Runtime Infrastructure - Bug #30

detect denial of service attempt and try to reduce the impact

09/15/2011 12:00 PM - Greg Shah

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

Related issues:

Related to Runtime Infrastructure - Bug #7498: allow a single Web client laun...	**New**

History

#1 - 09/15/2011 07:52 PM - Greg Shah

@45105 - issue description:

```
It is possible to connect a P2J client to the server and get to the login screen.  At that point, the user can
 walk away and the client can sit at the login screen for an indefinite period of time.

If this was done by a very large number of clients (accidentally or with malicious intent), it is possible to
exhaust the resources of the server system and thus deny service to new incoming connections.

However, we must not disable the use case of a set of terminals that are meant to sit at the login prompt, wai
ting for a user to walk up and use the system.  That is valid.

We must distinguish between this case and a real denial of service.
```

#2 - 09/18/2011 05:36 PM - Redmine Admin

@45189 - Status changed from WIP to Hold

#3 - 04/12/2012 11:23 AM - Greg Shah

Imported from JPRM on 2012-04-12 11:23:24.939:

```
TASKID      = 6098
PROJECTID   = 125
STATUS      = 3 (Hold)
DESCRIPTION = detect denial of service attempt and try to reduce the impact
OWNER       = GES
ASSIGNEE    =
BILLABLE    = false
PRIORITY    = 5 (Normal)
```

```
PHASE       = 31 (Development)
COMPONENT   = 97
ESTEFFORT   = 0.000000
ESTSTART    = 2011-09-15
ESTSTOP     =
ACTSTOP     =
CASENUM     =
VENDORID    =
COMMENT     = ''
LASTWIP     = 2011-09-18
```

#4 - 10/23/2012 11:30 AM - Greg Shah

-------- Original Message --------
Subject: Re: Oasis blocking other secure clients? [was: P2J Connection Problem]
Date: Mon, 04 Apr 2011 13:48:12 -0400
From: Eric Faulhaber <ecf@goldencode.com>
Reply-To: ecf@goldencode.com
Organization: Golden Code Development
To: Constantin Asofiei <ca@goldencode.com>
CC: Greg Shah <ges@goldencode.com>

Constantin,

I see. These are good findings. I don't see any drawback to moving the NetSocket instantiation into Incoming.run(), but I'm not sure of the best approach for a timeout. Greg and I talked about this issue recently. A client may legitimately sit idle for a long period of time and appear to be hung, but it should not necessarily be disconnected. A heartbeat would resolve this issue, but that is a non-trivial addition to the architecture, because of the complications of Ctrl+C processing.

I will discuss these items further with Greg when he returns.

Thank you for getting Benjamin straightened out with Oasis.

Regards,
Eric

On 04/04/2011 12:59 PM, Constantin Asofiei wrote:

> Eric,
>
> Some followup: the "Secure Socket Listener" thread is blocked as long as the i.e. Oasis app doesn't back out the connection attempt. I didn't test, but when it backs out the connection attempt and the socket is closed, then the listenWorker loop should notice and continue its job. My point is that a malicious client (or some automated code which connects to the P2J server incorrectly and this is not noticed in good time) can block the "Secure Socket Listener" loop and deny all subsequent secure connection attempts.
>
> Thanks,
> Constantin
>
> On Mon, Apr 4, 2011 at 7:41 PM, Constantin Asofiei <ca@goldencode.com> wrote:
>
> Eric,
>
> The issue is that the connection attempt blocks just when the new Incoming(...) object is created by listenWorker, when the socket is instantiated in the c'tor. I think it may be a good idea to just move the socket creation into the thread, and not in the c'tor.
>
> BTW, if a certain client hangs, we may end up with a runaway threads which may cause the server to crash at some point... some timeout should be added.
>
> On Mon, Apr 4, 2011 at 6:59 PM, Eric Faulhaber <ecf@goldencode.com> wrote:
>
> Constantin,

On 04/04/2011 11:39 AM, Benjamin Johnson wrote:

> Two, I am a little concerned that a client with a mis-configured bootstrap can hang the thread that listens for secure connections. Shouldn't there be some sort of timeout so that if a mis-configured client attempts to connect, other properly configured clients can still make a secure connection?

