

All your emails belong to us: exploiting vulnerable email clients via domain name collision

Ilya Nesterov
Shape Security
ilya@shapesecurity.com

Maxim Goncharov
Shape Security
maxim@shapesecurity.com

ABSTRACT

The Autodiscover HTTP Service Protocol provides a way for Autodiscover clients to find Autodiscover servers. This protocol extends the Domain Name System (DNS) and directory services to make the location and settings of mail servers available to clients. In this paper, we take a closer look at the Autodiscover protocol and identify its threat model. We analyse Autodiscover client implementations in two mobile built-in email clients to discover flaws which allow remote attackers to collect user credentials through domain name collision. We discover how many clients have vulnerable implementations by collecting and analysing HTTP request information received by our servers, registered with specially crafted domain names. We make our analysis based on data we collect from 25 different domains. Our dataset contains information on about 11,720,559 requests and we observe 9,726,028 requests containing authentication information. We identify 2473 different email clients which use vulnerable Autodiscover client implementation. Finally we propose different mitigation techniques for users, enterprises, and application developers to improve their email clients.

Keywords: email; Autodiscover protocol; Autodiscover client; Microsoft Exchange

I. INTRODUCTION

One of the central points of failure is an email address. We use email addresses to get access to bank accounts, social networks and much more. For SMB and Enterprise, email addresses are the most often targeted entry point for advanced persistent threat (APT) attacks. But how good are we at protecting our email accounts? There's always a tradeoff between security and usability. There was a time when you would need to obtain all the information about SMTP/POP/IMAP servers in order to configure your email account. Now it is as simple as just typing your email and password. But when you rely on technology that simplifies your life, but is complex and sophisticated inside, there is always a risk of failure in implementation.

Email infrastructure is old and mature, and serves billions of emails a year. Considering email service as a safe way of communication is not a valid viewpoint today, but still email communication is one of the most important and prevalent means of information exchange. The design of email clients, and how they communicate using Simple Mail Transfer Protocol, Internet Message Access Protocol or Post Office Protocol is defined by RFCs from the mid 80s [1],[2],[3]. The deployment and updating of email configuration for an email client has always been a challenge and this is why the Autodiscover protocol was introduced.

Autodiscover protocol overview

The Exchange Autodiscover service provides an easy way for an email client to configure itself with minimal user input. Most users know their email address and password, and with those two pieces of information, the email client can retrieve all the other details it needs to run. For Exchange Web Services (EWS) clients, Autodiscover is typically used to find the EWS endpoint URL, but Autodiscover can also provide information to configure clients that use other protocols. [4]

Simply explained, the Autodiscover client expands the email address provided by users during initialization to derive a list of Autodiscover server URLs which are then used to get all the information needed for the email client to operate.

Overview of the generic autodiscover process

Essentially the Autodiscover process has two main phases:

- Phase one: client generates list of potential Autodiscover servers
- Phase two: Autodiscover client tries each server from a list until it gets a response

Other Microsoft documentation [4] suggests a 3 phase process which is essentially the same. The only difference is that the list of potential Autodiscover services is split into

two categories. First the Autodiscover service tries to connect high priority groups of URLs. Then, if none of the potential Autodiscover servers works out, the client switches to a "last ditch" attempt to find an Autodiscover endpoint.

Let's take a closer look at these phases.

Phase 1: Defining the candidate pool

Before the Autodiscover client can communicate with the Autodiscover server it must locate the right Autodiscover server. The protocol defines the following ways to add Autodiscover servers into the pool:

- Query a well-known LDAP or AD servers [4]
- Perform text manipulations on the domain portion of the email address
- Search the DNS for Autodiscover SRV records
- Send an unauthenticated GET request to an endpoint derived from the user's email address

Query Well known LDAP or AD DS servers

Depending on the chosen protocol, the Autodiscover client performs LDAP or SCP queries. Upon success, the results are added to a pool of potential autodiscover servers.

Perform text manipulations on the domain portion of the email address

Autodiscover defines two standard endpoint URL forms that are derived from the domain portion of the user's email address. The following URLs must be added:

- `https://+ {domain} + /autodiscover/autodiscover.xml`
- `https://autodiscover. + {domain} + /autodiscover/autodiscover.xml`

If an HTTP POST to either of the above URIs results in an HTTP 302 redirect, then the redirect as found in the location field of the response is added to the list of possible Autodiscover server URIs.

