



207-1001 Kingsway
Vancouver, BC V5V 3C7
+44.7924.568.318
www.networktestlabs.com

Case Study on Online Casino's

*By NTL Associates
March - 19 - 2011*

Contents

Introduction	3
Online Casino types	4 - 5
The importance of a TSD	5
Gaming Regulators	6 - 7
Play Smart	7 - 8
Basic Concepts – Online Poker	8
Online Casino Breach	10 - 14
Conclusion	14

Network Test Labs Inc is Canada's leading security consulting team with a comprehensive security portfolio delivered to leading industry operations, software suppliers & manufactures and jurisdictional regulators to verify compliance with stringent regulatory requirements, supplier technical specifications, and industry best-practices.

Our vision is to provide quality, fairly priced professional services within the framework of industry leading practices and cutting edge security testing methodologies.

NTL's goal is to ensure that vendors and operators get the adequate protection and proven industry expertise.

Our experienced security consultants, test engineers, RNG testers, provide compliance testing and information consulting services in the following areas:

- iGaming
- Compliance Testing
- Consultation and Advisory Services

Introduction

Network Test Labs (NTL) was instructed by the regulators in partnership with an Accredited Test Facility (ATF) to perform a detailed security assessment of the Internet Gaming System (IGS). As a result of carrying out the security assessment, a number of security issues were addressed to meet compliance while enhancing security risk mitigation, and optimizing IT and network operations through documentation review, security architecture review, source code review, access controls review, review of cryptographic controls, penetration testing, web application security assessment of the individual operator secure gaming environment.

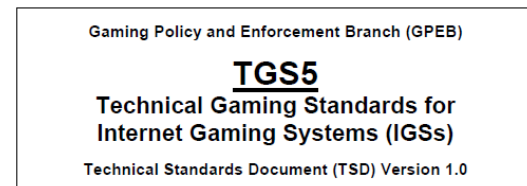
NTL researched, developed and updated the Technical Standards Document (TSD) for easy acceptance with the goal of attracting new online gaming operators to the regulators jurisdiction. NTL drafted a high-level comprehensive framework based on the harmonization of the following TSD's published by various bodies, regulators and other institutions.

Additional Controls for Compliance Evaluation also included the following ISS standards:

- AS / NZS 4360:2004 Risk Management,
- Control Objectives for Information and Related Technology (COBIT),
- Open Source Security Testing Methodology Manual (OSSTMM), and
- International Standards Organization (ISO) 17799 Standard – Information Technology – Code of Practice for Information Security Management.

As a result, this Case Study was created to better inform all parties involved about the current status of online operators and the effectiveness of the technical security standards.

Online Gaming Regulators Technical Standards Documents (TSD) used for this Case Study:



Online casino types

Online casinos can be divided into two groups based on their interface, either web-based and download-only casinos. Some casinos offer both interfaces.

Some casinos offer 'Live gaming', either exclusively, or as part of a wider online casino offering. In live online casinos, popular casino games such as roulette and blackjack are dealt by real dealers in casino studios, in an attempt to convey more of the atmosphere of a physical casino. Player actions (including chat) such as 'hit' in the game of blackjack may be transmitted to the dealer; in some online casinos more than one player may 'sit' at a particular 'seat' on the table, and in this case there is no interactivity between player and dealer, the question of which player(s) requested the extra card that the dealer dealt and which chose to 'stand' will be handled by the software. It is usual for players to be able to observe a video feed of the action, and equally common for players to opt to turn it off, if they lack the bandwidth - as the cards are read by OCR (Optical Character Recognition) and other technology, the video feed is only ever a visual cue.

Web-based online casinos

Web-based online casinos are websites where users may play casino games without downloading software to the local computer. Games are mainly represented in the browser plugins Macromedia Flash, Macromedia Shockwave, or Java and require browser support for these plugins. Also, bandwidth is needed since all graphics, sounds and animations are loaded through the web via the plugin. Some online casinos also allow gameplay through a plain HTML interface. Apple devices such as Ipad, Ipad and Iphone cannot play Flash games as the technology is not supported.