Is Ben drawing a valid conclusion here? I thought that RouterSessionManager.listenWorker() accepted the incoming connection and immediately spawned a new thread with a new instance of RouterSessionManager$Incoming to handle authentication and session setup. How are we getting into a situation where a bad Oasis config blocks other secure clients from connecting?

Thanks,
Eric

#5 - 10/23/2012 11:32 AM - Greg Shah

-------- Original Message --------
Subject: Re: Oasis blocking other secure clients? [was: P2J Connection Problem]
Date: Wed, 13 Apr 2011 18:59:38 +0300 (GTB Daylight Time)
From: Constantin Asofiei <ca@goldencode.com>
To: ges@goldencode.com
CC: ecf@goldencode.com

Greg,

I'm talking about a timeout at initial socket creation, when login is attempted. For the timeout, I think there should be a separate thread waiting for notification that the socket was established successfully or, if such notification isn't received, will terminate the login thread (i.e. the Incoming instance).

Thanks,
Constantin

On Wed, Apr 13, 2011 at 6:51 PM, Greg Shah <ges@goldencode.com> wrote:

> Constantin,
>
> I agree, these issues need to be addressed. We can't allow a known opportunity for denial of service to remain in the code.
>
> I am OK with moving the code out of the Incoming constructor. I am also OK with adding a timeout. On the timeout: what do you propose? Are you talking about a general session timeout or a timeout for login/setup processing?
>
> Thanks,
> Greg

#6 - 10/23/2012 11:33 AM - Greg Shah

-------- Original Message --------
Subject: Re: Oasis blocking other secure clients? [was: P2J Connection Problem]
Date: Wed, 13 Apr 2011 12:01:04 -0400
From: Greg Shah <ges@goldencode.com>
Reply-To: ges@goldencode.com
To: Constantin Asofiei <ca@goldencode.com>
CC: ecf@goldencode.com

This makes sense. Go ahead with both changes (moving code from ctor and the monitor thread).

Thanks,
Greg

#7 - 10/23/2012 11:33 AM - Greg Shah

-------- Original Message --------
Subject: Re: Oasis blocking other secure clients? [was: P2J Connection Problem]
Date: Thu, 14 Apr 2011 14:14:55 +0300
From: Constantin Asofiei <ca@goldencode.com>
Reply-To: ca@goldencode.com
To: ges@goldencode.com
CC: ecf@goldencode.com

Greg,

An idea of what the default timeout should be ? I'm thinking of something in the range of 5-10 seconds to be the default one, with maybe the possibility to be overridden using a directory setting.

Thanks,
Constantin

#8 - 10/23/2012 11:34 AM - Greg Shah

-------- Original Message --------
Subject: Re: Oasis blocking other secure clients? [was: P2J Connection Problem]
Date: Thu, 14 Apr 2011 07:36:16 -0400
From: Greg Shah <ges@goldencode.com>
Reply-To: ges@goldencode.com
To: ca@goldencode.com
CC: ecf@goldencode.com

Constantin,

It probably should be a bit higher. I think 30 seconds should be sufficient. I know this puts the system at more of a chance for a DOS attack. But on the other hand, in a case where the system is slow to respond because of high load, I don't want this timeout to cause a problem where none would have been.

I like the idea to allow overriding the value using the directory.

Ideally, we would be able to detect that a DOS attack is occurring. What if there was a single dedicated monitor thread for the P2J server? Each new Incoming thread would create a "record" in a queue in FIFO order. The monitor thread might have enough information to detect if a DOS was occurring. For example, if there was a rate of new incoming sessions more than X/minute and some percentage (99%?) of them were timing out... perhaps that would indicate a DOS and we could modify the timeout value to a lower number dynamically.

Just some thoughts to consider. The key is that I don't want to create problems for loaded systems that are not DOS attacks.

Greg

#9 - 10/23/2012 11:36 AM - Greg Shah

-------- Original Message --------
Subject: Re: Oasis blocking other secure clients? [was: P2J Connection Problem]
Date: Thu, 14 Apr 2011 15:18:40 +0300
From: Constantin Asofiei <ca@goldencode.com>
Reply-To: ca@goldencode.com
To: ges@goldencode.com
CC: ecf@goldencode.com

Greg,

> It probably should be a bit higher. I think 30 seconds should be sufficient. I know this puts the system at more of
> a chance for a DOS attack. But on the other hand, in a case where the system is slow to respond because of high load,
> I don't want this timeout to cause a problem where none would have been.

