Anti-Espoofer: An Approach to Enhance Email Spoofing Identification and Prevention

Pingfan Xu
pingfan@uoguelph.ca
University of Guelph
Guelph, Ontario, Canada

Yiwei Guo
yguo27@uoguelph.ca
University of Guelph
Guelph, Ontario, Canada

ABSTRACT

Email is one of the most important and frequently used means of daily communication. With the convenience of email communication, email spoofing raises concerns for communication security when using email. The major email service vendors have already employed authentication checking mechanisms like DKIM, SPF, and DMARC to prevent email spoofing attacks. However, according to a recent publication at the 29th USENIX Security Symposium, the implement inconsistency among these three checking techniques would leave possibilities for email spoofing attacks. In this paper, we conduct studies on the issues mentioned in the publishing and give out our countermeasures toward them. We try all the mentioned attacks toward five major email service providers as an experiment. With the experimental result, we find some issues have already been fixed for particular vendors. As one of our experimental targets, Yahoo Mail has the worst prevention performance against the tested attacking cases among the five targets. Based on this finding, we implement a tool called Anti-Espoofer, which works as a complement of Yahoo Mail's existing authentication checking mechanisms. We analyze the spoofing emails passed Yahoo's checking and explain our countermeasures' working principle in detail. Due to the time limitation, Anti-Espoofer is still on a primitive version. At the end of this paper, we also provide a possible direction to make Anti-Espoofer a mature product as a browser extension.

CCS CONCEPTS

• **Security and privacy** → *Web application security.*

KEYWORDS

email spoofing, email authentication, DKIM, SPF, DMARC, anti-spoofing

ACM Reference Format:
Pingfan Xu and Yiwei Guo. 2018. Anti-Espoofer: An Approach to Enhance Email Spoofing Identification and Prevention. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

1.1 Background

With the rapid development of the Internet, technology is widely used in all industries. Regardless of the industry, email, as one of the most commonly applied technologies, is an integral part of the modern work environment. Even for people's daily lives, email is an important means of communication. Emails provide the possibilities to create effective one-to-one or one-to-many communication channels. Because of this email capability, it is trendy for companies to keep in touch with their clients or customers. Companies regularly send promotional emails to their customers for marketing purposes. While people and companies enjoy the convenience of email, the public has been questioning and challenging the security of email itself. It is not hard to find news about spam emails and even email spoofing. For example, it was reported that spear-phishing attacks through email spoofing become the biggest scam in Canada, with about $37 million losses in the year 2019 [3]. Since email spoofing is common and often causes severe losses, more and more people pay attention to it and work on strategies that fight against it.

1.2 Employed Modern Countermeasure

Given the prevalence and severity of email spoofing, email service providers have made significant efforts to curb email spoofing in recent years. Simple Mail Transfer Protocol (SMTP), the current email transmission protocol, is unfortunately not having built-in mechanisms for effectively fighting against spoofing [8]. Extended anti-spoofing mechanisms for SMTP starts to be developed and employed in modern email service provides. There are three of them that are now built-in in most of the email service providers. Taking inspiration from asymmetric encryption, DomainKeys Identified Mail (DKIM) had been introduced. Meanwhile, the Sender Policy Framework (SPF) has been created by utilizing the usage of the Domain Name System (DNS). As the last step of the anti-spoofing process, as well as the complement of DKIM and SPF, Domain-based Message Authentication, Reporting, and Conformance (DMARC) has been developed to cover the deficiencies of DKIM and SPF. Modern email service providers usually apply DKIM, SPF, and DMARC together to prevent spoofing and protect their clients. In the following section, these three mechanisms are explained in detail on how they work together and what are their stand-alone working principles and vulnerabilities.

1.2.1 DomainKeys Identified Mail (DKIM). The establishment of DomainKeys Identified Mail (DKIM) was inspired by the concept of asymmetric encryption and decryption. To apply DKIM, the sender's email service provider is required to include a digital signature signed by the private key of the sender's domain. The digital signature is placed in the header of the original email. When the email service provider of the recipient processes the received email, it queries the DNS to get the public key of the sender and verify the email with the result public key [5]. To fetch the corresponding DKIM public key from the DNS, the recipient's email service provider needs to query with the sender's domain and a DKIM selector, which is also included in the header of the original email. Only when the digital signature successfully passes the verification, the recipient's email service provider will consider the email is not spoofed from a DKIM point of view.

Even though DKIM ensures that only the individual who knows the private key can send the email on behalf of the corresponding domain, it is still easy to bypass DKIM authentication. The problem with DKIM is that it does not validate the From field in the header, which is displayed to the end-user. This problem leads to a possibility for the attacker to let the end-user see the spoofed email address, but the DKIM authentication is passed.

1.2.2 Sender Policy Framework (SPF). In addition to DKIM, Sender Policy Framework (SPF) is another anti-spoofing mechanism. SPF basically verifies the IP address of the sending domain. The prerequisite of properly using SPF requires the domain owner to publish an authorized IP list via DNS. Typically, the authorized IP list is published either as a specific SPF record or a TXT (text) [4]. The email service provider of the recipient can then query the domain in HELO and MAIL FROM to obtain the authorized IP list. With the obtained list, if the sender's IP address is included in the obtained IP list, the received email passes the SPF checking.