Note: Autodiscover HTTP Service Protocol specification [5] and Autodiscover for Exchange [4] define these two URLs differently. In the first document, the first derived URL endpoint should use unsecure HTTP protocol. The second document, however, requires usage of HTTPS.

Search the DNS for Autodiscover SRV records

An Autodiscover client can find an Autodiscover server URL by querying the DNS server for the autodiscover SRV record, using the following query format: `_autodiscover._tcp.<domain>`.

If the result is `<host>`, add `"https://<host>/Autodiscover/Autodiscover.xml"` to the list of possible Autodiscover URIs.

Send an unauthenticated GET request

An Autodiscover client can also issue an HTTP GET method with the URI set to `"http://Autodiscover.<domain>/Autodiscover/Autodiscover.xml"`

Autodiscover servers list prioritization

When multiple candidates are found, Autodiscover also defines a way to generate and prioritize the list [6] as shown in Figure 1.


LDAP/AD QUERY	<code>https://mail.example.com/autodiscover/autodiscover.xml</code>	
LDAP/AD RESULTS	<code>https://example.com/autodiscover/autodiscover.xml</code> <code>https://autodiscover.example.com/autodiscover/autodiscover.xml</code>	
GET REQUEST	<code>http://autodiscover.example.com/autodiscover/autodiscover.xml</code>	
DNS QUERY	<code>_autodiscover._tcp.example.com</code>	

Figure 1: Autodiscover servers list prioritization

Phase 2: Trying each candidate

Once the list of potential Autodiscover servers is generated, the next step is to try each one in the list by sending a request to the URL and validating the results as shown in Figure 2.

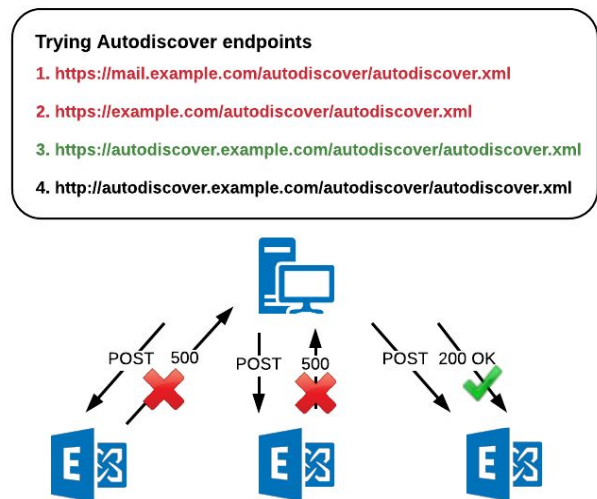


Figure 2: Autodiscover server discovery process

II. THREAT MODEL

In our work, we wanted to focus on exploitation techniques that let us gain access to highly protected assets such as user account data, including email address and password or other authentication mechanisms.

Since all requests to potential Autodiscover servers should be authenticated this makes it a perfect target for an attack.

There are a few different ways that an Autodiscover protocol can be attacked. In our threat model we decided

not to focus on man-in-the-middle attacks, although this is one of the possible high risk threats. Since it is only possible to attack LDAP or AD servers when the Autodiscover client is in a protected area behind a firewall, we chose to investigate a potential flaw in the algorithm used by the Autodiscover client to build a list of potential Autodiscover servers derived from the email address.

Email address complexity

The format of an email addresses is local-part@domain. On first glance, deriving the domain part from any given email address should be a trivial task. Unfortunately, however, this is much more complicated thanks to the multiple RFCs defining the format: RFC 5321 [7], RFC 5322 [8], RFC 6531 [9], RFC 6532 [10].

After reading through all these RFCs, deriving the domain portion from the user's email address doesn't look like a trivial task. Your application cannot just split an email address string by '@' character, because "my@email"@example.org is a valid address as well as "()<>[:;@\\\"!#\$%&'!/?^`_}{| ~.a"@example.org and simply user@example.org.