Download-based online casinos

Download-based online casinos require the download of the software client in order to

play and wager on the casino games offered. The online casino software connects to the casino service provider and handles contact without browser support. Download-based online casinos generally run faster than web-based online casinos since the graphics and sound programs are located within the software client, rather than having to be loaded from the Internet. On the other hand, the initial download and installation of a download-based online casino client does take time. As with any download from the Internet, the risk of the program containing malware does exist. Still the graphics and sounds at download-based online casinos are most of the time much better than those at web-based online casinos. Moreover, in some online casinos game effects in download-based games run more smoothly and much faster than in web-based online casinos (which is noticeable in spinning slots and dealing cards in blackjack)

Fraudulent online casino operator behavior

Fraudulent behaviour on the part of online casinos has been documented. The most commonly reported behaviours are refusal to pay withdrawals or cheating software with rigged payouts. An online casino with multiple confirmed cases of fraudulent behaviour is often called a **rogue casino** by the online casino player community.

One commonly reported behaviour related to refusal to pay withdrawals is the refusal to pay withdrawals promptly. A rogue casino may intentionally delay a withdrawal in hopes that the player will continue gambling with the money in the account and lose it all back. Other rogue online casinos attempt to not pay their customers by retroactively applying unfair terms to their house rules.

Some casino software has been mathematically proven to cheat, such as Elka System/Oyster Gaming and Casino Bar. Screen shots from the back office of an older brand of software indicated the odds could be adjusted by the operator.

Some casinos were proven to give higher winning probability when players play in fun

mode comparing to playing for real money, In order to attract players to register and play for real money.

Many casino gambling portals and player forums maintain blacklists of rogue casinos. While some carry more authority than others, most blacklists constitute individual webmaster and player opinions rather than anything official from any type of regulating body.

Gambling Commissions base their Technical Standards Document on the relevant sections of ISO/EIC 27001:2005, COBIT, NIST and other security requirements.

The importance of a Technical Standards Documents is:

- To eliminate subjective criteria in analyzing and certifying IGS operations
- To test the criteria that impacts the credibility and integrity of the IGS operations from both the Revenue Collection and Player’s point of view
- To create a TSD that will help ensure the Internet Gaming System (IGS) operating in the live environment are fair, honest, secure, auditable and able to operate correctly
- To construct a TSD that can be easily changed or modified to allow for new technology or functionality.

To the top right is a Regulatory Compliance Mapping Chart of the different standards required for the individual Gaming Jurisdictions.

The Alderney Gambling Controls Commission below in “blue” has the most stringent security requirements followed by the AAMS Italian requirements in “green”

REQUIREMENTS	INTERACTIVE GAMING STANDARDS	AGCC	UK	Kahnawake	AAMS Italian	ISO 27002
1	REQUIREMENTS FOR REQUESTING THE CERTIFICATION OF GAMING PLATFORM	X	X	X	X	X
2	REQUIREMENTS FOR REQUESTING THE CERTIFICATION IN JOINT VENTURE	X				
3	GAMING PLATFORM REQUIREMENTS	X	X	X	X	X
4	DOCUMENTATION REQUIREMENTS	X				
5	SOURCE CODE REQUIREMENTS	X	X	X	X	X
6	REMOTE ACCESS CONTROL	X	X	X	X	X
7	GAME EMULATION CAPACITY	X	X	X	X	X
8	INFORMATION SYSTEM SECURITY CONTROLS	X	X	X	X	X
9	IGS PLANNING	X				
10	PLAYER REGISTRATION	X				
11	ANTI-MONEY LAUNDERING REQUIREMENTS	X	X	X	X	X
12	PRIVACY	X	X	X	X	X
13	PROTECTION OF PLAYERS	X	X	X	X	X
14	RNG REQUIREMENTS	X	X	X	X	X
15	TECHNICAL REQUIREMENTS	X	X	X	X	
16	APPENDIX A: GAMBLING OPTION REQUIREMENTS	X			X	
17	APPENDIX B: ROULETTE GAME REQUIREMENTS	X			X	
18	APPENDIX C: DICE GAME REQUIREMENTS	X			X	
19	APPENDIX D: GENERAL CARD GAME REQUIREMENTS	X			X	
20	APPENDIX E: BLACKJACK GAME	X			X	

