



www.elock.com.my

WebAlarm Technical White Paper

(Covers v4.1)

June 2010

Revision 4.1

A Technical Whitepaper detailing the architecture of WebAlarm 4

In this Document :

Introduction	2
About WebAlarm	3
The Architecture	5
WebAlarm Agent Configuration	10
Alerts	11
Polling Configuration	11
Login Procedure	12
Integrity Monitoring Engine	13
Recovery Options	16
Restoration Processes	17
Update Management Agent Configuration	17
File/Directory Mapping	17
Email Notification	18
Data Publishing Process	18
Publishing To Multiple Servers	19
Appendix A	20
Appendix B.....	21
Appendix C.....	22
Contact Details	24

Introduction

Firewalls and Intrusion Detection Systems continue to mature offering various levels of functionality. Best practices recommend the use of Firewalls and Intrusion Detection Systems in tandem to have the most complete security picture.

The concept behind this is to use the firewall for prevention and the IDS for detection should the firewall perimeter be breached. In theory, this sounds like a good idea. In reality however, the idea is flawed due to the existence of product shortcomings such as faulty designs, platform dependencies and environmental problems.

Firewalls simply function to filter traffic that is deemed malicious. The threat lies in malicious traffic that is often masqueraded as genuine traffic resulting in the penetration of the firewall.

This paper introduces the concept of a last line of defense by exploring the architecture and functionality of the WebAlarm. Firewall and IDS exploits are published daily proving that these systems are no longer sufficient to uphold the integrity of the inner sanctum.

WebAlarm mitigates the risk of prevention reliance by ensuring full protection of internal data.

The introduction of WebAlarm implements an added layer of security by preventing unauthorized alteration of static data. It is designed to complement the outer perimeter security systems by enabling the protection of internal data.

Protection offered is in the form of restoration and not prevention entirely of file alteration hence being able to restore important data with a 100% efficiency. WebAlarm employs cryptographic algorithms to obtain digital fingerprints of monitored files to detect unauthorized file modification with pin sharp accuracy. Its unique integrity monitoring engine is also able to restore monitored files/folders back to their original form instantly upon tamper detection.

About WebAlarm

WebAlarm is essentially a recovery tool with the main function of immediately restoring data upon unauthorized modification. Commonly used in web server applications to ensure instant recovery of hacked web pages, it may also be used to protect the integrity of various other files and folders.

The integrity of files/folders are monitored by WebAlarm's proprietary integrity monitoring engine and upon unauthorized alteration, restores them back to original form almost immediately whilst alerting the system administrator of the security breach.

WebAlarm Features

WebAlarm was designed as a modular, scalable and flexible solution boasting the following key features.

1. Centralized User Management & Scalability

Various agents running on different platforms are configurable via a single console. This proven solution centralizes management and scales to accommodate corporate change and expanding remote needs.

2. Extensive Auditing & Log Filtering Ability

The log file serves as a forensic trail of hacker activity. All events such as file alteration alerts, file uploads and modification details are logged each tagged with a time and date. A common problem among security analyst is the sifting of pages and pages of logs. The log filtering ability enables quick filtering and easy searches on specific log events.

3. Remote Administration.

Agent and Console concept allows users to configure the agent only via the console which can be installed local to the agent or remotely. Remote console configuration is recommended as means of protecting the WebAlarm agent in the event of agent host pc compromise.

4. Multiple User Accounts

WebAlarm promotes the use of multiple user accounts with explicitly assigned permissions to support multiple users. Each agent possesses an individual user account database allowing assignments of different administrative and normal user accounts.

5. SNMP/Email / Sound Alerts

An alert is triggered when unauthorized modification of a monitored file has been verified. The agent can be configured to send out a SNMP trap message or an email or run a platform dependent program in the form of a batch file or shell script. The agent is also capable of emitting sounds to alert users of the breach. An additional feature is the ability to trigger an alert by emitting sounds upon the termination of an agent or the termination of an agent-console connection by the agent.

6. Requires very little CPU processing power

With its unique integrity monitoring engine, WebAlarm requires less than 5% processing power yet performs at lightning speed to ensure minimal downtime.

