



Developer Access to Production

[Solution NOTE]

Introduction

Most enterprises in financial services, energy, chemical, retail, government and others have developed a production and non-production (QA, test, etc.) environment. The necessity for two disparate environments is in large part driven by the sensitivity to resource/application problems and/or outages which can cost today's enterprise millions of dollars in lost revenue, regulatory fines and untold impact in corporate image. For these and other reasons, enterprises typically create a hard divide between their production network/infrastructure and development/QA/test/etc.

In a perfect world, production environments would run forever, without incident or need for human access. Unfortunately such a world does not yet exist – for example, change control is critical for maintaining the uptime, availability, and integrity of production systems. Most large organizations have policies that very discretely dictate the process for moving code or data to production environments. Many have automated systems that validate the build, ensure the move is properly executed, and verify the target environment. It would be great if this change process would be all that is necessary for a production environment. Unfortunately, even the best written applications sometimes fail, and when this occurs, the developers who support the application may need to access the production system to evaluate and potential repair the problem.

Depending on the company, developers accessing production may be a daily event or a monthly anomaly. The same is true for other support individuals, including database administrators, technical support personnel, or even system administrators. System maintenance and patching will always require some attention.

The issue of access to production is the following:

- 1. Is the individual authorized to access the production system?**
- 2. Is the individual authorized to make changes to the production system?**
- 3. Did the individual make only the changes authorized and were the changes made correctly?**

Companies have taken different approaches for the first issue. Almost all companies use a ticketing or change system to help manage the question of authorization. The question becomes a matter of whether the ticketing system can be used to ensure that no non-authorized access occurs.

Access control solutions are used to ensure that the individual must authenticate to the system. The question is whether a valid user is an authorized user. As an example, a developer may be a valid user on a production Unix system, but is only authorized to access the system when they have a valid problem ticket.

The ability to grant access only when authorized has been a driving force for various access control solutions. The ability to 'enable' an authorized user can be challenging. Some organizations will separate the networks (production and non-

production) so that gateway controls (firewalls, etc) can be used to provide 'on demand' access.

The second issue drives directly into access control. On a target system, security controls are in place to allow or deny changes based on an access level. To use the Unix example, a system administrator may have a userid for basic troubleshooting, but needs access to 'root' to make many of the changes required by their job function. There is also a trade-off for individual accountability. Using a generic 'developer' userid allows for significantly less user administration, but destroys the ability for individual accountability at the target system. Conversely, having a unique userid for each developer, database administrator, and system administrator requires significant administrative work and maintenance.

The third issue is the one that makes the newspaper headlines most often and is the one most difficult to answer and detect. Did the valid and authorized user only make the changes that were expected? When a change is made through a change control system, the automated process typically provides a source to load comparison, reports on all moves, and provides great detail into the pre and post status of the system. Unfortunately, when a developer must access a production system that has a failed application, their first concern is not the audit trail, it is availability. Even if there is no ill intent, many developers would be hard pressed to explain exactly what they did to resolve the issue. This also is the issue that allowed the Fannie Mae employee to install the malicious code, which through thwarted according to experts could have destroyed all data on the over 4,000 servers at Fannie Mae.