Gaming Regulators

Online Gaming Regulators want to provide protection for players who choose to participate in the online gaming offered by the Regulators permit holders. The Regulators use a variety of means to ensure that:

- interactive gaming is conducted responsibly, fairly and honestly; and
- the operators of interactive games treat players fairly; that they pay winners promptly and that all information related to player accounts is held in the strictest confidence.
- gaming sites must provide a mechanism by which a player has the ability to limit his or her play - including setting the limit to zero;
- a person who is close to a player (eg. family member) may apply to the Commission for an order prohibiting gaming sites from accepting bets or wagers from that player;
- the relationship between the operator, a gaming site and players must be contractual and must not contain terms that are unfair or unreasonable;

- gaming sites must take adequate measures to protect the confidentiality of player information; and
- rules of play must be clearly posted in English and any other such language as may be appropriate.

The Gaming Regulations also provide a number of dispute resolution mechanisms to address player complaints - including the possibility of third party arbitration. In fact, most Regulators staff includes a full time Dispute Resolution Officer to ensure that all player complaints are addressed in a timely way.

Gaming Regulators try to ensure that their regulatory and supervisory approach meets the very highest of international standards.

Below, you will find information on some of the Gaming Commissions that we feel excel above the others.

1. The Alderney Gambling Control Commission

The Alderney Gambling Control Commission (AGCC) was established in May 2000. The Commission, consisting of the Chairman and three members, is independent and non-political, and regulates eGambling on behalf of the States of Alderney.

Alderney, the third largest of the Channel Islands and one of the constitutive islands of the Bailiwick of Guernsey, is approximately 8 miles from France and 60 miles from mainland Britain. It has its own government, legislature and company laws and operates with the same modern banking, insurance and investment laws as Guernsey. The Guernsey Financial Services Commission is a statutory body responsible for ensuring that the finance industry on Guernsey and Alderney is well regulated. Alderney is subject to the provisions of Guernsey taxation laws.

Alderney licensees are by law permitted to take advantage of the modern hosting facilities and reliable telecommunication networks on both Alderney and Guernsey. For international network links, Alderney and Guernsey's telecommunication network offers reliable and high capacity links to the UK, Europe, the USA and Asia.

2. The State Monopoly Administration (The Italian Gaming Commission AAMS)

Control of the game is reserved to the state, which is exercised, since 2000, with AAMS. The Administration will award a concession to private operation of the game, which make the investments necessary and appropriate business tools and manage the game according to rules, compliance with which is constantly monitored, placed to protect the public. The Italian legislation will give the game away in the character of entertainment, socialization, and nice use of leisure time, differentiating it significantly from other games, based primarily on individual behavior and distance, both physical and temporal, between the time of the game and that of winning.

The Monopoly Administration of State is the guarantor of legality and security of equipment and devices for fun and entertainment to ensure the transparency of the game.

Clear rules, maximum transparency and security for all, this is the message AAMS wants to convey with the trademark "Fair Play" and through the game safe, AAMS supports the sport.

3. Gaming Policy and Enforcement Branch (Canada) GPEB

Gambling in Canada is an illegal activity except where it is made legal through provisions set out in the *Criminal Code of Canada* and as sanctioned under the authority of each of the provinces. The Government of Canada has minimal involvement beyond *Criminal Code* prohibitions and permissions. In 1985 an agreement between federal and provincial governments established annual provincial payments to the federal government to

assure only provinces can authorize gambling.

No two provincial regulatory and operational regimes are the same. Different regimes and models are in place across the country. Different games and lottery schemes are permitted or prohibited depending on the province. As an example, casinos may be commercial, charitable, owned/operated by government, and/or operated by private companies who are under contract to provincial gaming authorities (or a combination of these). All provinces license charities under their own regulations and permitted schemes as per Section 207(1) (b). All provinces participate in national lottery schemes – but provincial/regional gambling authorities permit and/or offer a variety of other games of chance.

