  #safe { display: none; }
```

Figure 3 – Primary Portion of the Remote CSS File

```
1  <head>
2  <link rel="stylesheet" type="text/css"
     href="http://evilsite/file.css">
3  </head>
4  <body>
5
6  <div id="evil">Hi Bob, I'm in a meeting with auditors and can't talk, but need you to verify this information is accurate -
     www.BadURL.com/Audit_HTML - I'll follow up with you after the meeting, but need this checked now. Thanks! -Mike
   </div>
7
8  <div id="safe">Hi Bob, I'm in a meeting with auditors and can't talk, but need you to verify this information is accurate -
     www.GoodURL.com/Audit_HTML - I'll follow up with you after the meeting, but need this checked now. Thanks! -Mike
   </div>
9
10 </body>
```

Figure 4 – The Content of the HTML Email with Both Bad and Good Versions of the Link

In the Switch Exploit example – which is purposely simple to introduce the ROPEMAKER concept - both of the URLs are sent in the original email and thus a security solution that rewrites URLs and inspects them on-click, such as Mimecast Targeted Threat Protection with URL Protect, would defend against this as both the "Good" and the "Bad" URL would be inspected before being resolved on-click. But of course, the post-delivery changing of the content could still confuse the recipient and make murky, in retrospect, what the message of the email was. And organizations without URL protections or sufficient Web security systems would be exposed to whatever threat is delivered by the bad URL.

Two examples of ROPEMAKER being exploited using remote CSS

Matrix Exploit

The potential exploit that we refer to as the "Matrix Exploit" is more sophisticated than the "Switch Exploit". With the Matrix Exploit a comprehensive matrix of ASCII text is sent by the attacker in an email, character-by-character in the form of a matrix of text. The attacker then uses the remote CSS to selectively control what is displayed, character-by-character and thus arbitrarily make it display whatever the attacker wants. For example, the attack could start off by displaying a blank email such as what is shown in Figure 5 below. And with a relatively simple change to the remote CSS file could change the display of the message to what is shown in Figure 6.

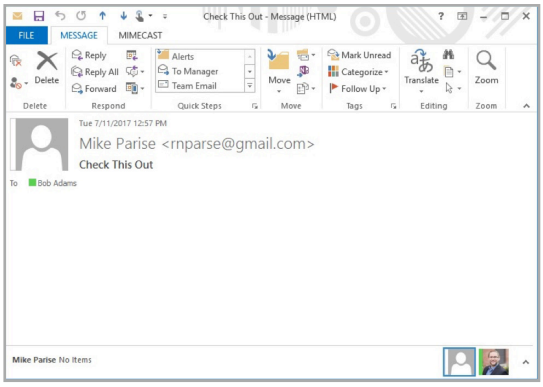


Figure 5 – Matrix Exploit "Before" Email

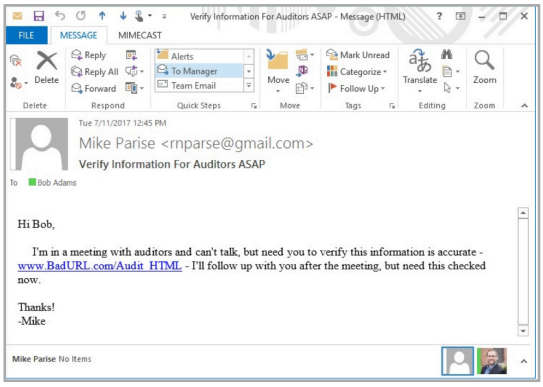


Figure 6 – Matrix Exploit "After" Email

The Matrix Exploit in many ways is harder to defend against because the email itself is just a blob of text with no URLs or other specific content yet apparent until post-delivery (although the relatively large number of HTML tags and the size of the message body could serve as a tip off). However, once the remote CSS file is changed to selectively display text and a URL, either the email client will present a clickable link (such as with Apple Mail) or in the absence of a clickable link will at least present text that resembles a URL and thus can be easily copied and pasted by an unsuspecting user. In this example, since the URL is rendered post-delivery, an email gateway solution such as Mimecast cannot find, rewrite, or inspect the destination site on-click, because at the time of delivery there would be no URL to detect. To do so would require the interpretation of CSS files, which is beyond the scope of current email security systems.