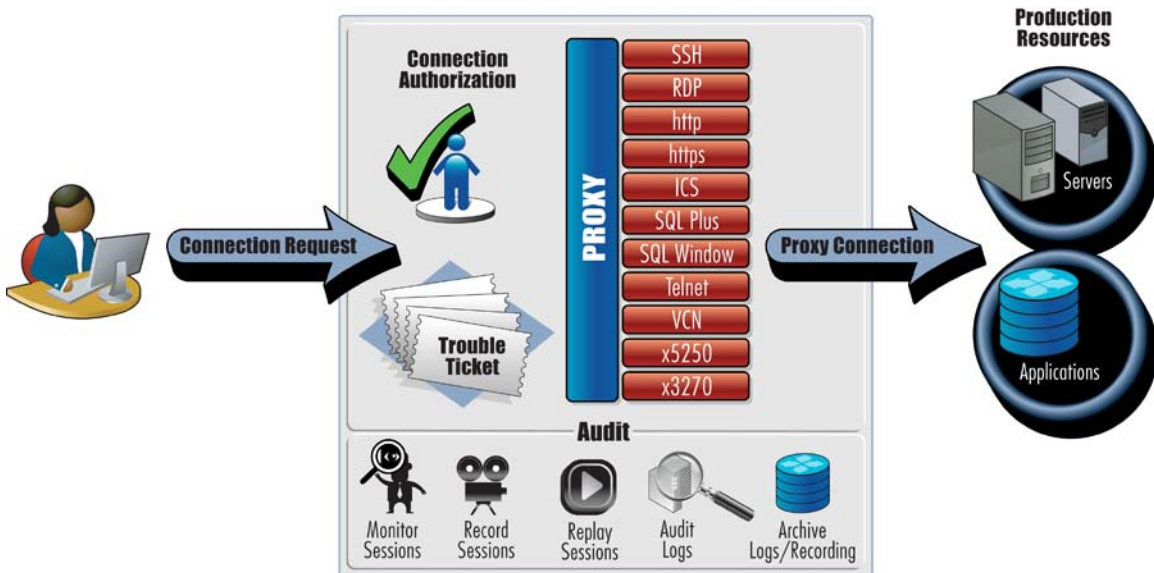
Solution

eDMZ Security's Total Privileged Access Management (TPAM) suite provides a unified solution that is able to address all three issues associated with developer access to production.

1. Is the individual authorized to access the production system?

TPAM provides the ability to define the users who may need access to production. When the access is needed, the user submits a request for the specific target system. The request can require another user to approve the request (maker-checker/dual control) and can also perform real-time validation against a ticketing system as part of the request approval process. TPAM can be configured to manage the credential of the account including auto-login on the production system, so that the user cannot bypass TPAM to access the production system directly (since they do not know the password) and the credential is never exposed.

The definition of the user in TPAM means that they are valid, they are approved and they are authorized. Once approved and authorized, TPAM establishes a proxy connection to the requested production resource as reflected in the diagram below.



In addition, TPAM easily integrates with your existing AD, LDAP or CMDB environment such that users, systems or accounts provisioned at a directory/CMDB level will automatically be synchronized into TPAM. TPAM also fully integrates with back-end ticketing systems to support ticketing system DB queries are part of access authorization.

2. Is the individual authorized to make changes to the production system?

TPAM provides SAPM (shared account password management) so that root or other accounts used to provide access can be controlled. TPAM allows for generic userids to be used at the target system, since individual accountability is provided and assured by TPAM as noted below. This allows a company to avoid the administrative overhead of unique accounts on the production system without sacrificing individual accountability.

TPAM also supports Privileged Command Management (PCM), through PCM, individual users can be granted access to a production system with limit to the command/environment they are able to execute.

3. Did the individual make only the changes authorized and were the changes made correctly?

Normal connection and/or host base audits provide information on who had access, when they had access and how long they had access. They are not able to answer to auditor question “what did they do” with the access granted.

TPAM through our innovative session monitoring and recording is able to answer the question “what did they do”. Every keystroke, mouse movement, application access can be monitored, recorded and archived for future audit/forensic requirements – without the need for host based agent software. Session recordings are activity driven (i.e. no recording of inactivity), compressed and encrypted – assuring both security and disk/storage efficiency. Session recordings can be retrieved and replayed via TPAM session log management as shown in the screen shot example below:

Session Logs Listing

No Selection

Filter | Layout | Listing

Session Dates

All Dates

Last N Days Days: (Default is 5. This is the only date filter that will save.)

