UNDERSTANDING PHISHING

INKY

Text Direction Deception

In this installment of Understanding Phishing we'll dissect this email to understand how it works. We'll also take a look at the countermeasures we've deployed to catch scams that use these CSS tricks.

Scamming with Sneaky CSS

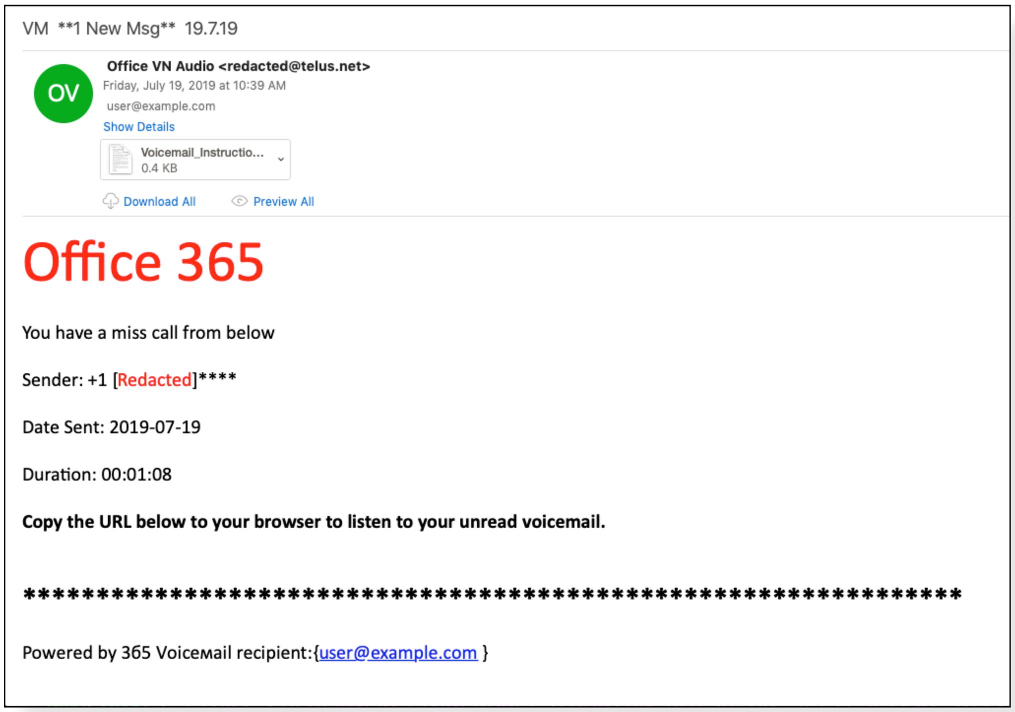
Every so often we at INKY come across a scam email that genuinely impresses us with its injurious ingenuity. One such message arrived in one of our customers' inboxes in July 2019.

What's so interesting about this malignant mail is that it uses capabilities of Cascading Style Sheets (CSS) that most people don't even know exist to sneak through incumbent mail protection systems like Proofpoint.

In this installment of *Understanding Phishing* we'll dissect this email to understand how it works. We'll also take a look at the countermeasures we've deployed to catch scams that use these CSS tricks.

Our primary finding is that CSS offers tools for mixing scripts like Arabic and Latin, which naturally flow in different directions on the page. Attackers can abuse this feature to make backwards text render forwards in an email, thereby hiding the text from the Secure Email Gateway (SEG) while preserving the normal appearance to the human recipient.

Here's what that July 2019 mail looks like (edited only to redact PII):



VM **1 New Msg** 19.7.19

OV

Office VN Audio <redacted@telus.net>
Friday, July 19, 2019 at 10:39 AM
user@example.com
Show Details