7. Simple, Intuitive Graphical User Interface (GUI)

Agent configuration is performed via the GUI console which provides a simple and easy to use Graphical User Interface.

8. Various platform support

WebAlarm supports a variety of Unix and Windows based platforms catering for the majority of server operating systems.

WebAlarm was designed to protect not only critical files and folders but also the software itself. It is

common hacker culture to sniff out ways and means of bypassing a locked door rather than searching for the key. With this in mind, WebAlarm employs a range of features wrapping the software in a layer of added security.

9. Secure Sockets Layer (SSL) Communication

Secure Sockets Layer form a secure means of communication between a remote console and agent defeating sniffing attempts at gaining passwords and configuration details flowing along the network wire. SSL requires digital certificates to verify the authenticity of a connecting host.

10. Admin user account

Admin account status or superuser status is the quest and target goal of hackers. WebAlarm requires an Admin username and password to allow connections between the console and the agent. Only one Admin account per agent is provided with the capacity to add multiple user accounts. Login as mentioned earlier is submitted via an SSL channel avoiding transmission of clear text passwords and thwarting sniffer attempts.

11. Login Security

The Admin user is able to restrict normal log on users to a range of IP addresses making it difficult for a foreign host within the internal network to pose as a valid host. This feature can also be applied to the administrative account. The use of a SSL channel between console and agent and digital certificates authenticate console hosts with valid IP addresses.

12. Backup Copy Monitoring

The Backup folder contains the backup copies of the monitored files/folders that are used by the agent to restore the original. To ensure the integrity of the backup copies, the files/folders in the backup folder are constantly monitored to retain its authenticity. Backup copies are also compressed to minimize disk space usage. Backup can be stored on a remote server that can be specified via the WebAlarm console.

13. Secure Upload Feature

The upload feature of WebAlarm allows an admin user to upload modified files to the agent as means of a web site update by allocating the admin user a specified amount of time to upload files. A scheduled upload feature allows upload time to be automatically launched based on a user selected schedule.

14. High Availability

Short of bypassing the agent, a hacker could just as easily terminate the agent to bypass its integrity monitoring engine. A unique feature of WebAlarm is its auto restarting feature which relaunches the application as soon as it is terminated and upon start up of the agent host.

15. Low bandwidth utilization

WebAlarm's Agent/Console architecture utilizes very little network bandwidth by sending only small amounts of configuration data and logon details down the network hence decreasing network latency.

16. Quick Response Time

WebAlarm uses a proprietary integrity monitoring engine that performs real-time surveillance on user selected files/folders and able to instantly detect file modification. The restoration process is simply a matter of replacing the tampered file with a copy of the original and occurs immediately keeping downtime to a minimum.

17. Content Update Module

A content update module is available to handle content update from various solutions and methods generally practiced by most organizations. The updated content would be automatically updated at the production server and the hash signatures as well as the backup copy would be seamlessly taken care of by WebAlarm.

18. Reporting Module

A reporting module is available to provide comprehensive charts, summary and detailed listing on the tampering data generated by one or more agents

The Architecture

This section describes how WebAlarm works under the covers. It describes the login processes, integrity monitoring , backup restoration and content update management.

WebAlarm Components

WebAlarm’s scalable, modular architecture allows agent configuration from a central management console to multiple multiplatform agents.

WebAlarm consists of four modules.

- WebAlarm Agent (WAA)
- WebAlarm Console (WAC)
- Update Management Agent (UMA)
- Update Management Console (UMC)
- WebAlarm Report Viewer (WRV)

The WAA runs as a Service on a Windows platform and as a daemon in the Unix world. It is in essence, the integrity checking and file restoration engine typically found on each monitored server.

The UMA, too, runs as a Service on a Windows platform and as a daemon on Linux. Its sole purpose is to provide a secure gateway for end users to perform content update to the production servers.