This is an area where the rendering software - usually email clients – should do more to protect the user. Why do they automatically load resources even if they are stored remotely? Or why do they depend on users to adjust email client security configurations or make decisions about something they often don't understand? In Microsoft Outlook on the desktop for example, it can be configured to warn before automatically downloading external resources. But how many users just dismiss the warning or disable the setting? From the users' point of view, If the original message looks fine, why not get the rest of the message?

Another compensating control that could address this type of exploit would be to use a Secure Web Gateway in proxy mode to inspect the destination site before allowing it to resolve to the user. To protect against the ROPEMAKER exploit Mimecast has added a feature which strips out references to external sources from all inbound emails.

How does the attacker pivot from Figure 5 to Figure 6? He could first start with the broad base of ASCII text character-by-character in the HTML email as shown in Figure 7 below.

Figure 7 – Broad Set of ASCII Text in the Email for the Matrix Exploit

Figure 8 – Text to Display to Generate the Bad URL for the Matrix Exploit

The attacker's goal of course is to avoid the target organization's security controls and to deliver a malicious email that will be acted on by the targeted person. The above ASCII text in an HTML email won't be caught by an email security system, as there is nothing inherently wrong with it. However, once the email is delivered, all the attacker needs to do is to selectively display the intended message by remotely causing the display of the bad text from the characters that are already resident in the delivered email. The highlighted HTML in Figure 8 highlights how to display the text that makes up the email that is in Figure 6 above.

Using the remote CSS file in Figure 9 the attacker can change the display of the email to that shown in Figure 6. It is important to note that the image in Figure 8 represents just one position in the matrix of the displayable characters and thus needs to be repeated for all the positions in the intended email message. So, one needs a relatively large email to present a relatively short message!

```
body {
    font-size: 0px;
}
div {
    font-size: 14px;
    margin: 0px;
    padding: 0px;
    display: none;
}
#offline_text {
    display:none;
}
/* message */
#upH0, #e1, #y2, #ws3, #c4, #h5, #e6, #c7, #k8, #ws9, #t10, #h11, #i12, #s13, #ws14, #o15, #u16, #t17, #ws18, #w19, #w20,
#w21, #dt22, #upB23, #a24, #d25, #upU26, #upR27, #upL28, #dt29, #c30, #o31, #m32 {
    display: inline;
}
```

Figure 9 – Remote CSS File for the Matrix Exploit

What are the negative implications of the ROPEMAKER exploit

Clearly, giving attackers remote control over any aspect of ones' applications or infrastructure is a bad thing. As was shown in the two examples above, this remote-control-ability could enable bad actors to direct unwitting users to malicious Web sites using a technique that could bypass both common security controls as well as fool even the most sophisticated users. ROPEMAKER could be leveraged in ways that are limited only by the creativity of the threat actors, which, experience tells us, is often unlimited.

There are other implications given the reality that supposedly immutable emails are in fact dynamic, post-delivery. For one, the integrity and thus the non-repudiation of emails is impacted. How can an email serve as a reliable business record if it can be "changed" at any time without the involvement of both parties? Also, what if an actor could send the "same" email to different recipients in an organization, but could have different content presented to each recipient or could modify specific participant's replies to others? Also, how can email archives be considered trustworthy given post-delivery dynamism?

ROPEMAKER weakness exploited in the wild

To date, Mimecast has not seen ROPEMAKER exploited in the wild. And this is certainly good news! Given that Mimecast currently serves more than 27K organizations and sees billions of emails monthly, if these types of exploits were being widely used it is very likely that Mimecast would see them. However, this is no guarantee that cybercriminals aren't currently taking advantage of ROPEMAKER in very targeted attacks and directing them to organizations that are not being served by Mimecast.