Similar to DKIM authentication, using SPF checking alone may not effectively mitigate the risk of email spoofing. The problem of the SPF is that it does not check the From field in the header, which is shown to the end-user. As long as the attacker find a proper way to bypass the SPF checking process, the spoofed email address will still be displayed to the end-user.

1.2.3 Domain-based Message Authentication, Reporting and Conformance (DMARC) . Domain-based Message Authentication, Reporting, and Conformance (DMARC) is designed to fix DKIM and SPF's problem that neither one of them check the From field included in the header of the received email. When the sending service deploys the DMARC mechanism, the recipient's email service provider must perform an identifiable alignment test if the domain of the From field in the header of the received email matches the DKIM-verified or SPF-verified domain. There are two different modes for domain matching check: the strict, exact match and the default relaxed match of having the same registered domain [6]. Another essential point is that DMARC only requires either DKIM or SPF to show a positive result. It means that if a received email passed either DKIM or SPF and the domain of the From field in the header passes the alignment test of DMARC, the email would be considered as valid with respect to DMARC.

Since DKIM or SPF validates the sender's information in the header and DMARC tests the alignment of the validated header and the displayed From domain, having DKIM, SPF, and DMARC working together seems to be a perfect solution for fighting against email spoofing attacks. However, the seemingly robust inspection mechanism is still vulnerable under particular circumstances. In the next section, a research paper about bypassing the DKIM, SPF, and DMARC combination in modern email service providers will be briefly reviewed.

1.3 Conference Paper

Composition Kills: A Case Study of Email Sender Authentication is a research paper written by Jianjun Chen[2] and his colleagues recently published at the 29th USENIX Security Symposium. This paper is mainly focused on researching the implementation inconsistencies between the mail servers and mail clients as well as bypassing the email authentication, which includes impersonating arbitrary senders, forge DKIM signature with legitimate domains' signature. As a result, they have discovered 18 kinds of attacks by manual analysis and black-box testing on several mail servers and mail clients.

1.3.1 Non-existent subdomains (A1). The attackers can craft a non-existent subdomain of a legitimate domain of the mail from in SMTP. In this way, they could bypass the SPF. It checks the HELO domain and the DMARC because it tests the domain of the mail from. (e.g. "(any@notexist.legitimate.com") (Discuss this attack in detail in 3.3.1)

1.3.2 "Empty" MAIL FROM addresses (A2). The attackers can add a left parenthesis "(" before the forged legitimate address of the mail from in SMTP. This approach will pass the authentication checks because the SPF (treat as empty) and the DMARC (treat as usual) will get different parsing results for different mail from addresses in SMTP. (e.g. "(any@legitimate.com")

1.3.3 NUL ambiguity (A3). The attackers can construct the DKIM-Signature header with an "\x00" after their domain. This operation can pass the DKIM verification because it considers "\x00" as the terminator. The DMARC also pass "because the 'd= 'domain is identical to the from header domain." (e.g. "attack.com.\x00.any._domainkey.legitimate.com")

1.3.4 DKIM authentication results injection (A4). The attackers can generate DKIM-Signature headers embedded in a literal open parenthesis "(." Both the DKIM verification (verify the domain behind the parenthesis) and the DMARC alignment test (take the domain before the parenthesis) will show positive results because of their parsing difference. (e.g. "legitimate.com(.attacker.com") (Discuss this attack in detail in 3.3.2)

1.3.5 SPF authentication results injection (A5). Similar to the A4 attack, the attackers can craft the mail from in SMTP with the open parenthesis "(." All the tests will pass since the SPF checks the second half and the DMARC tests the first half. (e.g. "legitimate.com(.attacker.com") (Discuss this attack in detail in 3.3.3)

1.3.6 Multiple From headers (A6). The attackers can create multiple from headers. Some email servers will take the first header to do the DMARC alignment test but display the last header, which will mislead the users.

1.3.7 Space-surrounded From headers (A7). The attackers can insert the whitespace before or after the header name. Due to the different implementations, the whitespace may bypass the validation, and it will also cause the display problem described in A6.

1.3.8 From alternative headers (A8). The attackers can create a sender header that includes the forged legitimate address. The DMARC alignment test will pass because it queries the domain in the from header, but the mail user agents will display the address in the sender header.

1.3.9 Multiple email addresses (A9). The attackers can add multiple email addresses in the from header. Some email servers will take the first email address to do the authentication, while their web interfaces only show the last one. (e.g. From: <any@attack.com>, <admin@legitimate.com>) (Discuss this attack in detail in 3.3.4)

1.3.10 Email address encoding (A10). The attackers can encode the forged legitimate email address in base64 and also add the attackers' email address in the from header. Since the DMARC alignment test cannot recognize the encoded domain, it will test with the attackers' domain, which will show positive. (e.g. From: bs64(<admin@legitimate.com>), <any@attack.com>) (Discuss this attack in detail in 3.3.5)

1.3.11 Route portion (A11). The attackers can construct a route portion in the from header. Some mail servers cannot recognize the route portion, so they will use the domain in the route portion to do the DMARC alignment test. But when the mail clients present the email, it will ignore the route portion and only show the rest part. (e.g. From: <@attack.com, @any.com: admin@legitimate.com>) (Discuss this attack in detail in 3.3.5)