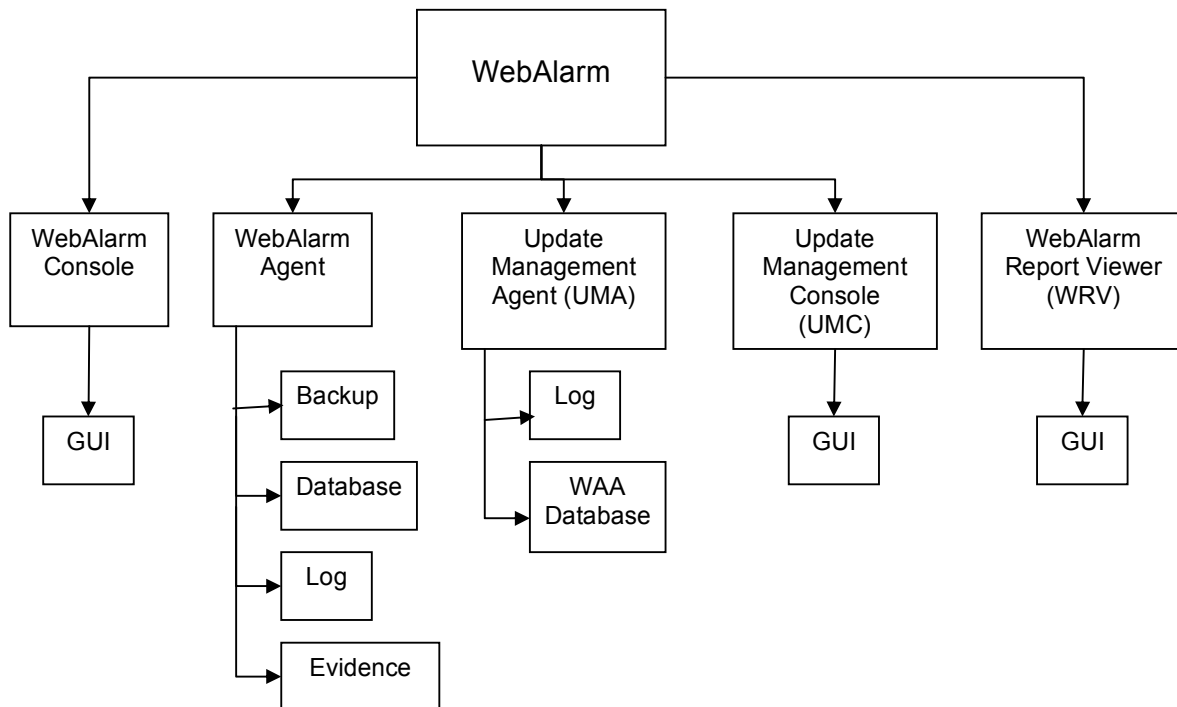


Figure 1: WebAlarm underlying components

The WebAlarm agent configuration is defined on the WebAlarm GUI console while the WebAlarm agent database is maintained on the WebAlarm agent Server. Although the WebAlarm agent can be deployed in a distributed configuration, security enforcement is completely integrated. Any number of

WebAlarm agents can be monitored and controlled from a single WebAlarm console providing a simple solution to system expansion. The UMA configuration is defined on the UMC GUI and just like its WebAlarm console counterpart, it can manage and control any number of UMAs within a single UMC GUI. Figure 2 shows a distributed Console/Agent configuration. The agent can also be configured via multiple consoles as illustrated in Figure 3.