Of course, an application should implement input validation logic for cases when user inputs an invalid email address like john.doe@example..com (with double dot after '@') or john@doe@example.com (with two '@' characters) before deriving the domain part.

But this is just a part of the problem for the Autodiscover client. For instance, the email address john.doe@com.au or any other email address where the domain part is a public domain suffix [11] or a top level domain (TLD) [12] will be under a huge risk, because someone can register the autodiscover.com.au domain and receive autodiscover client's requests (which contain the user's credentials). Therefore, the Autodiscover client should implement some logic to understand whether the domain part is not a TLD and is not a public suffix list.

Autodiscover servers at complex environment

Large organizations usually have a complicated domain infrastructure with several subdomains, specific to different parts of a company. It is also quite typical that different departments, or overseas offices have their own country or department specific domain, so users can have complicated email addresses such as user@department.uk.example.com or user@uk.example.com. This adds another layer of complexity.

The Autodiscover protocol doesn't explain how developers and administrators should deal with the situation when the domain portion of an email address contains multiple subdomains. There are several possibilities on how it can be implemented:

The administrator should redirect all requests from

multiple potential Autodiscover servers to a single one, or run multiple Autodiscover servers.

The Autodiscover client developers should implement additional logic and add all autodiscover + subdomains to a pool of possible Autodiscover servers during Phase 1. e.g. user@uk.example.com as an email address might produce a list of domains autodiscover.uk.example.com, autodiscover.example.com. Of course, in such an implementation you should check that example.com is not in a public suffix list for the reason we explained above.

III. METHODOLOGY

In order to find out how secure the implementations of Autodiscover clients are, we decided to analyse email clients built into the iOS and Android mobile operating systems.

Testing mobile Autodiscover clients

According to our threat model, improperly deriving the domain part from an email address might lead to a leakage of user credentials. So, to understand if this happens we needed to build a test environment which allows us to do the following:

- Capture all network traffic from the mobile device for future analysis. We are most interested in DNS, HTTP, TLS/SSL protocols.
- Decrypt TLS/SSL traffic.

We also need to build a set of valid and invalid email addresses to test the Autodiscover client logic.

Finding more vulnerable clients

It is almost impossible to find all email clients or services which implement the Autodiscover client, so it is also impossible to test them all. But after analyzing the threat model, the only possible scenarios when critical information can be leaked to external, non-controlled environments is when an Autodiscover server URL's host part is derived as a public domain in the following forms:

- autodiscover + <TLD>
- autodiscover + <public domain suffix>

In order to collect data, we registered a few AUTODISCOVER domains as shown above, and set up an HTTP server. To terminate the TLS connection and support HTTPS we obtained SSL certificates signed by a publicly trusted certificate authority. All request information was logged into an http access log for future analysis. Because request data might potentially contain access credentials we made sure not to record this data. The only recorded HTTP header is the User-Agent, which is obviously the one we are interested in, to see how many Autodiscover clients potentially might be affected.

IV. RESULTS AND ANALYSIS

We tested two mobile built-in email clients from two vendors to figure out whether they have any issues with the Autodiscover client implementation. It turns out that the discovered flaws could be classified in the following way:

- Domain part improperly derived from email addresses for the second-level domains (SLD). For instance, john.doe@example.com.au is an email address for the organization registered domain example in a second-level public domain .com.au
- Email address with double @ characters is allowed which leads to deriving the wrong domain part, and as a result the Autodiscover client server sends requests to a different server.
- Email address for a TLD and SLD allowed without any warning to a user, which leads to leaking sensitive information such as user credentials to an unauthorized control sphere.

Samsung mail client analysis results

We used the Samsung Galaxy S5 model SAMSUNG-SM-G900A running Android version 6.0.1 with Android security Patch level January 1, 2017. Built-in Mail App version is 5.0.0.0400. The discovered vulnerability is in the way the Autodiscover client builds a list of potential Autodiscover servers. In particular, if company domain is registered in a second-level domain, then the two following autodiscover servers will be added to a list:

- autodiscover. + <domain> + SLD
- autodiscover. + SLD

That doesn't happen in the case when a company domain is registered in a top-level domain.