1.3.12 Quoted-pairs (A12). The attackers can insert a "\" after an email address. Due to the different implementations of email servers and email clients, some servers and clients only recognize the email address before the "\" while others validate and display the after part. (e.g. From: <admin@legitimate.com>\, <any@attack.com>) (Discuss this attack in detail in 3.3.6)

1.3.13 Parsing inconsistencies (A13). The attackers can delete the "<>" of an email address. Some email servers believe "<" has higher priority so that the DMARC alignment test will validate the email address in "<>." But when displaying the email address, it will not consider "<>" with a higher priority. (e.g. From: admin@legitimate.com, <any@attack.com>) (Discuss this attack in detail in 3.3.7)

1.3.14 Header spoofing 1 (A14). The attackers can perform a replay attack if the headers in "h=" tag are incomplete. They can modify the "unprotected fields in signed messages without invalidating DKIM signatures" and send it to other victims.

1.3.15 Header spoofing 2 (A15). The attackers can also perform a replay attack if the different email services have different parsing and interpreting mechanisms to treat the extra header. They can make use of this to add a new header (e.g. Subject) to the signed email. And some email clients will display the newly added one rather than the original one.

1.3.16 Body spoofing (A16). The attackers can perform a replay attack by exploiting the optional "l=" tag in the DKIM-Signature header. They could use the "l=" tag to "append some malicious contents to the original email body without breaking the DKIM signature."

1.3.17 No sufficient checks (A17). The attackers can send signed messages with others' addresses. If the email provider does not perform sufficient checks on the from header, the messages send by the attackers could pass both DKIM and DMARC.

1.3.18 The spoofing attack (A18). The attackers can first send deceptive contents to themselves without given the email providers strict validation. Then, they can add a new from header with another user's email address and resend it to the victims. Due to some implementation vulnerabilities (verify the original from header) of the DKIM validation on some email servers, the result will be positive.

After the researchers found these existing vulnerabilities, they have reported these vulnerabilities to the vendors, and many of them have fixed related issues.

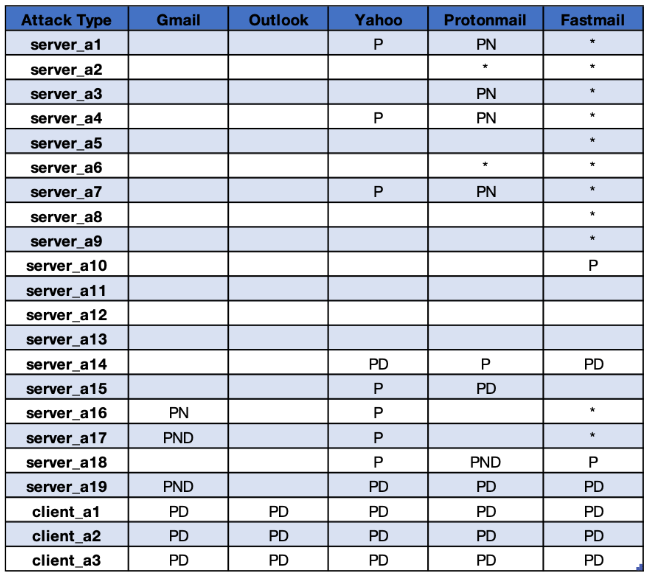
2 EXPERIMENT

2.1 Verification Process

As mentioned in the conference paper, some email service providers have started to employ countermeasures toward some of the spoofing attacks. We need to re-create these attacks to determine the scope of countermeasure that we will implement. To perform experimental attacks similar to those in the paper, we used the testing attacking tool called espoofer. Espoofer is an open-source testing attacking tool created by the author of the paper. It provides the 19 server mode testing attacks and 3 client mode testing attacks mentioned in the paper.

In order to use espoofer to re-create all of these email spoofing attacks, we needed to initialize an email server owned by us. We registered for a testing domain, "testdomaintest.company." With the testing domain, we then set up a mail server using Zoho. The DNS records of the testing domain were properly set to satisfy the SPF and DKIM info fetching requirements. However, because of the limitation of using Zoho instead of creating a manually initialized mail service server, SPF checks on the recipient might be shown as "soft fail" rather than "success." In the upcoming sections, we will explain this limitation had no material effect on our experimental results.

After properly setting up our email server, we then used our server along with the espoofer tool to sequentially send spoofed emails of both server mode and client mode attacks pretending the emails were sent by "security@cibc.com." We selected several major email service providers as our targets, including Gmail, Outlook, Yahoo, Protonmail, and Fastmail. We registered testing email accounts with each of the tested email service providers. The testing email accounts were named in the format of "cis6510victim@<corresponding domain>". The testing attacking results will be shown and analyzed in detail in the following section. During the testing process, we found that most Internet Service Providers (ISP), for instance, Bell, would block outgoing packets from port 25. Since sending spoofed emails through the espoofer tool requires packets going out via port 25, we found an alternative way for sending out the emails. One of the working alternatives, which is also the one we employed, was sending the packets via port 25 through hotspots of our cellphone instead of through home Internet. The drawback of this alternative was the sending frequency. At least for the email service providers we tested, all of them had prevention measures for frequent sendings from one IP address to an account. If the sending frequency limit were reached, the sender's IP address would be banned temporarily for various periods ranging from a few to dozens of minutes. Thus, even with the espoofer tool and several cellphones, the whole experimental attacking verification process still took a very long time, which was more than we expected.