OTHER MODES OF EXPLOIT

While beyond the scope of this write-up, researcher Francisco Ribeiro has also shown that similar outcomes can be achieved using different methods, as follows:

- Man-in-the-middle attacks leveraging remote CSS. What if the attacker could get in the middle of a call to a legitimate remote CSS and substitute their own?
- Using external Scalable Vector Graphics (SVGs) with text and links to forge URLs or other content displayed to the end user.
- Using <embed> and <iframe> tags as alternative techniques to bring in a document, application, or interactive content into an email.
- Using dynamically generated remote fonts where non-required characters are used as slots that point to the desired character representation for a given position.

PGP, S/MIME, and DKIM and ROPEMAKER?

Does using PGP, S/MIME, or DKIM security technologies help against ROPEMAKER? No. These technologies only protect against the manipulation of emails using cryptology while an email is in transit or at rest. To use ROPEMAKER there is no need to manipulate the content of an email while it is in transit or at rest, as the actual content of the email is not changed, just what is displayed to the user is changed via remote control.

How to defend against the ROPEMAKER exploit

To address this risk for those organizations that are not customers of Mimecast, one approach which unfortunately is quite draconian, would be to disable the use of HTML mail and only allow plain text emails throughout the organization.

To our knowledge, a good workaround to consider is the adoption of web clients (such as Gmail, Outlook.com, and icloud.com) which are not affected by these types of exploits and yet do support the presentation of HTML emails – as opposed to just text-based ones.

For use by Mimecast customers, Mimecast has added "Strip external source mode" to the configuration of the Mimecast Targeted Threat Protection URL Protect service (part of the broader cloud-based email security gateway service provided by Mimecast), as shown in Figure 10. With this setting "on" the Mimecast service will inspect inbound emails for references to external resources including CSS, font-types, SVG files and the tags <embed>, <iframe>, <frame>, and <object>, and strip this content from the email before delivery. Enabling this setting will protect organizations against the ROPEMAKER exploits. However, legitimate use of these Web technologies in an email would also be disabled, thus potentially impacting the intended user experience with the email.

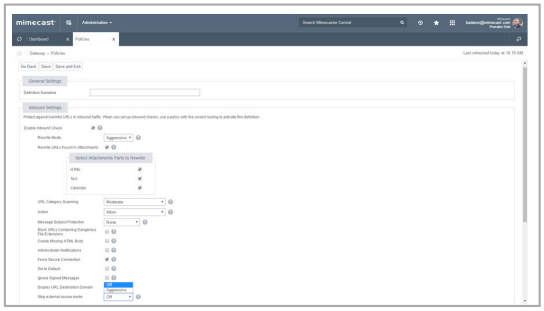


Figure 10 – Mimecast URL Protect Configuration Screen Capture

MIMECAST TESTING OF THE ROPEMAKER WEAKNESS

Mimecast tested the most recent versions of the following email clients as part of our validation of ROPEMAKER. It is our expectation that previous, current, and future versions of these email clients will remain susceptible to the ROPEMAKER type of exploit until the related exploit methods are more broadly understood and accepted.

- Microsoft Outlook both desktop and mobile
- Apple Mail both desktop and mobile.
- Mozilla Thunderbird

Interestingly the version of Android mail client that was tested was negative for this mode of exploit. However, given the numerous versions of Android that exist on the market, no definitive statement can be made about its ability to protect against these types of exploits. Common browser-based email clients such as Gmail, Outlook.com, and icloud.com were found not to be susceptible to the ROPEMAKER exploit.