OK

> I like the idea to allow overriding the value using the directory.

Where should this be placed ? Should we add a "no timeout" value too (i.e. -1), which will disable the monitoring thread ?

> Ideally, we would be able to detect that a DOS attack is occurring. What if there was a single dedicated monitor
> thread for the P2J server? Each new Incoming thread would create a "record" in a queue in FIFO order. The monitor
> thread might have enough information to detect if a DOS was occurring. For example, if there was a rate of new
> incoming sessions more than X/minute and some percentage (99%?) of them were timing out... perhaps that would indicate
> a DOS and we could modify the timeout value to a lower number dynamically.

My idea of monitoring was for the listening loop (i.e. listenWorker) to create a monitor thread instead of an Incoming thread, for each incoming connection. The monitor thread will be responsible for creating the Incoming thread and closing the socket if it doesn't get good news in due time.

About using a single monitor thread - I think its usage would be like this:
1. Incoming thread adds the record to the queue before the connection attempt (i.e. NetSocket is created).
2. after the connection was made: if the queue is not locked (i.e. the monitor thread is walking it), it removes the record from the queue; if it is locked, it updates a flag in the record so that the monitor knows the connection was possible
3. the single monitor thread walks through the records in the queue (at each X seconds?) and, if the timeout is reached and the "connection made" flag is off, it closes the socket and terminates the Incoming thread. The record is removed from the queue if timeout is reached or the "connection made" flag is on.

So, the record for each Incoming thread should know:
- the socket
- the Incoming thread
- the time at which the connection attempt started
- the "connection made" flag

To detect a DOS attack, your idea sounds good. But, can we consider as malicious also the connections which did not timeout, but were terminated

without authentication or with authentication failure ?

Thanks,
Constantin

#10 - 10/23/2012 11:37 AM - Greg Shah

-------- Original Message --------
Subject: Re: Oasis blocking other secure clients? [was: P2J Connection Problem]
Date: Fri, 15 Apr 2011 11:13:36 -0400
From: Greg Shah <ges@goldencode.com>
Reply-To: ges@goldencode.com
To: ca@goldencode.com
CC: ecf@goldencode.com

Constantin,

> Where should this be placed ?

In the runtime area.

> Should we add a "no timeout" value too (i.e. -1), which will disable the monitoring thread ?

Yes, this is a great idea.

> 1. Incoming thread adds the record to the queue before the connection attempt (i.e. NetSocket is created).

Yes.

> 2. after the connection was made: if the queue is not locked (i.e. the monitor thread is walking it), it removes the record from the queue; if it is locked, it updates a flag in the record so that the monitor knows the connection was possible

This part I am not as sure about. I prefer some form of semaphore that is non-contended for the Incoming thread. Look at the java.util.concurrent.CountDownLatch. This seems to have the features needed. You can use it in a polling manner with getCount() or you can use the blocking await(timeout,units) method...

The idea is that this way, the Incoming thread does not ever have to care about locking the queue. The queue is simply a way to aggregate data for analysis of DOS by the monitoring thread.

> 3. the single monitor thread walks through the records in the queue (at each X seconds?) and, if the timeout is reached and the "connection made" flag is off, it closes the socket and terminates the Incoming thread. The record is removed from the queue if timeout is reached or the "connection made" flag is on.

Is there any problem calling close() from another thread? Race conditions to worry about?

> But, can we consider as malicious also the connections which did not timeout, but were terminated without authentication or with authentication failure ?

Good point. Yes, I think these must be considered as malicious too. Normal cases will only have a few of these at any given time. So the normal use case won't get confused with a DOS.

Thanks,
Greg

#11 - 10/23/2012 11:39 AM - Greg Shah

-------- Original Message --------
Subject: Re: Oasis blocking other secure clients? [was: P2J Connection Problem]
Date: Tue, 26 Apr 2011 17:48:04 +0300
From: Constantin Asofiei <ca@goldencode.com>
Reply-To: ca@goldencode.com
To: ges@goldencode.com
CC: ecf@goldencode.com

Greg,

> Where should this be placed ?

In the runtime area.

OK, I named it "socket_timeout".