Unlike other Online Gaming sites, all the funds generated go to the government. All the operators in Canada for example OLG is owned and operated by the government, therefore it is safe, secure, and trusted all around by its players.

Play Smart

What should I look for when I choose an interactive casino or sports book?

Many of the features that dictate your preference for a site are subjective and relate to the games on offer. We offer no guidance on these. However, in our view the following components should be present on a well-operated site.

- The predicted return to player or “RTP”.
- A clear identification of the regulator and their jurisdiction, probably with a logo
- Customer service contact data
- Terms and conditions for all play; an outline of the disputes process
- Secure payments process

Is my deposit safe?

Licenses’ “suitability” is a matter of careful investigation and most jurisdictions pride themselves on the exclusion of criminal influence. However casino players’ money may also be at risk from business failure. Most Commissions require its licensees to advise players, clearly, of the separate protection status of the player fund deposits in the casino, within three categories:

- “Not segregated” – this means that the operator has not segregated player funds from the company’s own funds. Player funds are therefore at risk in the event of a business failure.
- “Segregated” – this means that the player funds are held in one or more accounts separate from the business’s own funds and that the interest of the players in the account/s is noted in the title of the account. This may provide protection in the event of a business failure.
- “Segregated and protected” – this means that the player funds have been segregated and the Commission has been given reasonable assurance that the segregation is likely to be effective in protecting the player funds in the event of a business failure.

How do I make a complaint against a licensee?

A complaint about a gambling transaction should, in the first instance, be directed to the operator concerned. This should be done in writing (email is acceptable) and you should receive a written reply.

What happens if the operator has not dealt with my complaint to my satisfaction?

If the player is not satisfied with the resolution offered by the operator, the complaint and all relevant correspondence can be forwarded to the Gaming Commission for arbitration. The Commission will acknowledge your complaint

- After that: The operator will update the player on progress after we have been in communication with the operator.
- The operator will write to the player with a “preliminary determination” of the player’s complaint, which will identify the operators opinion of the player’s complaint and a proposed resolution.

Please note that the Commission does not provide advice to individual consumers.

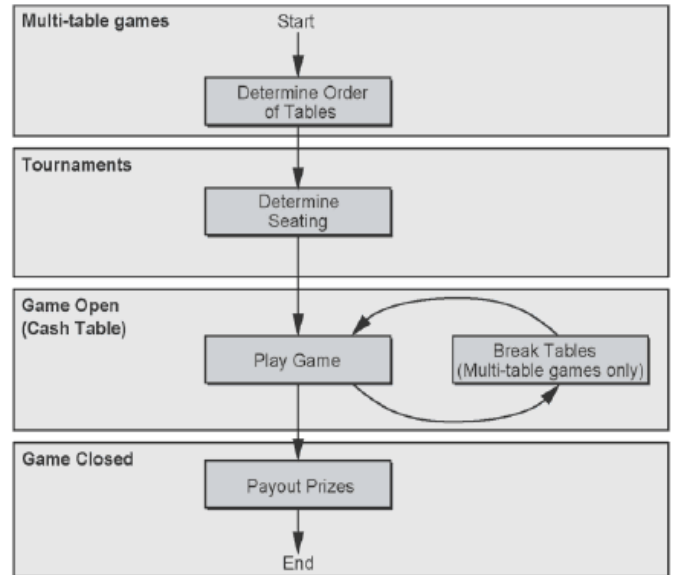
How do I deal with a gambling problem?

All licensees are expected to have procedures in place to identify and support anybody who may be a problem gambler or is worried about somebody else’s gambling. Many of them work with organisations such as Gamcare who can offer advice or support. Please click below for links to relevant organisations.

www.gamcare.org.uk
www.gamblingtherapy.org

Basic Concepts – Online Poker

The number #1 online game still seems to be Online Poker. The flow of events in a poker game can basically be described according to the figure below. The core of the game is the game round and for tournaments, additional events take place to the opening of the game.



Determine Order of Tables: For performance reasons, the starting time for the game action at the tables in a multi-table game is randomly decided so that playing does not start at the same time at all tables.