Voicemail_Instructio...
0.4 KB

Download All   Preview All

Office 365

You have a miss call from below

Sender: +1 [Redacted]****

Date Sent: 2019-07-19

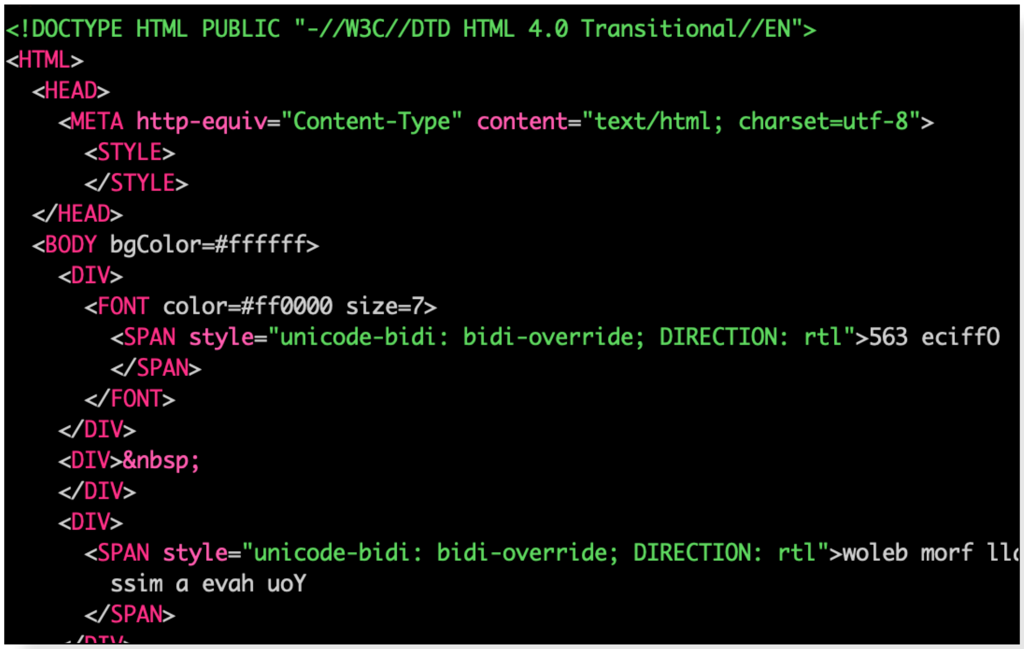
Duration: 00:01:08

Copy the URL below to your browser to listen to your unread voicemail.

****

Powered by 365 Voicemail recipient:{user@example.com }

At first this looks like a typical fake voicemail notification. We see many variants of these, usually (but not always) impersonating Microsoft. As is fairly common, the Office 365 logo in the upper left corner is actually just red text set in large type — the attributes `color=#ff0000 size=7` effect this “logotype” style:

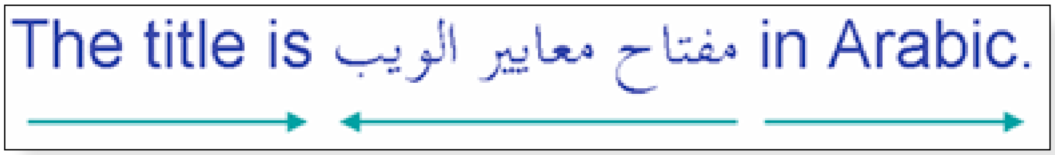


```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
  <HEAD>
    <META http-equiv="Content-Type" content="text/html; charset=utf-8">
      <STYLE>
      </STYLE>
  </HEAD>
  <BODY bgColor=#ffffff>
    <DIV>
      <FONT color=#ff0000 size=7>
        <SPAN style="unicode-bidi: bidi-override; DIRECTION: rtl">563 eciffO
        </SPAN>
      </FONT>
    </DIV>
    <DIV>&nbsp;
    </DIV>
    <DIV>
      <SPAN style="unicode-bidi: bidi-override; DIRECTION: rtl">woleb morf llo
        ssim a evah uoY
      </SPAN>
```

But wait! Where's the text saying Office 365? If you look closely, it says "563 eciffO" instead of "Office 365". And notice the style attribute on the span element: **`unicode-bidi: bidi-override; DIRECTION: rtl`**. Mozilla describes these obscure CSS properties as follows:

The unicode-bidi CSS property, together with the direction property, determines how bidirectional text in a document is handled. For example, if a block of content contains both left-to-right and right-to-left text, the user-agent uses a complex Unicode algorithm to decide how to display the text. The unicode-bidi property overrides this algorithm and allows the developer to control the text embedding.