> 2. after the connection was made: if the queue is not locked (i.e. the monitor thread is walking it), it removes the record from the queue; if it is locked, it updates a flag in the record so that the monitor knows the connection was possible

> This part I am not as sure about. I prefer some form of semaphore that is non-contended for the Incoming thread. Look at the java.util.concurrent.CountDownLatch. This seems to have the features needed. You can use it in a polling manner with getCount() or you can use the blocking await(timeout,units) method...
>
> The idea is that this way, the Incoming thread does not ever have to care about locking the queue. The queue is simply a way to aggregate data for analysis of DOS by the monitoring thread.

I couldn't find how to use the CountDownLatch in our case... here we have a single monitoring thread which waits for data from a various number of threads. The CountDownLatch is used to sync a specific number of threads so that they start at the same time or to sync a certain action so that is performed after all worker threads have finished... Also, we can't get rid of the queue sync'ing completely, as at least we need to sync on it when the data is added or removed.

So, my approach was that the queue is locked only when:
- a new record is added
- a copy is made, which will be used for processing; during processing, the "to be removed" records are collected (i.e. records which timeout, went through authentication, etc).
- when the queue is cleaned up (i.e the collected "to be removed" records are actually removed from the queue).

This is the only way I could find to minimize overhead in the Incoming thread.

> Is there any problem calling close() from another thread? Race conditions to worry about?

I couldn't find any.

Thanks,
Constantin

#12 - 10/23/2012 11:39 AM - Greg Shah

-------- Original Message --------
Subject: Re: Oasis blocking other secure clients? [was: P2J Connection Problem]
Date: Tue, 26 Apr 2011 10:57:33 -0400
From: Greg Shah <ges@goldencode.com>
Reply-To: ges@goldencode.com
To: ca@goldencode.com
CC: ecf@goldencode.com

Constantin,

> I couldn't find how to use the CountDownLatch in our case... here we have a single monitoring thread which waits for data from a various number of threads.

Each record in the queue has a separate CountDownLatch. The single thread only ever needs to wait or poll the first record in the queue.

Greg

#13 - 10/23/2012 11:40 AM - Greg Shah

-------- Original Message --------
Subject: Re: Oasis blocking other secure clients? [was: P2J Connection Problem]
Date: Tue, 26 Apr 2011 18:56:47 +0300
From: Constantin Asofiei <ca@goldencode.com>
To: ges@goldencode.com
CC: ecf@goldencode.com

Greg,

Lets assume the first record established the connection and is in the authentication code, which now waits for the user to enter its credentials. If we consider the "authentication failure" connections as malicious, then we can't remove the record from the queue until it passed or failed authentication (no timeout is involved here). So, waiting for a notification that the first record in the queue is finished, might lead the monitoring thread into an apparent "deadlock", which stops processing any subsequent records which might be timeout or otherwise need to be removed from the queue (i.e. passed or failed authentication).

I tried to compromise this by waiting for the specified timeout interval (but not less than i.e. 100ms), before processing the records in the queue.

Thanks,
Constantin

#14 - 10/23/2012 11:40 AM - Greg Shah

-------- Original Message --------
Subject: Re: Oasis blocking other secure clients? [was: P2J Connection Problem]
Date: Tue, 26 Apr 2011 13:39:54 -0400
From: Greg Shah <ges@goldencode.com>
Reply-To: ges@goldencode.com
To: Constantin Asofiei <ca@goldencode.com>
CC: ecf@goldencode.com

You can always using polling. The monitor thread can walk an ordered list of CountDownLatch instances and poll each one as needed.

Greg

#15 - 10/23/2012 11:42 AM - Greg Shah

- File ca_test20110427a.zip added

-------- Original Message --------
Subject: Re: Oasis blocking other secure clients? [was: P2J Connection Problem]
Date: Wed, 27 Apr 2011 17:24:24 +0300
From: Constantin Asofiei <ca@goldencode.com>
Reply-To: ca@goldencode.com
To: ges@goldencode.com
CC: ecf@goldencode.com

Greg,

Please take a look at the attached RouterSessionManager file and see the following notes (which I think can make more sense if you take a look at the code too):

1. I've used a java.util.concurrent.ConcurrentLinkedQueue queue to collect the records, so that the queue is still thread-safe, but without the need to explicitly synchronize on it.