Attack Type	Gmail	Outlook	Yahoo	Protonmail	Fastmail
server_a1			P	PN	*
server_a2				*	*
server_a3				PN	*
server_a4			P	PN	*
server_a5					*
server_a6				*	*
server_a7			P	PN	*
server_a8					*
server_a9					*
server_a10					P
server_a11					
server_a12					
server_a13					
server_a14			PD	P	PD
server_a15			P	PD	
server_a16	PN		P		*
server_a17	PND		P		*
server_a18			P	PND	P
server_a19	PND		PD	PD	PD
client_a1	PD	PD	PD	PD	PD
client_a2	PD	PD	PD	PD	PD
client_a3	PD	PD	PD	PD	PD

Figure 1: Experimental Results

P: Passed to spam folder
D: Wrong display name
N: Noticeable notification of authentication failure
*: Error message "I can break rules, too." returned

2.2 Identified Existing Problems

After attempting all testing attacks toward all of the target email service providers, we summarized the result as Figure 1. Overall, the vendors showed a much higher security level against these attacks comparing to the time when the authors of the conference paper tested against them. None of the three client mode attacks was working anymore. However, for each email service provider, the performances under our testing attacks of server mode were very different.

For Gmail, most of the 19 server mode testing attacks were filtered except for server_a16, server_a17, and server_a19. All of the testing attacking emails filtered by Gmail was not shown in either the inbox folder or the spam folder. The testing attacking emails of server_a16, server_a17, and server_a19 could pass Gmail's authentication checks and were placed in the spam folder. However, for all of the passed testing attacking emails, there was a large noticeable banner showing on the top saying that the email was identified as a spam email if the user chose to display the content of the email. Additionally, for server_a17 and server_19, the from header value of their corresponding emails could not show as "security@cibc.com" as we expected.

Outlook's performance was the best among all five target email service providers. None of the 19 testing attacking emails could pass its authentication validation and show in either the inbox or spam folder. Outlook employed the Spamhaus project as part of their authentication validating process. The Spamhaus project directly rejected most of the 19 testing attacking emails from entering into Outlook's reception service. Additionally, when we were logging in to check the reception status of these attacking emails, we found an interesting point that Outlook would freeze the account because of suspicious activities' involvement. The Outlook was the only vendor that we found with this interesting feature during the experiment.

For Yahoo, it had the worst performance in defending these testing attacks. In total, 9 of the 19 testing attacking emails could pass its authentication checks. All emails of these passed 9 attacks were placed in the spam folder except the one for server_a14. The email for attack server_a14 was put into the inbox folder. However, since its display of from header was not "security@cibc.com" as we expected, it became meaningless to get a chance being shown in the inbox folder. For the email of attack server_a19, it had a similar display name, which was not "security@cibc.com" as we would like. The rest 7 passed testing attacking emails were placed in the spam folder and probably would lead to successful phishing for regular users. Another significant vulnerability of Yahoo Mail was the lack of noticeable reminder of the spam checking result.

Protonmail's performance was only better than Yahoo Mail from the passing ratio point of view. There were 8 attacking emails passed its checks. Nevertheless, only the email of attack server_a14 can be considered a possible one that might lead to successful phishing. For emails of attack server_a15, server_a18, and server_a19, their display of From header was not "security@cibc.com" as we wanted. Meanwhile, if the user chose to show the email content, the rest 4 passed attacking emails, and the one for attack server_a18 showed a noticeable banner of authentication failure. A wise displaying setting of Protonmail was that it displayed the whole From header of the raw email file to the user, which would highly possibly make the attack meaningless even if the spoofed email passed the authentication validation.

For Fastmail, it had the mid-level performance. 4 of the attacking emails passed its validation. Among the 4 passed emails, the ones for server_a14 and server_a19 had the From header display issue as the other email service providers. None of the 4 passed attacking emails had a noticeable reminder of potential spamming. As mentioned in the conference paper, all the vendors under the paper's testing were notified about these potential attacks. Fastmail and Protonmail did put efforts into fixing these corresponding vulnerabilities. Both of them were found to reply to an error message like "I can break rules, too." for some of the attacks during our experimental process.

2.3 Yahoo Mail Service

From the result of our identified problems, we concluded that Yahoos' mail server provides the worst validation system because there are seven attacks that might lead to successful phishing, which is much higher than the second-worst mail server, Fastmail who only has two. Besides, compared to the other tested mail servers, Yahoo neither had a detection system like Outlook to reject messages nor display warnings like Google. Also, as for the Protonmail whose spoofed email passing number was close to Yahoo, but it contained the functionality of showing the whole from header, allowing the users to discover the attackers' crafts. However, Yahoo did not contain any protection system like that. In addition, due to the difference of raw email files from different mail services, if we adapt our parser to recognize raw email files from the mail services other than Yahoo, we need to spend more time but gain little effect, which is not worth. Finally, we decided to only focus on Yahoo's raw email files to build our parser and the corresponding detection system.