For example if the email is john.doe@example.com.au than the following Autodiscover servers will be added to a list:

- autodiscover.example.com.au
- autodiscover.com.au

Domain name autodiscover.com.au could be easily registered by anyone to passively collect information about email accounts and credentials, for any domains registered with .com.au, when users use email clients with such flaw.

Apple iOS Mail app analysis results

For Apple iOS tests we used iPhone 6s running iOS 10.2.1 (14D27). We tested the built-in Mail app which has capabilities to add exchange email account.

The flaw we discovered leads to leaking user credentials to a third party autodiscover domains in case user email address contains two @ characters without first one being included into quotes. For example if a user enters an email address as john.doe@example@org, the Autodiscover

client will derive the domain as .org and add the Autodiscover server to a list as autodiscover.org which could be owned by a different organization and will allow collect credentials.

Another issue with both the Samsung Mail app and the Apple iOS Mail app is that they allow users to enter email addresses, with a domain part that is a TLD or SLD. That leads to user credentials leaking to publically registered autodiscover domains. For instance john.doe@com.au is a valid email address, but whoever is an owner of com.au should take care about the security of user accounts and not allow anyone to register and own autodiscover.com.au domains to protect users. Another possible solution would be to not allow the Autodiscover client to add potential Autodiscover servers into a list if the derived domain part from a given email address is in the IANA Root Zone Database [12] or in the public suffix list [11].

Autodiscover domains HTTP server access log analysis

To understand the bigger picture and see how many users might be affected by the issues we discovered in Samsung and iOS built-in Mail apps, we registered 25 different autodiscover domains and pointed their traffic to a single http server. The access log data contains about 12GB of information for the period of time August 27 2016 to February 25 2017. For this period of time our server received 11,720,559 requests. 11,097,143 of the requests are requests from different Autodiscover clients and 9,726,028 of these requests have been sent with user credentials using Basic authentication. You can see month-by-month the Autodiscover clients traffic in Figure 3.

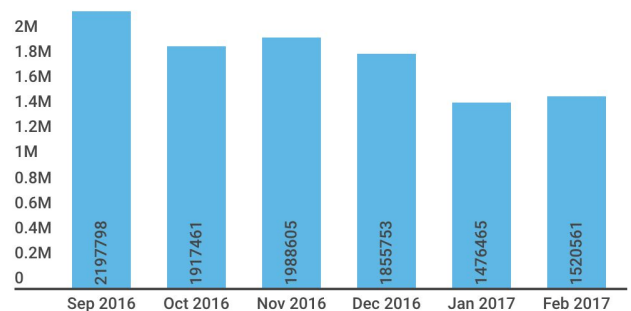


Figure 3: Autodiscover clients requests per month

Discovered vulnerabilities affected 212,307 individual email accounts on 65,576 different domains. The distribution of affected email accounts per platform is shown in Figure 4.

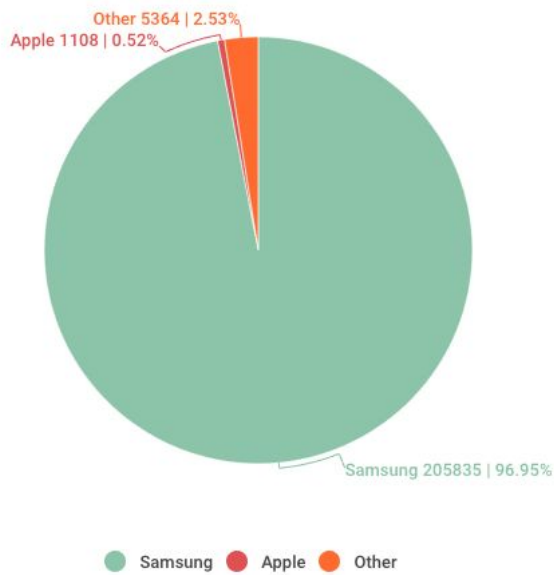


Figure 4: Number of email addresses affected by platform

in Figure 5 you can see the distribution of Autodiscover client requests (with credentials) using non secure (HTTP) and secure (HTTPS) protocols.