Determine Seating: In tournaments the seating at the poker table is determined by randomisation.

Play Game: This is where the actual game takes place. Cards are dealt and players make their decisions according to the value of their cards.

Break Tables: There are never more than the minimum number of tables required to seat all players. As the number of players reduces, the number of tables also reduce. A table is removed as soon as the remaining players can all be seated around the remaining table.

Breach-Investigation-Remediation

The following is about an incident that occurred a few months ago involving an important Online Gaming Operator. Network Test Labs (NTL) was called in to assess the situation and remediate the issues.

Breach

Certain customers logged in and playing on the site had noticed that at some point their own session had been exchanged for that of another player, and they could play games as that other person. This issue was reported by at least 4 customers and the site was subsequently taken down.

Investigation

NTL in a partnership with an ATF performed the following tasks to identify and isolate the issues.

Initial Steps

The initial analysis was undertaken by analysing the log files produced by the applications within the relevant infrastructure, including the Apache web server, the Tomcat Servlet container running the game server, and the Portal and Login Navigation applications.

This investigation showed several instances where an authentication cookie sent in a request from one IP address, would then subsequently be sent in a request originating from a different IP address. This authentication token is created when the customer logs into the site, and used to identify that customer on subsequent requests. Since this cookie is not restricted by IP address, a person in possession of another's login cookie can therefore act as the other person.

Code Analysis

The initial concern was that one or more of the applications were either

- setting an incorrect cookie value in an invalid response

- playing out an incorrect response

A code analysis of all the applications that included an authentication cookie in a response to a customer was then performed, looking for areas where either the cookie value or the response may have been cached, persisted between requests, or involved non-synchronised access to static or shared data. This analysis did not provide any likely candidates for this problem.

Log Analysis

Further analysis of the log files highlighted a number of areas of interest.

Simulation and Isolation

The log analysis detailed above suggested that the issue lay either with the game server, or with the communication between the game client and the game server.

An initial attempt to reproduce the issue on the staging environment through load testing failed; however there wasn't the same level of connectivity issue experienced in the live environment. The load tests run on staging were run at a higher level of debug information, and yielded a further error message in apache

This suggested that a possible cause of the problem might be located within the mod_proxy module in Apache, and related to proxy requests to Tomcat that timed out.

Test Client

In order to isolate the area responsible, and to remove the game server application code from the equation, a test harness was created that consisted of a Java client that sent requests to a custom Java Servlet installed within the Tomcat server instance on the staging environment.

The client creates an XML request that contains a randomly created request identifier, a defined length of padding, and a specified server delay. This delay is selected randomly, but is weighted so that the majority of delays will be relatively small. The Servlet will simply echo the identifier

and padding back to the client after the delay specified in the request. The client will then compare the identifier sent to the one received, and error if they do not match.

Mod Proxy Investigation

Architecture

A three tier architecture consisting of web, application and DB tiers was used. All web requests from clients (which includes both web pages and Flash game activities) are handled by the Apache httpd server on the web tier. A front-end load balancer handles SSL termination, so all requests seen by Apache are HTTP requests.

Web requests for the game server are forwarded to the Tomcat server on the application tier. The forwarding is done by the Apache mod_proxy module (which is part of the Apache httpd distribution). The protocol used by mod_proxy_http to talk to Tomcat is HTTP - mod_proxy is acting as an HTTP proxy and is effectively itself an HTTP client of Tomcat.

Connections

To improve performance, the latest version of the HTTP protocol (HTTP/1.1) allows the network connection to be re-used for multiple requests, provided both the client and the server agree to do so. This re-use of the connection is often known as "HTTP KeepAlive" since instead of closing the network connection after sending a response, the server "keeps it alive" in case the client wants to send more requests on the same connection. HTTP connections that have been kept alive after the end of the first response are sometimes known as "persistent connections". In HTTP/1.1, the client is allowed to send multiple requests down a connection without waiting for a response to each request. This is often known as "HTTP pipelining". It relies on the server sending back the responses in exactly the same order as the requests were sent since there is no mechanism for specifying which response was for which request.