3 COUNTERMEASURE

To mitigate the potential damage that the successful attacks toward Yahoo Mail Service may cause, we have implemented a comprehensive countermeasure working as a double checker to verify whether the emails in the spam folder are indeed spoofed emails or false-positive ones. This countermeasure tool is named Anti-Espoofer. Anti-Espoofer takes raw email text input of the emails passed the Yahoo Mail Service checking process and located in the spam folder. With the raw email text input, Anti-Espoofer then sequentially works on parsing the raw email, handling the parsed result, and calling the corresponding attacking double checker to draw the final conclusion. In the following sections, the three working steps of Anti-Espoofer will be described and explained in detail.

3.1 Parsing Raw Email Text Input

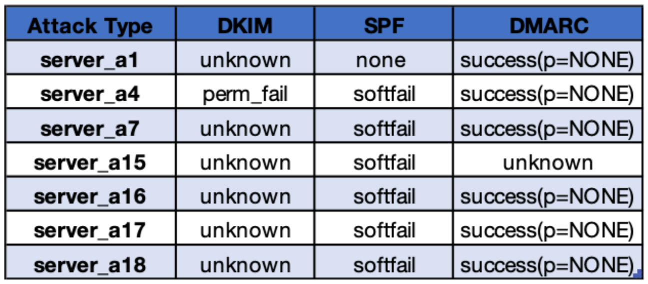
Firstly, Anti-Spoofer needs to parse the raw email text input so that the later handling and verifying steps can effectively fetch the information contained in the raw email. The RawEmailParser module located in raw_email_parser.py is in charge of parsing the raw email input file. The RawEmailParser module splits the parsing process into two steps: initial parsing and parsing modifications. During the initial parsing step, the RawEmailParser module uses an existing Python library mail-parser to get a parsing result draft. Mail-parser[7] is a parsing tool using email Python Standard Library to convert raw mail text files to Python objects. With the initial parsing result object, the RawEmailParser module then modifies certain entries for later use. The modified entries include From field in the header, To field in the header, Authentication-Results section, Received section, and Return Path. The parsing modification process mainly uses regular expressions to capture and re-categorize these mentioned entries from the initial parsed result object.

After the modification, the From entry contains a string copy of the content in the From field of the original raw email file header regardless of whether the content is the valid value for From field of an email. The modified To entry contains a list of all email addresses in the To field of the original raw email file header. The modified Authentication-Results entry includes an object containing three sub-objects: DKIM, SPF, and DMARC. Each sub-object has the corresponding testing result of that test and the extra information for that test. For the modified Received entry, it contains the re-formatted information of SMTP and HTTP receiving information separately. Lastly, the Return-Path entry includes the original full return path, the real return path domain, and the return path after stripping the route portion, if applicable.

3.2 Handling Parsed Result

With the parsed result from the raw email text file, Anti-Espoofer then handles the parsed result by detecting and matching the fingerprints of the attacks mentioned in the conference paper that can pass the spam dropping mechanism of the Yahoo Email Service. If the parsed result matches any attack fingerprints, then the corresponding attacking double checker is called. The double-checking result is displayed to the user after the execution of the attacking double checker. Specifically talking about the fingerprint detecting and matching process, each of the seven types of attacks that successfully passed the spam dropping mechanism of Yahoo Email Service has its identical characteristics. These identifiers will be discussed here. And the more detailed attack descriptions will be discussed in the following section.

For server_a1, server_a4, and server_a15 attacks labeled in the espoofer tool, they can be easily detected and matched by their identical authentication result statuses combination in the parsed result. However, for server_a7, server_a16, server_a17, server_a18 attacks, their authentication result statuses combination in the parsed result are the same. Thus, more specific characteristics in the parsed result have been used for identifying these four types of attacks from each other. Since server_a7 attack aims to include route portion in Return Path to bypass the SPF authentication, the route portion identifier ":" is used to detect this attack. Attack server_a16 uses a similar strategy as server_a7 attack. But it includes the route portion in the From field of the header. Then, the found ":" in the From field identifies this attack. For server_a17 and server_a18 attack, both of them insert particular parts into the From field of the header to introduce chaos in order to bypass. Because of that, the unique identifier "\" and not starting with "<" are used as the criteria to confirm server_a17 and server_a18 attack correspondingly.



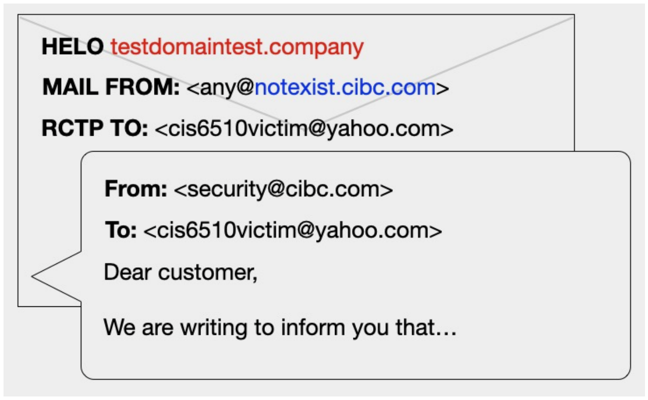
Attack Type	DKIM	SPF	DMARC
server_a1	unknown	none	success(p=NONE)
server_a4	perm_fail	softfail	success(p=NONE)
server_a7	unknown	softfail	success(p=NONE)
server_a15	unknown	softfail	unknown
server_a16	unknown	softfail	success(p=NONE)
server_a17	unknown	softfail	success(p=NONE)
server_a18	unknown	softfail	success(p=NONE)

Figure 2: Yahoo Authentication Validation Results

3.3 Specific Attack Verification

For each successful attack, we analyzed the validation result of Yahoo's DKIM, SPF, and DMARC.(Figure 2) According to the combination of the outcomes, we classified the various types of attacks and parsed the email raw files into the corresponding detections based on their categories.