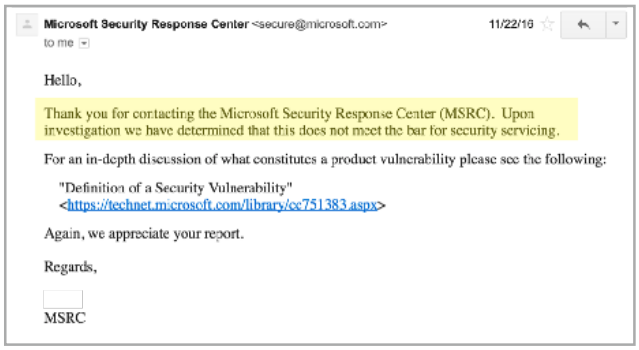
ROPEMAKER Disclosures

Have the ROPEMAKER exploit techniques been disclosed to the primary email client vendors? Yes. Mimecast first notified the market's primary email client vendors, notably Apple and Microsoft, in late 2016. Mimecast presented further details of ROPEMAKER at a private email security technical forum in early 2017. As of the date of this writing there has not been a general acceptance of ROPEMAKER as a vulnerability or a form of potential application exploit by any impacted email client application owner. Mimecast is now sharing the details of ROPEMAKER publicly to improve security awareness and encourage appropriate action to address the risks that we believe exist with email. The defensive threat research community needs to continue to disclose potential new exploits such as these in a timely and responsible manner.

Mimecast has also filed a request for a CVE with the MITRE Corporation. Recently we heard back from MITRE that they will not issue a CVE number primarily because the application vendors concerned do not consider ROPEMAKER to be a vulnerability. In our opinion ROPEMAKER should be classified by MITRE as an Inclusion of Functionality from Untrusted Control Sphere.

Microsoft's Response

To date the only response we have received from Microsoft characterized the ROPEMAKER style of exploit as not a vulnerability.


Microsoft Security Response Center <secure@microsoft.com> 11/22/16
to me

Hello,

Thank you for contacting the Microsoft Security Response Center (MSRC). Upon investigation we have determined that this does not meet the bar for security servicing.

For an in-depth discussion of what constitutes a product vulnerability please see the following:

"Definition of a Security Vulnerability"
<https://technet.microsoft.com/library/cc751383.aspx>

Again, we appreciate your report.

Regards,

MSRC

Figure 11 – Microsoft's response to the disclosure of the ROPEMAKER exploit

Apple's Response

Very recently Mimecast did receive the following response from Apple in response to a draft version of this paper:

Users can choose to disable the loading of remote content by navigating to Mail > Preferences > Viewing and unchecking "Load remote content in messages"

While this control is comparable to the "Strip external source mode" filter provided by Mimecast, the fact that it is provided at the client level (unlike Mimecast which is provided at the Gateway and thus can be easily applied to all email users by administrators) and thus is under the control of each individual user, adds implementation risk and complexity. Do users really understand this area of potential weakness? Would they tend to leave this setting "on" once they receive a poorly displayed legitimate email that depends on a remote resource? More defenses can likely be put in place on the client side. In addition iOS doesn't have this same configuration capability.

[2] Common Weakness Enumeration, Weakness ID: 829, Inclusion of Functionality from Untrusted Control Sphere
[3] Email from Apple - Public Disclosure of Previously Submitted Vulnerability; Follow-up: 669337226

Conclusion

Clearly, potentially giving attackers remote control over any aspect of ones' applications or infrastructure is a bad thing. As was shown in the two examples above, this remote-control-ability could enable bad actors to direct unwitting users to malicious Web sites using a technique that could bypass both common security controls as well as fool even the most sophisticated users. ROPEMAKER could be leveraged in ways that are limited only by the creativity of the threat actors, which, experience tells us, is often unlimited.

Is ROPEMAKER a software vulnerability, a form of potential application abuse/exploit, or a fundamental design flaw resulting from the intersection of Web technologies and email? Does it really matter which it is? For sure attackers don't care why a system can be exploited, only that it can be. If you agree that the potential of an email being changeable post-delivery under the control of a malicious actor increases the probability of successful email-borne attack and that attackers will evolve as necessary to achieve their malicious goals, the issue simplifies itself. Experience tells us that cybercriminals are always looking for the next attack technique to use. As an industry let's put our heads together to reduce the likelihood that the ROPEMAKER style of exploits gains any traction with cybercriminals!