To make sense of this, we need to first recognize that an HTML document might contain multilingual text, where some text is in a left-to-right script like Latin and some is in a right-to-left script like Arabic. Here's an example from the W3C site:



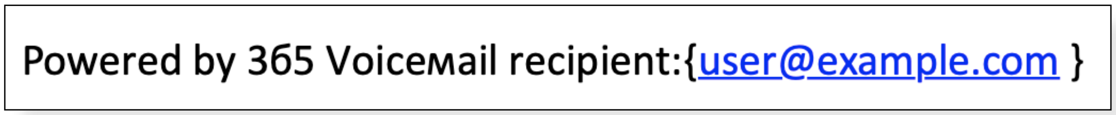
In most cases the HTML rendering engine is smart enough to automatically figure out how to display mixed script sequences like this properly. But in tricky cases the web designer may need to provide explicit hints to tell the renderer exactly which directionality to use for which text span. The scammers are using this hinting mechanism to force the backwards text "563 eciffO" to render right-to-left — this is the meaning of the **`rtl`** value of the **`direction`** property — and appear to the end user as "Office 365". But why?

The answer is: to trick the Secure Email Gateway (SEG). The attacker knows SEGs use Bayesian statistical models that learn which text sequences distinguish good/legitimate mail from bad/spammy/scammy mail. These models — the workhorse of mail protection since the 90s — learn, for example, that dollar signs in the subject line and "make money fast!" in the body correlate with spam.

These models also learn to detect *scam-indicative* patterns like "Office 365 ... voicemail". While the presence of a pattern like this obviously doesn't *guarantee* a mail is bad, it may trigger the SEG to do deeper analysis of the mail.

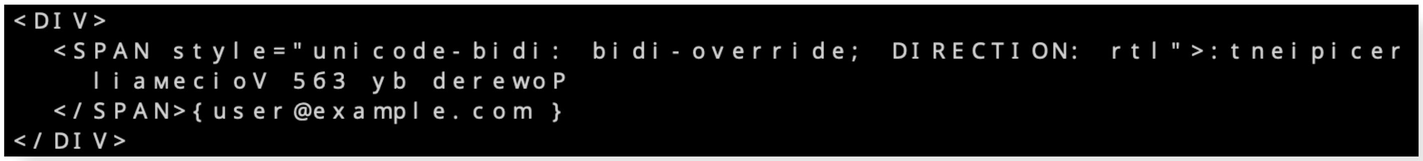
The CSS "direction trick" allows the attacker to thwart the SEG's pattern matcher, because the indicative text *never actually appears in the HTML code of the email.* But this concealment remains invisible to the end user, who sees the text as perfectly normal. Thus the attacker succeeds in fooling both the SEG and the end user... simultaneously!

Interestingly, this email also exploits another HTML trick we cover in an earlier installment of the *Understanding Phishing* series. Look carefully at this text:



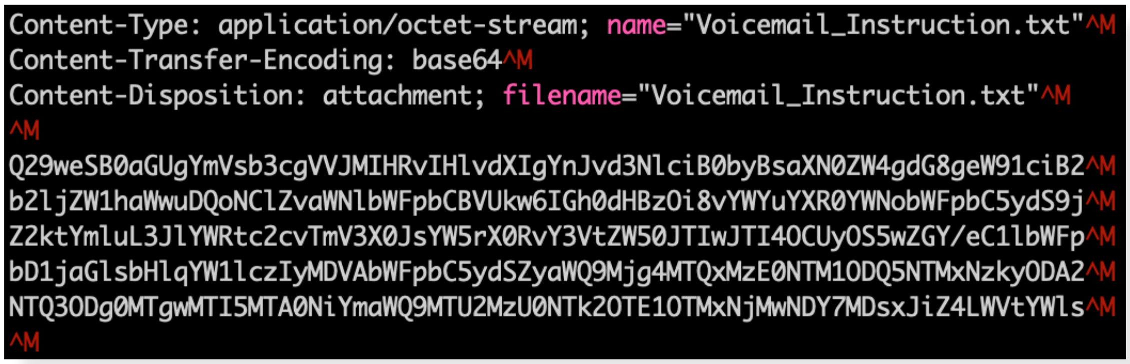
Powered by 3б5 Voiceмail recipient:{user@example.com }

Notice how the 6 in 365 and the m in Voicemail look funny? That's because those glyphs are actually Unicode CYRILLIC SMALL LETTER BE and Unicode CYRILLIC SMALL LETTER EM respectively. This entire text appears reversed in the HTML, too:



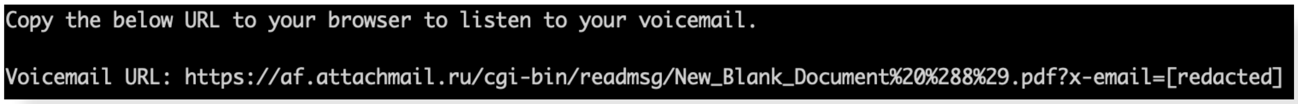
```
<DIV>
  <SPAN style="unicode-bidi: bidi-override; DIRECTION: rtl">:tneipicer
    liamecioV 563 yb derewoP
  </SPAN>{user@example.com}
</DIV>
```

A double deception! And as if all this weren't enough to confuse the SEG, this email hides its phishing link — the bait the attacker hopes the victim will take — inside a text file attachment:



```
Content-Type: application/octet-stream; name="Voicemail_Instruction.txt"^M
Content-Transfer-Encoding: base64^M
Content-Disposition: attachment; filename="Voicemail_Instruction.txt"^M
^M
Q29weSB0aGUgYmVsb3cgVVJMIHRvIHlvdXIgYnJvd3NlciB0byBsaXN0ZW4gdG8geW91ciB2^M
b2ljZW1haWwuDQoNClZvaWNlbWFpbCBVUkw6IGh0dHBzOi8vYWYuYXR0YWNobWFpbC5ydS9j^M
Z2ktYmluL3JlYWRtc2cvTmV3X0JsYW5rX0RvY3VtZW50JTIwJTI4OCUyOS5wZGY/eC1lbWFp^M
bD1jaGlsbHlqYW1lczIyMDVAbWFpbC5ydSZyaWQ9Mjg4MTQxMzE0NTM1ODQ5NTMxNzkyODA2^M
NTQ3ODg0MTgwMTI5MTA0NiYmaWQ9MTU2MzU0NTk2OTE1OTMxNjMwNDY7MDsxJiZ4LWVtYWls^M
^M
```

The rectangular block of gibberish is base64-encoded data that decodes as this:



```
Copy the below URL to your browser to listen to your voicemail.

Voicemail URL: https://af.attachmail.ru/cgi-bin/readmsg/New_Blank_Document%20%288%29.pdf?x-email=[redacted]
```

In one final flourish, the scammer has also set the **`Content-Type`** of this attachment to **`application/octet-stream`**. This media type is only meant for an opaque byte stream — data in an unknown format – but the attacker knows mail readers like Outlook and Gmail will automatically auto-detect the true file type and display the attachment properly as a plain text file — even though attachments of this "unknown" type are likely ignored by the SEG!

How INKY Solves it

There’s so much going on in this email that one really has to marvel at the sender’s mastery of malign mail. To fight back, INKY must be clever too.

To identify emails like this, INKY first renders the HTML content of each email. In essence, INKY pretends to be a mail reader like Outlook that’s about to display the mail to the recipient end user. This all happens in the cloud, so there’s no physical display to render to and no actual user — but the process mimics *what will happen* when Outlook displays the mail.

This virtual rendering step gives INKY two parallel views of each email: the raw HTML *elements, attributes, and properties* view, and the visual *what the user will actually see* view. By comparing and contrasting these two views, INKY can readily spot directional deceptions like this. Here INKY sees “Office 365” both in its reversed form (in the raw HTML) and in its normal form (in the rendered output). So INKY can see brand-indicative text here where other systems can’t. In this case INKY ses that this mail appears to be from Microsoft, but is sent from **`telus.net`** which is not a legitimate sending mail server for Microsoft. It is therefore a brand impersonation which should be quarantined.

INKY also simulates the mail readers’ behavior with attachments. Like Outlook, INKY automatically determines file types regardless of the opaque **`Content-Type`**, and is able to extract the URL from the attachment to this email. If a URL in an attachment has been reported as phishing, INKY can quarantine the message on that basis alone. In general, INKY is able to extract URLs from virtually any kind of attachment.

INKY®

Conclusion

Phishing scams continue to evolve, and attackers apply an ever growing bag of tricks to fool both human recipients and mail protection software. Sophisticated bad actors bring extensive knowledge of how SEGs and mail readers work to craft scams that fool both the humans and the machines.

INKY renders HTML email just like a real mail client does, and then uses computer vision techniques to "see" the result much like a person would. This allows INKY to see through these kinds of scams and prevent more phishing emails from reaching users' inboxes.