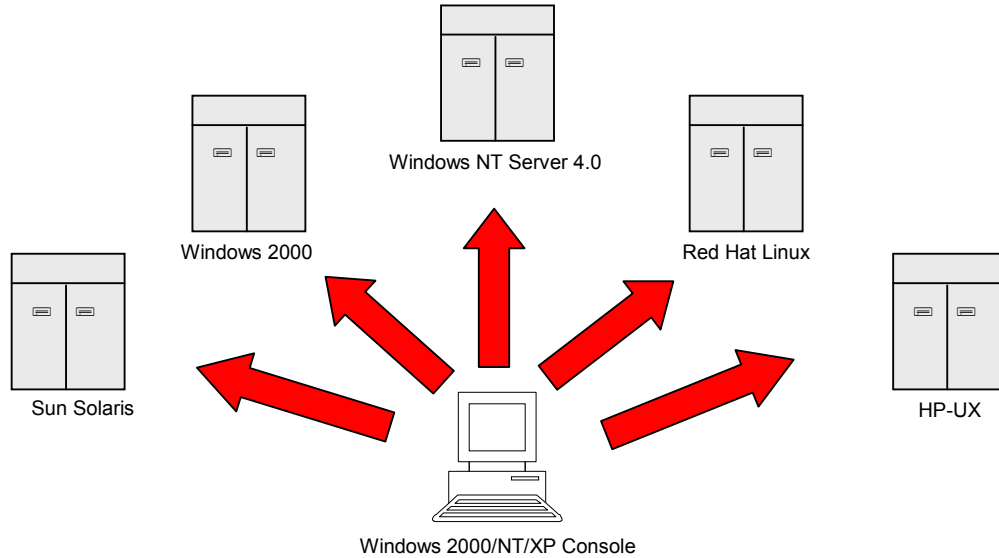


Figure 2 : Distributed WebAlarm Console/Agent Configuration

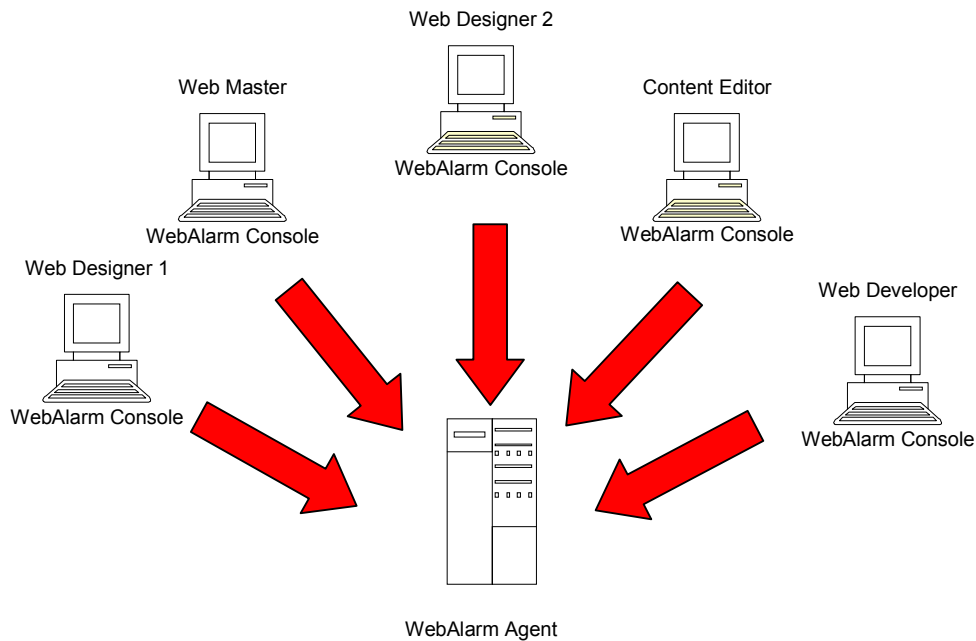


Figure 3 : Multiple WebAlarm console to single WebAlarm agent configuration support

Figure 4 shows a typical configuration involving a UMA and multiple WebAlarm agents while a UMC

managing multiple UMAs is illustrated in Figure 5.