3.3.1 server_a1 (Figure 3). Server_a1 attack is introduced as a non-existent subdomains (A1) attack. The technique used for this attack is to forge a non-existent subdomain in SMTP (mailfrom.notexist.cibc.com), which looks like belonging to a legitimate domain. After

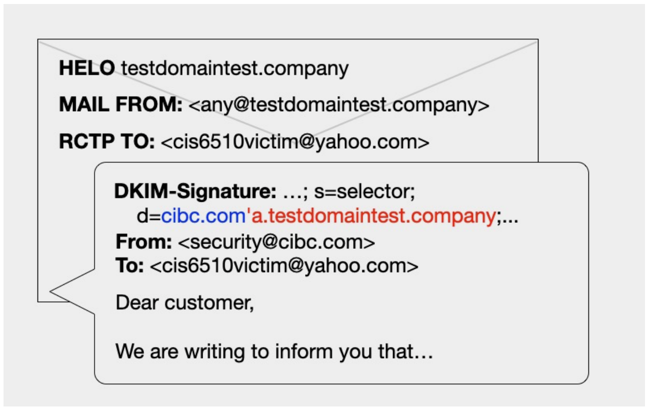


```
HELO testdomaintest.company
MAIL FROM: <any@notexist.cibc.com>
RCTP TO: <cis6510victim@yahoo.com>

From: <security@cibc.com>
To: <cis6510victim@yahoo.com>
Dear customer,

We are writing to inform you that…
```

Figure 3: server_a1: Non-existent Subdomains

performing this attack, the SPF result is none because the non-existent subdomain does not contain any SPF policy, leading to the SPF unable to verify the mail from in SMTP. As the mechanism of the email verification, DMARC will perform the following alignment test. In this case, the result of DMARC is a success due to the domain mail from (cibc.com) in SMTP is a subdomain of the domain in from header. At this point, DMARC authentication is subverted because both detections in SPF and DMARC provide a positive result.

In our experiment, if we forge the subdomain as an existent one, the SPF will fail (hardfail) because it queries the real legitimate domain for the verification, which cannot be controlled by the attackers. In this case, the email server will directly reject the spoofed email. So only the non-existent subdomain can pass the SPF's validation. According to the footprint of the SPF's none result, we indicate that it might belong to the server_a1 attack. To verify the message, we will judge whether the subdomain in SMTP is exactly the same as the domain in from header. In this way, we could avoid the validation vulnerability of the SPF.

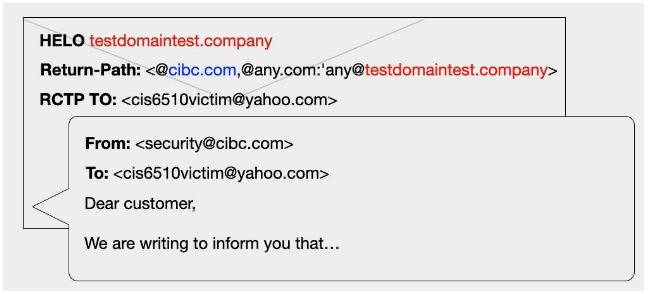


```
HELO testdomaintest.company
MAIL FROM: <any@testdomaintest.company>
RCTP TO: <cis6510victim@yahoo.com>

DKIM-Signature: …; s=selector;
  d=cibc.com'a.testdomaintest.company;...
From: <security@cibc.com>
To: <cis6510victim@yahoo.com>
Dear customer,

We are writing to inform you that…
```

Figure 4: server_a4: DKIM Authentication Results Injection

3.3.2 server_a4 (Figure 4). Server_a4 is introduced initially as DKIM authentication results injection (A4) attack. The attackers can generate a verifiable DKIM-Signature by signing with their own private keys and embedded a literal open parenthesis "(" in the "d=" value. When the email server receives the message, the DKIM will query the domain after the parenthesis, which is controlled by the attackers to do the DKIM authentication by obtaining the attacker's public key. DMARC alignment test will also pass because it queries the domain before the parenthesis, which is the forged legitimate domain, and compares it to the domain in the from header. According to the authors' research, besides open parenthesis, "(", double " and single ' quote also works for this kind of attack.