Apache Configuration

The Client was using version 2.2.3-43.el5 of the Apache httpd server, which was supplied by RedHat. This version is essentially equivalent to version 2.2.3 of the mainline (i.e. Apachesupplied) server with patches applied by RedHat to bring it up to version 2.2.9.

Apache is configured in "prefork" MPM mode with keepalives enabled for clients that support them. In this configuration, Apache uses a dynamic pool of child processes to handle incoming HTTP connections, with one child process per client connection. Since keepalives are enabled, a client can make more than one request down the same connection (until the keepalive timeout expires). After a client connection has closed, the child process is returned to the pool for re-use for the next connection from any client.

Requests for the /Servlet URL (i.e. request from the games client to the game server) will be handled by the Apache mod_proxy module which will make HTTP connections to the backend server on behalf of the client. The mod_proxy module has been supplied by RedHat built-in to Apache, and is configured in the default manner, which includes the use of persistent HTTP connections to the backend servers.

The Apache and mod_proxy timeouts are configured such that if a request to a backend server takes more than two minutes, mod_proxy will return a "502 Bad Gateway" error to the client on the assumption that there is a problem with the backend server.

In this case, the backend server is Tomcat. Version 6.20 of Tomcat is installed and configured to use the HTTP/1.1 protocol with keepalives enabled (where possible) and no limit on how long a request may take.

To improve performance, mod_proxy will (where possible) maintain a dynamic pool of backend connections to Tomcat. If, when processing a request, mod_proxy realises a connection to Tomcat in its pool, it will take that connection out of the pool and use it

rather than making a new connection. After using a connection to send a request to Tomcat and receive a response, mod_proxy_http will normally return that connection to the pool for future re-use.

Apache is configured such that each child process has its own independent copy of mod_proxy within it, which means that each child process has its own backend connection pool, configured with up to one connection per pool. There is no limit to the length of time mod_proxy will keep open an idle connection in the pool.

Source Code Analysis

The two most relevant functions in mod_proxy are proxy_http_handler and ap_proxy_http_process_response in the modules/proxy/mod_proxy_http.c file. The first function, proxy_http_handler, essentially contains the following logic:

```
proxy_http_handler () {
<< Get connection from pool. >>
<< Send request to backend. >>
<< Call ap_proxy_http_process_response to
handle the response (by sending it to
the client). >>
<< If something bad happened
(i.e.ap_proxy_http_process_response
returned a non-OK code)
then mark
the connection as bad (which will cause it to
be closed instead of returned to the pool).
>>
}
```

The other function, ap_proxy_http_process_response, is rather more complex, but we need only look at what happens if the backend fails (or takes too long) to respond. In version 2.2.2 of Apache and earlier, the logic was very simple, as can be seen below.

```
// v2.2.2 ap_proxy_http_process_response
() {
...
<< Try to read status line from backend. >>
len = ap_getline(buffer, sizeof(buffer), rp, 0);
...
if (len <= 0) {
<< If we couldn't read it (for any reason), then
log it and return an error. >>
```

```
ap_log_error(APLOG_MARK,
APLOG_ERR, 0, r, "proxy: error reading
status line from remote
server %s", backend->hostname);
return ap_proxyerror(r,
HTTP_BAD_GATEWAY,"Error reading from
remote server");
}
<< Otherwise, send response to client and
return OK. >>
...
return OK;
}
```

Version 2.2.2 behaves correctly; if the backend server takes too long, then ap_proxy_http_process_response will return an error to proxy_http_handler to indicate that the backend connection should be closed. However, in the next version, ap_proxy_http_process_response has some extra logic added in certain circumstances when an error occurs getting the response from the backend, as shown below:

```
// v2.2.3 - v2.2.9
ap_proxy_http_process_response () { ...
<< Try to read status line from backend. >>
rc = ap_proxygetline(tmp_bb, buffer,
sizeof(buffer), rp, 0, &len);
...
if (len <= 0) {
<< If we couldn't read it (for any reason), then
log it first. >>
ap_log_error(APLOG_MARK,
APLOG_ERR, rc, r, "proxy: error reading
status line from
remote " "server %s", backend->hostname);
<< Confusingly, c->keepalives here is the
number of times the connection has been
kept
alive after a response. So c-keepalives is
essentially a counter of the number of
responses
sent on this connection, and this if statement
really says "If this WASN'T the first request
from the client then: " >>
if (r->proxyreq == PROXYREQ_REVERSE
&& c->keepalives) {
<< Log error again at debug level. >>
...
<< Send 502 error to client (in slightly odd
way). >>
...
<< Close connection to client. >>
```

```

...
<< This is the bug: the code returns OK
which tells proxy_http_handler that the
backend connection can be returned to the
pool. >>
/* Need to return OK to avoid sending an
error message */
return OK;
}
...
}