2. a copy of the queue is walked instead of the main queue, because if we walk the main queue and records keep getting added to the queue (i.e. connections keep coming), the walk may take a very long time. Using a copy ensures the walk will be done in an finite amount of time and any connections which are in the authentication code will get a chance to be rechecked on the next walk.

3. statistics are collected in a map, per each minute, about the:
- total number of incoming connections
- number of timeout connections
- number of failed authentications
- number of passed authentications

4. a DoS is presumed when 90% of the connections are malicious

5. something I'm not sure about - if less than i.e. 10% of the connections passed authentication, and lets say more than 100 (or a certain percent ?) are still in the authentication code, can this still be considered a possible DoS ? Maybe we can add a condition like 'if, for a certain minute, more than X hours have passed and more than 90% of the connections are still in authentication code, a DoS is possible'.

6. the statistics for a certain minute are removed from the map when the total number of processed records (i.e. timeout + failed + passed) is equal to the total number of incoming connections in that minute. The idea is that I don't think we don't need to track the complete history of statistics... maybe we could keep the statistics for the last hour or so, or maybe add a new directory parameter. Do you think it can be useful to add a screen in the Admin Console with this history ?

7. this wasn't implemented: I was thinking that we could keep some average statistics (for the last day or week), which could be used for some more analysis.

Thanks,
Constantin

#16 - 10/23/2012 12:11 PM - Greg Shah

I have finally taken the time to review this. Sorry about that. Better late than never, I guess.

Answers to your specific questions:

```
5. something I'm not sure about - if less than i.e. 10% of the connections passed authentication, and lets say
 more than 100 (or a certain percent ?) are still in the authentication code, can this still be considered a p
ossible DoS ? Maybe we can add a condition like 'if, for a certain minute, more than X hours have passed and m
ore than 90% of the connections are still in authentication code, a DoS is possible'.
```

Yes, I think such a condition is a good idea.

```
6. the statistics for a certain minute are removed from the map when the total number of processed records (i.
e. timeout + failed + passed) is equal to the total number of incoming connections in that minute. The idea is
 that I don't think we don't need to track the complete history of statistics... maybe we could keep the stati
stics for the last hour or so, or maybe add a new directory parameter. Do you think it can be useful to add a
screen in the Admin Console with this history ?
```

Optimally, the amount that we track (and log periodically) would be configurable. I'm thinking of being able to set thresholds that allow everything to be kept (at one extreme) and very little/nothing to be kept (at the other extreme).

Yes, I can see this being useful in the admin console.

```
7. this wasn't implemented: I was thinking that we could keep some average statistics (for the last day or wee
k), which could be used for some more analysis.
```

Yes, higher level stats are of interest.

General feedback on the code:

1. The approach looks fine. It is a bit difficult to understand, but it is in an area that is complex, highly variable and difficult to synchronize.

2. I would like to see the SocketMonitor as a separate class.

3. I would like to see the IncomingConnection as a separate class. Make sure that any necessary features are encapsulated (e.g. no direct access to the members like signal).

3. I would prefer a more object oriented approach to the statistics instead of the long[4] instances. If that would make things more complex, then it is not a hard requirement. But it seems like some of the code for handling the statistics might be better handled in a (small) common class rather than being spread around the other code. I am also hoping this would make the main monitoring loop shorter, which would make it easier to read/maintain.

4. For consistency, getStats() should always be used in preference to stats.get().

5. Perhaps it would be better to clear changed at the bottom of the loop instead of at the top? Not a big deal, but it would be a little better for memory management.

6. Go ahead and make specific proposals on the TODOs in the code.

7. Generally, I think we need the major heuristics (e.g. the 90% threshold) to be configurable. We may certainly find that in real life cases, the hard coded algorithms don't work. But if we can use the directory to configure all the thresholds, timeouts and so forth, then this should make the code much more likely to succeed over time.

#17 - 10/23/2012 12:18 PM - Greg Shah

- Project changed from Core Development to Runtime Infrastructure

#18 - 10/31/2012 11:56 AM - Greg Shah

- Tracker changed from Feature to Bug

#19 - 11/07/2023 07:38 AM - Galya B

- Related to Bug #7498: allow a single Web client launch if the client presses ENTER key multiple times added

Files

ca_test20110427a.zip	16.9 KB	10/23/2012	Greg Shah