Telco SMTP -> SMS/MMS Crypto?

Champ Clark (champ@softwink.com) AKA "Da Beave"

[Champ Clark (AKA - "Da Beave") is a security researcher and developer at Softwink, Inc.]

It's a lazy Sunday. Rather than being down at the beach fishing, I'm at the Jacksonville Beach Softwink, Inc office contemplating SMTP to SMS gateways offered by various cell phone Telecomunications providers. There's a reason for this....

I'm one of the developers for the PoC (Proof of Concept) code of "pam-obc" (PAM == Pluggable Authentication Modules, OBC == Out-of-Band Challenge). PAM provides the back-end authentication schemes into an API (Application Programming Interface). When you SSH (Secure Shell) into a remote system, PAM is likely providing the functionality to determine if you entered a correct password. PAM is also used in the Unix world to provide low-level authentication schemes for commands like "su" and "sudo".

The point of "pam-obc" is to send an "out-of-band" challenge. In other words, it sends a "token" not tied or related to your current communications or session (for example - SSH) that acts as a one-time token for further identification. The token is randomly generated string of characters and can only be used for that session. "pam-obc's" only job is to generate this random token and compute whether the string the user sends as the challenge/response is "correct" or "incorrect".

While "pam-obc" works, I refer to it as PoC as it's something that is still being developed. For more information on "pam-obc", see the links at the end of this article.

Here's a simple example using the sudo command with the "pam-obc" module enabled. At a Unix command prompt, you'd enter:

```
$ sudo vi /etc/passwd
```

The first thing you'd be prompted for is:

```
Out-of-Band Challenge:
```

This means that "pam-obc" has generated a random, one-time use string and has sent it to you "out of band". "pam-obc" doesn't really "care" how the token gets sent. That's up to the administrator. In other words, it is up to the administrator to determine how securely or insecurely the challenge is sent to you. Let's say that on this system, the challenge is sent via Jabber/XMPP over SSL. In this case, you would receive an instant message with a "random string" (token). You would then copy this random string and paste it into the prompt above and hit "enter". You would then be prompted for:

```
Password:
```

You enter your password as normal. The point is that only you, through alternate communication means, such as Jabber/XMPP over SSL, should ever receive that random string. If either the challenge or password is entered incorrectly, access to the command and/or system is denied.

Pretty simple, eh? It also serves another function. Let's say you have "pam-obc" working with OpenSSH. It's 3:00 AM, and you're sound asleep. Suddenly, on your laptop, a challenge "token" pops up. It obviously wasn't you and now you know that someone or something is trying to access your account.

It isn't meant as a replacement SSH shared keys or anything of that sort. It is simply another means by which to verify that you are who you claim to be. It also ensures that even if your "password" is stolen, without access to the token, the attacker won't be able to gain access.

There are multiple ways to configure "pam-obc", but believe it or not, that's not the point of this article.

While chatting with a user about "pam-obc", he mentioned this nifty way he was sending the challenge tokens; "I have a modem hooked up to a computer and it pages me with the token". I replied, "Like an old fashioned pager? Like from the 1990's?". He said, "Yep!".

But, there is a problem with this. Radio hobbyists have known how to intercept pager traffic for many years now. A recent example is the September 11th, 2001 Wikileaks release of pager traffic. This was likely done by an amateur radio enthusiast. For more technical information about pager interception, check out my friend NYCMIKE's talk at Defcon, "The World of Pager Sniffing/Interception: More Activity than one may suspect ".

I pointed this out to the user, after which he came up with an idea; "I'll just send it to my cell phone via my provider's SMTP to SMS gateway". Although that would work, there is a good chance that the SMTP session will be sent "in the clear" (non-encrypted) and open to interception. That got me thinking; I wondered if Telco provided SMTP to SMS/MMS gateways provide TLS (Transport Layer Security - encryption). While still not 100% ideal for various other reasons, it still got me thinking.

When I mention a Telco provided SMTP to SMS/MMS gateway, I'm referring to the ability to send an e-mail to a specially crafted e-mail address that shows up via SMS on your cell phone. For example, you might own a cell phone tied to the provider "AT&T". If you send an e-mail to "yourphonenumber@txt.att.net", it will pop up on your phone. Sometimes network administrators use this functionality to report possible problems. For example, when a computer "monitoring" network connectivity is unable to "ping" a server it will fire off a SMS stating, "Server XYZ is unreachable!".