Single Day Date: (i.e. mm/dd/yy, dd-mon-yy, mon dd yy)

Date Range From: To:

Text Filters

System Name:

Account Name:

User Name:

Ticket Number:

Status

All Session Logs

Default Filter Settings

No Action



Session Logs Listing

RequestID: 2671 Start Date: 9/1/2010 12:32:56 PM

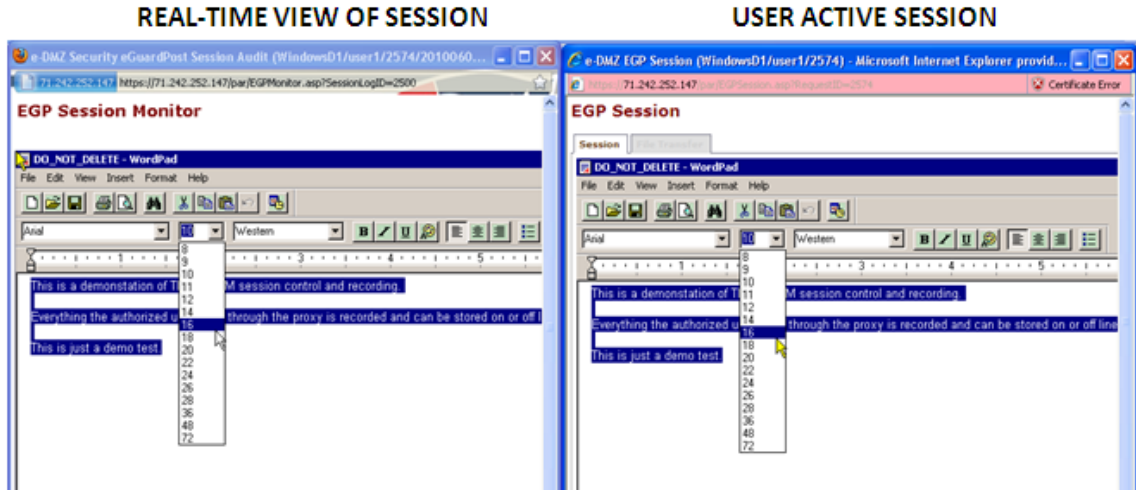
Filter | Layout | Listing | Comments

UserName	User Full Name	Start Date	RequestID	File Size	Duration	System	Account	Ticket Number	Archive Server Path	Status
reqD1	Req. Test	9/1/2010 12:38:21 PM	2673	2607	0:04:55	WindowsD1_PCH	Task_Manager_Us (user1 on WindowsD1) {Command: Windows Task Manager}			Expired
reqD1	Req. Test	9/1/2010 12:32:56 PM	2671	1340	0:06:03	WindowsD1	user1			Expired
reqD1	Req. Test	9/1/2010 12:26:32 PM	2670	1623	0:03:27	WindowsD1	user1			Expired

1 thru 3 of 3 records. Page: 1

Export to Excel | Export to CSV | **Replay Session** | Monitor Session | Archive Now

For those requirements that require real-time monitoring, TPAM session monitoring allows an auditor, reviewer or appropriately privileged administrator to view active sessions – the below example reflects the user session on the right and an active monitoring of that session on the left.



Enhancements for command level controls and audits extend the capabilities of TPAM beyond “what did they do” to control what they are able to do with granted access.

Conclusion

TPAM provides the ability to allow ‘on demand’ access for developers to production systems with full individual accountability, audit and control. The built in mechanisms to prevent circumvention (through target account credential management) can ensure that organizations know who is accessing their production environment, who authorized the access, and what the individual did with that access. Enhancements to control what an authorized user “can do” with granted access will further extend the values TPAM brings in solving security and compliance issues associated with developer’s access to production.

Contact Us

If TPAM/PSM or any of our TPAM suite modules is of interest, please give us a call or send us an email. We can easily support a more detailed webex and/or support on-site proof-of-concept as needed.

Phone: 772-252-1147

Email: sales@dfyj.com

Web: www.dfyj.com