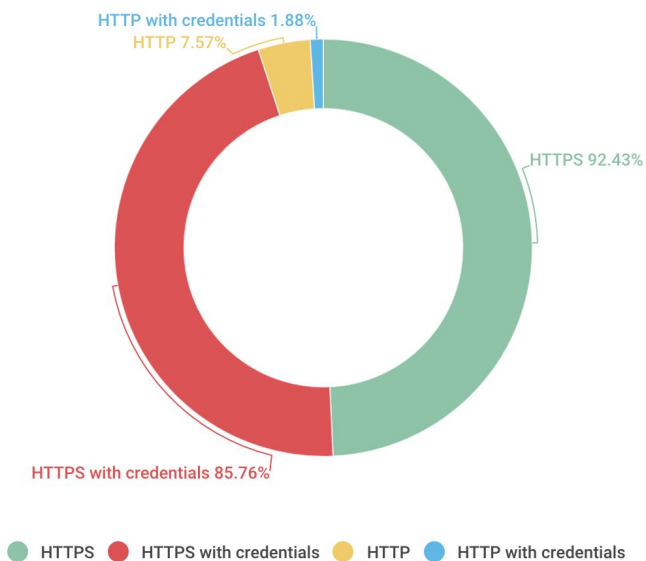


Figure 5: Autodiscover client traffic distribution HTTP vs HTTPS

The total number of different User-Agents affected by the improper implementation of Autodiscover client is 2,743. Figure 6 shows the number of Apple and Samsung clients affected by the issues we discussed above, in comparison with other clients which have similar Autodiscover client implementation issues.

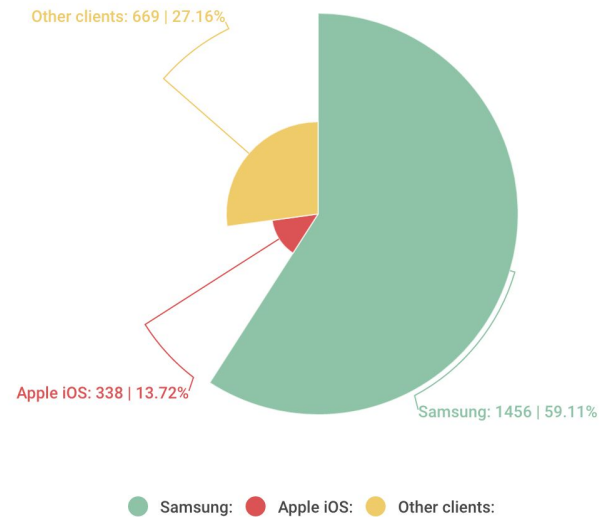


Figure 6: Comparison of Apple and Samsung vulnerable clients vs. all other clients

Vulnerability disclosure to vendors

We communicated discovered vulnerabilities to Samsung and Apple.

MITRE assigned the following CVE IDs for the issues we described above:

- CVE-2016-9940 Samsung mail client
- CVE-2017-2414 Apple iOS mail client

As a result both issues address and delivered as a security updates.

Samsung delivered the fix as a part of a monthly Security Maintenance Release (SMR) process in January 2017 [13].

Apple delivered the fix for the vulnerability in March 2017, as a part of iOS 10.3 security update [14].

V. CONCLUSION

The Autodiscover protocol is designed to seamlessly provide any complex configuration needed for the email client to communicate with an email server based on user entered email address and password information. It also makes changes to infrastructure that is easily propagated to clients. In this paper we reviewed the Autodiscover protocol implementation and discussed its threat model. We also explained the complexity of deriving the domain part from a given email address and how it can lead to leaking user credentials through vulnerable autodiscover protocol implementations.

We analysed built-in email clients on two popular mobile platforms and explained vulnerabilities in their Autodiscover client implementation.

We developed and deployed an autodiscover server sinkhole to collect data about different vulnerable Autodiscover clients. Collected data shows a significant

number of the clients which trying to get the Autodiscover configuration from a different autodiscover domain names we have registered for our project. The fact that all these clients blindly send user's credentials along with autodiscover requests might allow even non experienced attackers to passively collect huge amounts of sensitive user information, such as email address and credentials, which later might be used for other types of attacks.

The number of available autodiscover domains, which anyone can register, creates a huge risk for organizations, taking into consideration the number of vulnerable clients. In order to lower the risks we suggest the following possible mitigation approaches as described below.