This article is not referring to corporate-level SMS/MMS systems such as those used by the banking industry, who appear to think that this is a nice way to let you know your checking account balance. Other applications use it to "verify who you are" (Google voice, for example). Since I'm thinking of American providers we won't even go into the 26C3 (CCC) A5/1 GSM encryption problems. Even though it was a great talk to see in Germany! [26C3 - GSM SRSLY?]

However, if you're a network administrators, it is unlikely that you would spend the "big bucks" for such a system to simply tell you that a server is "unreachable". This led me to another thought: there are currently applications available that are likely using Telco-provided SMTP to SMS gateways rather than corporate level SMS/MMS servers. If you are a developer using these types of back-end Telco-provided services, pay attention.

A simple solution would be to enable TLS to the Telco SMTP to SMS gateways. Simple enough - right? But who says they support TLS? That became my big question today.

Using Google, I was able to compile a list of about 160 different methods to send SMS or MMS messages via SMTP.

The "methods" I'm referring to are similar to the example above; "younumber@txt.att.net" or "yournumber@sms.t-mobile.at". Among these 160 or so "methods" of sending an SMTP based SMS or MMS message, I tested about 60 companies. The reason for this is that a company might have multiple methods for sending an SMS/MMS. For example, you can send an SMS via SMTP via "txt.att.net", but you can also send a message to the same phone via "cingular.com". The reason is that AT&T bought-out Cingular but the service remains in place for legacy reasons.

Once I had my list of ways to send an e-mail to SMS/MMS, I built a list of MX records corresponding to the provider's e-mail address. For example, if you're a "Boost Mobile" user, you'd send your e-mail to "yournumber@boostmobile.com". Your e-mail would then be sent to the mail server (MX records) handling mail for the "boostmobile.com" domain. In this case, it would be mail.global.frontbridge.com. This resolves to a mail servers IP address (65.55.88.22), which is a Microsoft hosted service.

With a list of IP addresses in hand, it was time to write a small routine to test if the remote SMTP service supports TLS/encryption on port 25. This is simple enough using tools provided by OpenSSL. To test from the Unix/Linux command prompt, we simply run:

```
$ openssl s_client -starttls smtp -connect 1.2.3.4:25
```

(Replace 1.2.3.4 with the remote target).

Since we're checking for TLS support, I also scanned for port 465 (SSL over SMTP). To test for that, the command would be:

```
$ openssl s_client -connect 1.2.3.4:465
```

My thought was that while TLS on port 25 might not be supported, perhaps it might be supported on port 465. That seemed even less likely, and even less useable.

If a successful TLS/SSL connection is established, OpenSSL will dump the details about the session (encryption supported, server certificate, certificate chain). If it fails, you will see something like:

```
CONNECTED(00000003)
didn't find start tls in server response, try anyway...
19378:error:140770FC:SSL routines:SSL23_GET_SERVER_HELLO:unknown protocol:s23_clnt.c:601:
```

Results:

I wasn't terribly surprised by the results. Of the 160 ways to send an SMTP to SMS/MMS message, 8 providers supported TLS/SSL. In actuality, the only reason these some of these supported TLS was because they used a third party to handle e-mails for junk and spam filtering.

Telco SMTP -> SMS/MMS providers that support SMTP TLS or SMTP over SSL. Click the links to see

SSL/TLS information.