However, in our exploration, the reason why the email server (Yahoo.com) receives this message is not because of the DKIM-Signature. The result of DKIM is perm_fail (hardfail) since the single quote does not spoof the DKIM successfully. We find that the SPF check shows softfail (considered success), and we see it queries the attacker's domain in SMTP for this check. DMARC alignment test results positive because it compares the domain before the single quote and the domain in from header. According to our research, the forged DKIM-Signature does not work as expected to pass the DKIM authentication. However, we find this mechanism could deceive the DMARC from getting the domain that needs to be verified. We add a specific comparison between the mail from in SMTP and the domain in from header to solve this vulnerability. By doing so, we correct the DMARC alignment test because the test is supposed to obtain the SMTP domain in the SPF rather than the domain in the failed DKIM.

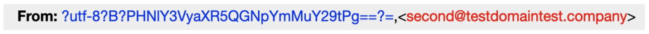


```
HELO testdomaintest.company
Return-Path: <@cibc.com,@any.com:'any@testdomaintest.company>
RCTP TO: <cis6510victim@yahoo.com>

From: <security@cibc.com>
To: <cis6510victim@yahoo.com>
Dear customer,

We are writing to inform you that…
```

Figure 5: server_a7: SPF authentication results injection

3.3.3 server_a7 (Figure 5). Server_a7 attack is introduced as an SPF authentication results injection (A5) attack. The attackers could forge a malformed email address in the return-path of the email server. In this case, based on the recognition of the SPF, it will take the last "@" as the delimiter, which means the SPF checks the attacker's domain (testdomaintest.company). And the SPF result from Yahoo's server is softfail (considered success). However, the DMARC alignment test has some implementation problem which takes the first "@" as the delimiter from the return path to obtain the legitimate domain (cibc.com) and check it with the spoofed legitimate domain in from header (cibc.com), which result in success. In this way, the attacker can successfully bypass the validation because the result of SPF and DMARC are both positive. From our analysis, the problem is that the return path is forged by multiple "@" to confuse the SPF authentication and the DMARC alignment test. In our case, the SPF delimits the right domain from the path. What we need to do is to fix the mistake verified by the DMARC.

We filtrate the real domain from the return path (remove forged domains before the last @) and identify if the domain in from header can match it.

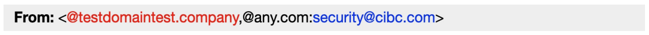


From: ?utf-8?B?PHNlY3VyaXR5QGNpYmMuY29tPg==?=,<second@testdomaintest.company>

Figure 6: server_a15: Email Address Encoding

3.3.4 server_a15 (Figure 6). Server_a15 attack is introduced as an email address encoding (A10) attack. The premise of the attack is the complex from header syntax. Due to the different implementation of the email server and client, the email addresses extracted in from header could be different. This mechanism brings a challenge for email verification. In the from header, the attacker could encode the forged legitimate email address (?utf-8?B?PGFkbWluQGxlZ2l0a W1hdGUuY29tPg==?=) and accompany with the attacker's email address (second@testdomaintest.company). And for some mail servers (in our case Yahoo.com) could not recognize it. In this situation, the SPF result is softfail (considered success) because it will query the attacker's domain (testdomaintest.company), and the DMARC alignment test results in unknown since it does not support the decode features.

From our test result, the footprint of this situation is unique because the DMARC cannot identify the encoded from header, which results in none. Based on this feature, the solution to solve this is that we first do a check of the encoded part in from header. Then, we decode the email addresses and append their domain into the unencoded domains' list. After that, we check whether each domain in that list can match the domain in SMTP. If it comes up a domain is different from the SMTP domain, it means that this domain is an illegitimate one, and the users will be notified.

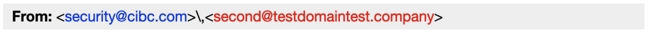


From: <@testdomaintest.company,@any.com:security@cibc.com>

Figure 7: server_a16: Route Portion

3.3.5 server_a16 (Figure 7). Server_a16 attack is introduced as a route portion (A11) attack. Route portion is an outdated feature which originally supports the delivery path that the message is supposed to follow. The syntax of it is to consider comma as a separator of different domain names, and each domain name is behind the delimiter "@". The terminator of the route portion is a colon. In this case, the SPF is softfail (considered as success) because it will query the attacker's domain (testdomaintest.company). By forging a route portion in from header and with the server's defect of recognizing the route portion, the attack could bypass the DMARC alignment test, which shows success.

In this case, we have a basic inspection of if the from header contains a route portion. Once we match the route portion pattern (by checking the colon in from header ":"), we need to screen out the real domain without the route portion. Then we use the domain to compare with the domain in SMTP for making up for the defect (cannot recognize the route portion) of the DMARC. If the domains could not match, we indicate that this email contains the route portion attack.

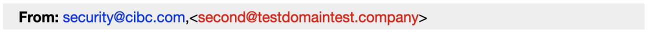


From: <security@cibc.com>\,<second@testdomaintest.company>

Figure 8: server_a17: Quoted-Pairs

3.3.6 server_a17 (Figure 8). Server_a17 attack is introduced as a quoted-pairs (A12) attack. The internet message format has reserved some specific characters for the particular use. In order to use these characters as a part of the uninterpreted data, "\" could be used as a way to achieve that by the email senders. After performing this attack, the SPF check shows softfail (considered success) since it does the test with the attacker's domain (testdomaintest.company). And the DMARC alignment test provides a positive result because of the different implementations of recognizing the quoted-pair feature, which leads the email server (Yahoo.com) to query the forged domain for doing the verification.