```

This change introduced in v2.2.3 is the bug that causes misdirected responses - it's not safe to re-use a connection that timed out, since another response might appear on it! Presumably in response to a bug report, an attempt was made to fix this issue in v2.2.10 - note how the test has changed to include the `rc != APR_TIMEUP` condition:

```

//                                v2.2.10-v2.2.15
ap_proxy_http_process_response () { ...
<< Try to read status line from backend. >>
rc = ap_proxygetline(tmp_bb, buffer,
sizeof(buffer), rp, 0, &len);
...
if (len <= 0) {
<< If we couldn't read it (for any reason), then
log it first. >>
ap_log_error(APLOG_MARK,
APLOG_ERR, rc, r,"proxy: error reading
status line from remote "
"server %s", backend->hostname);
<< If this wasn't the first request from the
client, AND this backend error wasn't a
timeout, then: >>
if (r->proxyreq == PROXYREQ_REVERSE
&& c->keepalives && rc != APR_TIMEUP) {
<< Log error again at debug level. >>
...
<< Send 502 error to client (in slightly odd
way). >>
...
<< Close connection to client. >>
...
<< This is the bug: the code returns OK
which tells proxy_http_handler that the
backend connection can be returned to the
pool. >>
/* Need to return OK to avoid sending an
error message */
return OK;
}

```

```

}
...

```

This fixed the error (at least in timeout scenarios) on UNIX-like systems, including Linux, but apparently not on Windows and OS/2, since the timeout condition should have been written as differently to work correctly on those platforms. This is promised to be fixed on all platforms in the not-yet-released version 2.2.16, which will also contain a extra line to force the backend connection to be closed before returning OK, hopefully addressing the defect in case the error was not a timeout.

Remediation

Solutions

No workaround is possible at the application level; one or more of the following changes are required on the web tier to fix the issue.

- Upgrade to Apache 2.2.10 or newer (but it's possible that there are other vulnerable code paths in `mod_proxy` for non-timeout cases which are not covered by the fix until the as-yet-unreleased 2.2.16 version)
- Disable Apache keepalives (this works because `mod_proxy` does not go into the erroneous code on the first request received on a connection – and without keepalives, every request is the first request)
- Disable `mod_proxy` backend connection re-use, using the parameter `disablereuse`
- Disable `mod_proxy` backend keepalives
- Use alternative proxy software (i.e. not Apache & `mod_proxy`) for games requests

Testing a fix

The test client was run using the configuration #1, #2 and #3 (as detailed earlier), with the `disablereuse` parameter either On or Off. The results are shown in the following table. The issue was defined as not occurring if no instance of the error had occurred after a minimum of 500,000 requests.

Bug Report

We had logged a bug report with Red Hat.

Reproduction

Obviously the creation of a similar issue through a test client provided very strong evidence that the issue was within the `mod_proxy` module; the focus then became to reproduce the actual issue seen consistently on the staging environment.

Environment Configuration

Since the occurrence of the issue relies on a number of factors, including the interplay of persistent connections between clients, specific Apache threads and the game server, reproducing the error consistently on a standard environment would be very difficult. Therefore the following configuration changes were made so that the error could be reproduced at will.