Mitigation

Users:

Use only the email client approved and recommended by your IT department. Discovered flaws only affect clients implemented using the Autodiscover protocol, thus only if your organization uses Microsoft Exchange.

Enterprises:

Make sure your services are deployed as recommended by vendor and Autodiscover clients are able to discover and connect Autodiscover servers inside corporate networks as well as outside your protected perimeter.

In order to protect your users make sure that you use clients officially recommended by Microsoft to work with exchange servers. If you use third-party clients, use the threat model we described in our work to test your final deployment and double check that there is no sensitive data leakage to third-party domains.

Application developers:

If your application needs to implement the Autodiscover client protocol, follow the Autodiscover HTTP service protocol specification [2] and best practices recommended by Microsoft. Remember proper email address validation and techniques that allow properly derived domain and local parts from an email address.

ICANN:

Due to the very large number of vulnerable clients implementing the Autodiscover protocol in such a way to allow passive attackers to collect sensitive information, and the possibility that many of these clients will never be fixed (due to lack of support or other circumstances), we recommend considering placing a ban on registration of new autodiscover domain names for top, second level public domains and assigning existing ones to 127.0.53.53 [11] as it was implemented for the Web Proxy Autodiscovery protocol. [15]

VI. FUTURE WORK

Active attack vector

We are continuing our work on discovering threat vectors around the Autodiscover protocol. One of the evil methods that can be used by the attacker, is the ability to respond to the GET request of the client with autodiscover.xml file with IMAP/POP3/SMTP configurations that differ from the settings the client intended to receive. IMAP/POP3/SMTP can be pointed to a man-in-the-middle server where access credentials and mail communication can be eavesdropped. One of the related problems to the attack we have described above is that there is no way to implement certificate pinning for out of the box email clients.

VII. REFERENCES

- [1] Jonathan B. Postel, RFC 821: SIMPLE MAIL TRANSFER PROTOCOL. Available <https://tools.ietf.org/html/rfc821>
- [2] Network Working Group, RFC 1393: Post Office Protocol - Version 3. Available: <https://www.ietf.org/rfc/rfc1393.txt>
- [3] Network Working Group, RFC 3501: INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1H. Available <https://tools.ietf.org/html/rfc3501>
- [4] Autodiscover for Exchange. Available: [https://msdn.microsoft.com/en-us/library/office/jj900169\(v=exchg.150\).aspx](https://msdn.microsoft.com/en-us/library/office/jj900169(v=exchg.150).aspx)
- [5] [MS-OXDISCO]: Autodiscover HTTP Service Protocol. Available: [http://interoperability.blob.core.windows.net/files/MS-OXDISCO/\[MS-OXDISCO\]-160613.pdf](http://interoperability.blob.core.windows.net/files/MS-OXDISCO/[MS-OXDISCO]-160613.pdf)
- [6] How to generate a prioritized list of Autodiscover endpoints. Available: [https://msdn.microsoft.com/en-us/library/office/dn467397\(v=exchg.150\).aspx](https://msdn.microsoft.com/en-us/library/office/dn467397(v=exchg.150).aspx)
- [7] Network Working Group, RFC 5321: Simple Mail Transfer Protocol. Available: <https://tools.ietf.org/html/rfc5321>
- [8] Network Working Group, RFC 5322: Internet Message Format. Available: <https://tools.ietf.org/html/rfc5322>
- [9] Internet Engineering Task Force, RFC 6531: SMTP Extension for Internationalized Email. Available: <https://tools.ietf.org/html/rfc6531>
- [10] Internet Engineering Task Force, RFC 6532: Internationalized Email Headers. Available: <https://tools.ietf.org/html/rfc6532>
- [11] Public suffix list. Available: <https://publicsuffix.org>
- [12] IANA Root Zone Database. Available: <https://www.iana.org/domains/root/db>
- [13] Security Maintenance Release, January 2017. Available: <http://security.samsungmobile.com/smrupdate.html#SMR-JAN-2017>
- [14] About the security content of iOS 10.3. Available: <https://support.apple.com/en-sg/HT207617>
- [15] Max Goncharov, badWPAD. Available: <https://www.trendmicro.co.uk/media/misc/wp-badwpad.pdf>