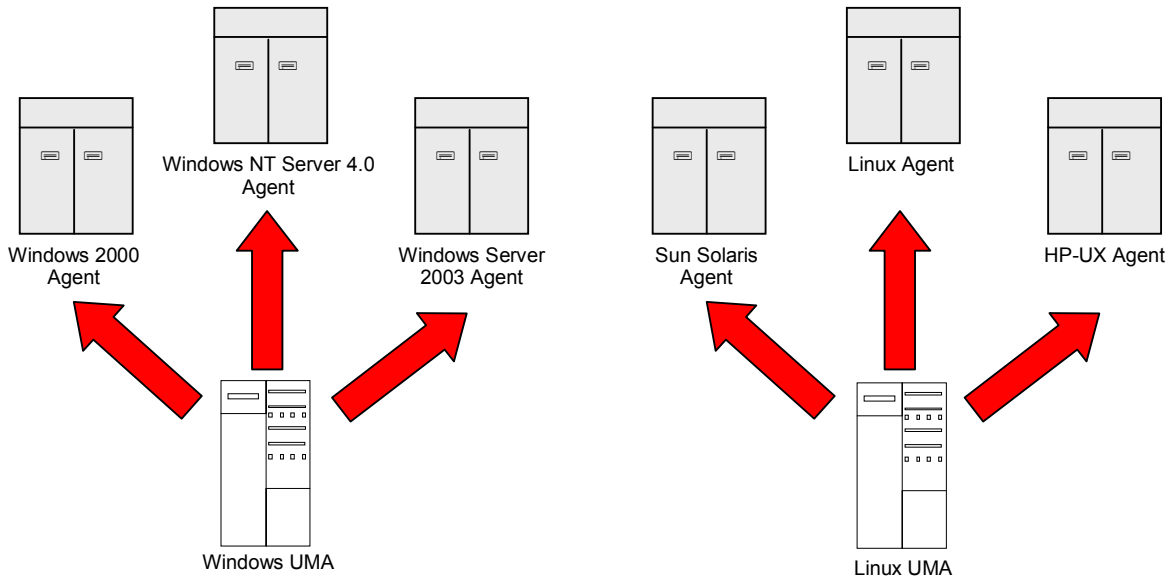


Figure 4 : One UMA serving multiple WebAlarm agents

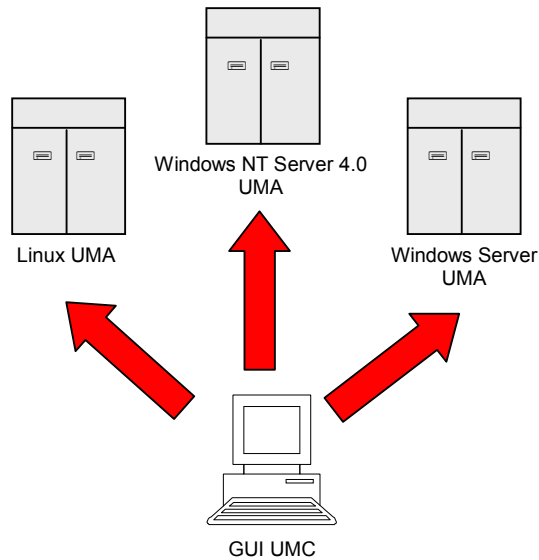
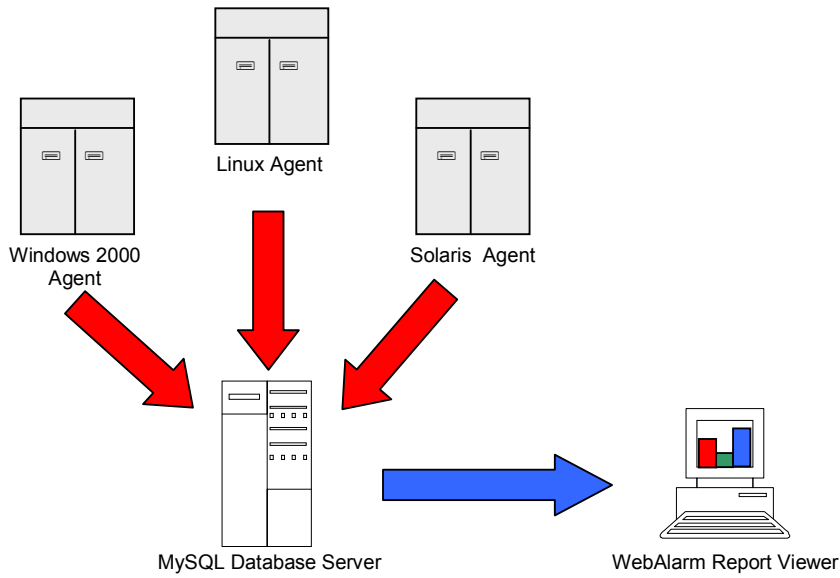


Figure 5 : One UMC managing multiple UMAs

WebAlarm Report Viewer is a standalone module that can retrieve and generate comprehensive charts, summary and detailed listing on the tampering data generated by one or more agents. The data generated would be stored in an active MySQL database server.