Provider	Country	E-mail	Tested MX/IP	TCP Port	Notes
Virgin Mobile	U.S.A	bills.com	##.psmtp.com (various)	25	Hosted/Managed SMTP services by Postini. To many to check
i-wireless (Sprint PCS)	.	iwirelesshometext.com	mx5.telcordia.com (192.4.253.105), mx4.telcordia.com (192.4.253.106), mx7.telcordia.com (192.4.253.110)	25	*[none]*
Iridium	(Satellite Comms)	msg.iridium.com	istmlb.iridium.com (208.25.12.85)	25	*[none]*
T-Mobile	Austria	sms.t-mobile.at	mail1.t-mobile.at (213.162.65.1), mail2.t-mobile.at (213.162.65.2), web-mail.t-mobile.at (213.162.64.24)	25	*[none]*
Cingular (AT&T) Postpaid)	U.S.A	cingular.com	cluster##.us.messagelabs.com (various)	25	*[none]*
T-Mobile	Germany	t-d1-sms.de	thor.ic3s.de (62.159.211.165), mail.t-d1-sms.de (62.159.211.165)	25	*[none]*
Boost Mobile	U.S.A.	boostmobile.com	mail.global.sprint.com (65.55.88.22), mail.global.sprint.com (216.32.180.22) No TSL	25	Boostmobile's MX/IP address resolves to "mail.global.frontbridge.com". This is Microsoft's Managed Exchange services. One address support TLS, the other doesn't.
O2	Germany	o2online.de	mail.o2online.de (82.113.101.173)	25	*[none]*
O2	Germany	o2online.de	mail.o2online.de (82.113.101.173)	465	*Would this port really work? Couldn't test reliably.*
Chennai RPG Cellular	India	rpgmail.net	rpgmail.net (74.52.34.114)	465	*Would this port really work? Couldn't test reliably.*

Key : # == provider with many MX records. For example, Virgin Mobile uses psmtp.com (Postini, a filtering service owned by Google since 2007). This means there are to many records to check.

Conclusion:

It appears that about .05% of Telco SMS/MMS gateways support TLS/SSL SMTP. Again, I'm not terribly surprised. Is this a 100% accurate representation of the Telco industries public SMTP to SMS/MMS gateways? It's probably not.

A friend pointed out to me, "who cares, nobody ever sends anything important via SMS anyways". This isn't true. Think of a network administrator or an application developer that sees a simple and easy way to send what might need to be "secure" SMS's. This obviously isn't the way to go.

TLS isn't complex or difficult to set up, so why don't the Telco based SMS/MMS gateways support it? Well, it is likely that they don't care and "it's your problem".

The Solution:

For small-time applications for which you want this type of functionality, but keep it as secure as possible and without the cost overhead of a "corporate level" SMS/MMS server/services, there is hope.

Using an old cell phone, such as a Nokia handset, a data cable, a Linux box and "gnokii" you can build a small SMS server that'll meet your security needs better than relying on Telco-based SMTP to SMS/MMS gateways.

As stated at the beginning of the article, this all came up because we wanted to send a secure "token" to a cell phone. Seeing that Telco gateways are likely not going to work for security and other reasons (the Internet is down!), setting up a small Linux server to handle sending out "challenge tokens" is trivial enough.

Basically the idea is to connect a cell phone via USB to a server. The server can communicate directly with the cell phone via the old AT command set (Hayes command set, like with old modems). Using this command set, you can send out SMS message and consequently by-passing the need for SMTP gateways. This means there is nothing to capture across a network (the Internet) and you can control the security of "how" the message gets out. The Linux

box talks directly to the cell phone, the cell phone sends it out over the cell network (encrypted). There are several "how-to" articles to get you started on this type of project. It's quite easy to accomplish. Perhaps I'll write an article on that next.

In the mean time, here's a few helpful links:

"pam-obc" (Out-of-Band Challenge) module
"pam-obc" PoC was orginally created by Paul Sery of Sandia. Champ Clark, of Softwink, Inc. joined the project later and contributed code and autotool support. For more information see:

pam-obc SourceForge page

For general questions about "pam-obc", you'll probably want to post/read in the "pam-obc" mailing list. It's a very low volume mailing list for responding to general questions.

Linux SMTP to SMS guides to get you started
Linux - SMTP to SMS gatway
Gnokii getting started
Gnokii - User Guide
(How to Use a Linux PC to Send and Receive SMS Messages (Non-Developer's Perspective)

Champ Clark (AKA - "Da Beave") is a security researcher/developer for Softwink, Inc". He's also a founding member of The Deathrow OpenVMS cluster which is used for security research on the OpenVMS operating system. He's also a founding member of Telephreak.org, a Asterisk/VoIP "Hacking" group. He's an author of Asterisk Hacking from Syngress Publishing and has been featured in Syngress's InfoSecurity Threat Analysis 2008 (Syngress Publishing). He is the author of iWar (Intelligent Wardialer), X.25 pen-testing software and various other security related utilties. He is a regular speaker at "cons" (Defcon/HOPE/26C3/etc) and enjoys fishing. Special thanks to Kathrin Ritter @ Softwink for assisting with this article.