- All Apache servers except one were shutdown. The remaining server was restarted in debug mode. This creates only one worker (see <http://httpd.apache.org/docs/2.2/programs/httpd.html>)
`sudo /usr/sbin/httpd -X -f /etc/httpd/conf/httpd.conf`
- The hosts file on the client machine running the test was changed to point to the single web server running the debug-mode Apache instance. This ensured that the whole test was conducted through a single worker.
- All Tomcat servers except one were shutdown. This was not strictly necessary, but enabled easier log file monitoring. The database wait time out (defines

how long a connection will wait for a locked row to be unlocked) was increased to 3 minutes to exceed the Apache client timeout of 2 minutes

A further requirement to reproduce the issue is that a request to the server times out. This could be done by introducing an artificial delay into the server code, however this would have required a code change, which would reduce the validity of the test. Instead, database load was simulated by locking a row in the database that the server would normally read during the processing of a request. A SQL script was written that carried out the following

```
-- Start a transaction
begin work;
-- Lock the row
update tCGGameLastPlay set version =
version where cg_id = X and cg_acct_id = Y;
-- ... wait for the client request to timeout
-- ... then release the lock to allow the
response to be returned to Apache rollback
work;
```

A second TCL script `req.tcl` was created that sent two HTTP requests (from a single user) to Apache down a single keepalive connection, the second of which would time out due to the locked database row. This is the crucial step to reproduce the bug, and ensures that the correct branch of `mod_proxy` code was entered.

Test Pre-Conditions

Two customers were required for the test. Alice had a balance of \$10,000 and Bob had a balance of \$0 (these are arbitrary values, the important part is that the two balances are obviously different).

Two separate games, Roulette and Blackjack were selected for the test, and Player B had played both of these games before (necessary for locking). It is important that for the duration of the test, no other game play occurs on the system.

Test Procedure

- Load the portal into a browser
- Log in Bob
- Using a web development tool, copy the authentication cookie set for Bob
- Close the browser window WITHOUT logging out Bob
- Relaunch the browser, and navigate to the portal. Log in Alice
- Launch any game in real play. Note that it should show Alice's balance in the top bar of the game, and in the header of the portal
- Paste Bob's authentication token into the request script req.tcl
- Lock the row in table tCGGameLastPlay for Bob's last play of Roulette
- Run the req.tcl script. This sends one Blackjack Init request to the game server, followed immediately (without waiting for the response) by a Roulette Init request, down the same HTTP 1.1 connection. A Blackjack response will be received immediately, but no Roulette response will be received, and this request will time out after 30 seconds with a HTTP 502 error
- The Apache web server error log will show a 502 error
- Unlock the row in table tCGGameLastPlay (At this point, the game server log should show that the second request processes successfully)
- Hit the Play or Bet button in Alice's game
- Alice's balance will drop to \$0. At this point, she is now logged in as Bob
- When the next AJAX request polls the portal, the header bar will change to show Bob's information and balance. Clicking on the link to My Account will bring up Bob's details.

Conclusion

Obviously the production of a similar issue through a test client provides very strong evidence that the issue lies within the mod_proxy module; however for full

confidence it would be ideal if the actual error raised in the original support call could be reproduced on the staging environment with the original live environment configuration. Following this, a further test could be carried out with the amended configuration to check that the error did not re-occur. The client could then have a high degree of confidence that this issue will be resolved by the fix proposed, and could release these changes to live.

This is the current focus of the ongoing investigation. The aim is to be able to consistently reproduce the problem through end-to-end game play making only minimal changes to the standard configuration of the site. In order to do this, the following scenario should be followed

- The Apache client request time-out should be set to a low value
- Client A sends an initial request to the Server over a persistent HTTP connection
- Client A sends another request to the Server over the same
- HTTP connection which takes longer than the Apache timeout, but the Server still returns a response
- Client B sends a game request that uses the same proxy thread as Client A

After more analysis and testing all issues were solved and the site was re-launched.

Conclusion

We hope this document has shed some new light on the Online Casino Gaming World. As Online Casino's become more popular and new products are introduced, more threats and security concerns will arise.

Network Test Labs (NTL) is at the forefront of information security. Our main goal is to make the World Wide Web safe and secure for all to enjoy.

If you have any questions or concerns about online Casino's, or would just like to say hello, feel free to e-mail us info@networktestlabs.com