It is simple to solve the quoted-pairs problem. To achieve this attack, the attackers must forge the multiple from headers with the "\". Once we detect this specific symbol, we remove it. (use regex to do the pattern match). In this way, we could recover the use of quoted symbols "<>", and then we could obtain all the domains in from header. Then we use these domains to match the domain in SMTP as a means of remedial measure for the flaw of the DMARC alignment test.



From: security@cibc.com,<second@testdomaintest.company>

Figure 9: server_a18: Parsing Inconsistencies

3.3.7 server_a18 (Figure 9). Server_a18 attack is introduced as a parsing inconsistencies (A13) attack. The mechanism of this attack depends on the implementation of how the email server will treat the precedence of the domain with "<". Some servers regard "<" as a higher priority, so the authentication queries the domain inside the chevron and vice versa. In our experiment, the SPF result in softfail (considered as success) by doing the test with the attacker's domain (testdomaintest.company). And for the DMARC alignment test, we find that Yahoo's server considers "<" with a lower priority, which leads to success as a result.

In this situation, we face the multiple from headers with or without chevron ("<"). To solve the vulnerability due to various email servers' different precedence implementations, we remove all the chevrons (by regex pattern matching) and obtain the domain names in from header. In this way, there are no priority issues, and we can use these domains to match the SMTP domain. Similar to the above attacks' solution, if there is a domain in from header that cannot match the domain in SMTP, we conclude that it is a spoofed email.

4 SUMMARY

According to our experiment to all the attacks introduced by the researchers, we found that many vendors have fixed many vulnerabilities. We finally select Yahoo Mail as our target since it receives the most (seven) kinds of spoofed emails among all the five email services we have tested. Additionally, Yahoo Mail does not have noticeable reminders for emails that are considered spam emails. Due to Yahoo's mail service's unique implementation, we have found its verification practice (DKIM, SPF, DMARC) performed very differently from the other vendors'. Yahoo Mail's authentication verification process is not the same as what the former researchers described since the researches mainly focus on the authentication mechanisms in general. Based on Yahoo's identical performance, we have built the tool Anti-Espoofer, which is highly customized for all the exploitable cases on Yahoo Mail under our experimental attacks. Anti-Espoofer works as a complement of Yahoo Mail's existing authentication checking, which covers the vulnerabilities caused by implementation differences between DKIM, SPF, and DMARC. Anti-Espoofer basically utilizes the footprints of the different attacks to recognize them and passes the raw files into the corresponding validation functions. From the given back result, the users can easily know whether the test has passed and which kinds of exploits those attacks belong to.

5 FUTURE WORK

Our next step is to add or modify our parser, which will let it be able to parse different kinds of email raw files from different email service providers. Once our parser gets more powerful, our Anti-Espoofer will have better adaptability. Anti-Espoofer would be able to handle more exploit cases on various email services, such as the server_a10, server_a18 attack toward Fastmail, as well as server_a14 attack against Protonmail. After that, another step we can take to improve the usability of Anti-Espoofer would be making it a browser extension. We will learn how to turn our Anti-Espoofer into a browser extension, which could expand the user base. Data shows that 8% more accesses of emails are through webmail clients than desktop clients' count of accesses [1]. An extension version of Anti-Espoofer, as a complement of email service vendors' existing protecting mechanisms, could help a larger number of people to detect the potential spoofed emails and prevent the following phishing or even worse consequences.

REFERENCES

[1] Kenneth Burke. 2020. How Does Mobile Email Engagement Compare to Desktop Rates? https://www.textrequest.com/blog/mobile-email-engagement-compare-desktop-rates/

[2] Jianjun Chen, Vern Paxson, and Jian Jiang. 2020. Composition Kills: A Case Study of Email Sender Authentication. *29th USENIX Security Symposium (USENIX Security 20)* (2020), 2183–2199. https://www.usenix.org/conference/usenixsecurity20/presentation/chen-jianjun

[3] Pat Foran. 2020. Canadians lose at least $130 million in scams last year. https://toronto.ctvnews.ca/canadians-lose-at-least-130-million-in-scams-last-year-1.4799282

[4] Stefan Görling. 2007. An overview of the Sender Policy Framework (SPF) as an anti-phishing mechanism. *Internet research* 17, 2 (Apr 10, 2007), 169–179. https://doi.org/10.1108/10662240710737022

[5] Hang Hu and Gang Wang. 2018. Revisiting Email Spoofing Attacks. (Jan 2, 2018). https://arxiv.org/abs/1801.00853

[6] M. Kucherawy and Elizabeth D. Zwicky. 2015. Domain-based Message Authentication, Reporting, and Conformance (DMARC). *RFC* 7489 (2015), 1–73. https://www.semanticscholar.org/paper/Domain-based-Message-Authentication%2C-Reporting%2C-and-Kucherawy-Zwicky/2283952c2a94a879020492979de9f73c9e822baa#citing-papers

[7] Fedele Mantuano. 2020. PyPI: The Python Package Index. https://pypi.org/project/mail-parser/

[8] Jonathan B. Postel. 1982. Simple mail transfer protocol. https://tools.ietf.org/html/rfc821