Figure 6 shows a typical setup involving a multiple WebAlarm agents, a MySQL database server and WebAlarm Report Viewer:



Graphical User Interface (GUI) WebAlarm Console

The WebAlarmagent is configured via an intuitive graphical user interface. The WebAlarmGUI console also includes a Log viewer. (Figure 7 & 8)

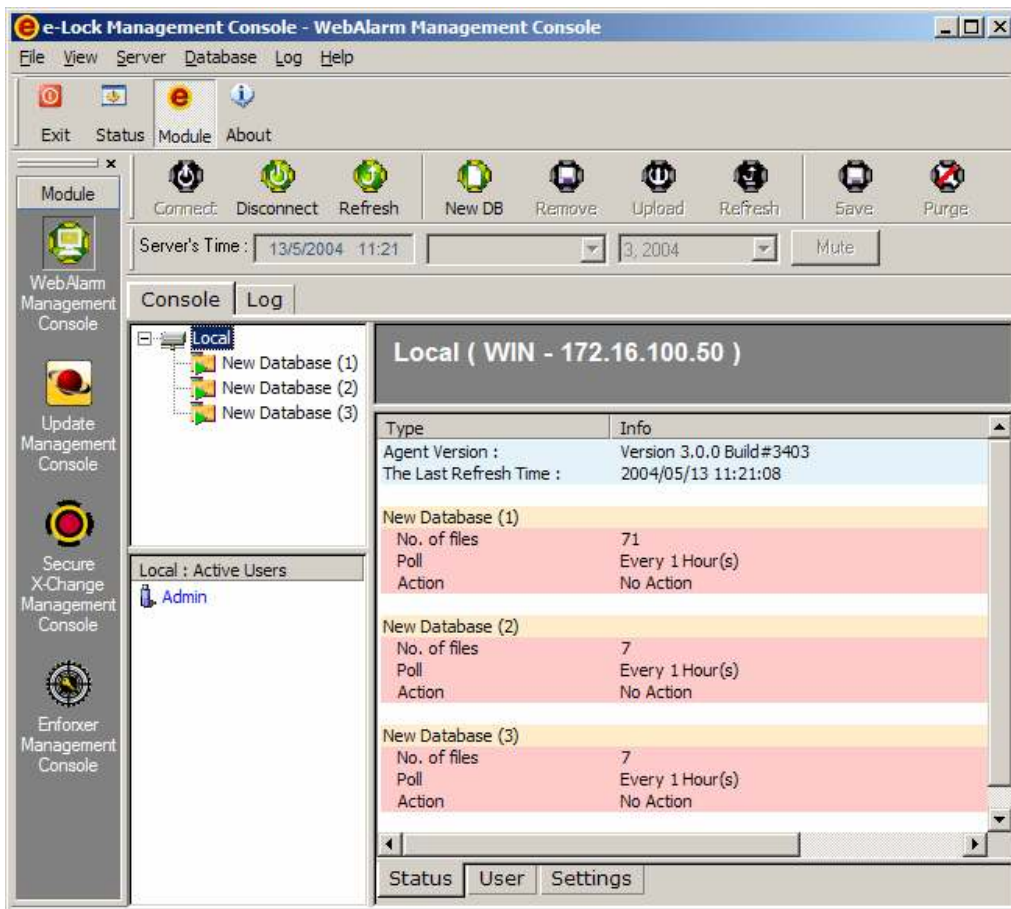


Figure 7: WebAlarm Console - Simple and intuitive Graphical User Interface

WebAlarm Agent

The agent is configured via the GUI console and configuration information is saved on the agent. The agent maintains the databases comprising integrity monitoring engine configuration, monitored files and alert configuration. It also stores the log files, user account details, backup copies and hash values.

The WebAlarm agent and console can be deployed on the same machine or in a Console/Agent configuration. For a list of supported platforms, see Appendix A “Supported Platforms” on page.

Four other crucial folders are installed along with the agent. Backup, Database, Log and Tamper. A description of the folders and their contents are as shown :

Folder	Contents
Backup	Copy of the selected original files and/or folders
Database	Hash values, monitored files and integrity monitoring engine settings.
Log	Logs of events
Tamper	Stores altered files/folders

These folders are constantly accessed by the agent to compare hash values, restore backups, add log events and store evidence.

Graphical User Interface (GUI) Update Management Console

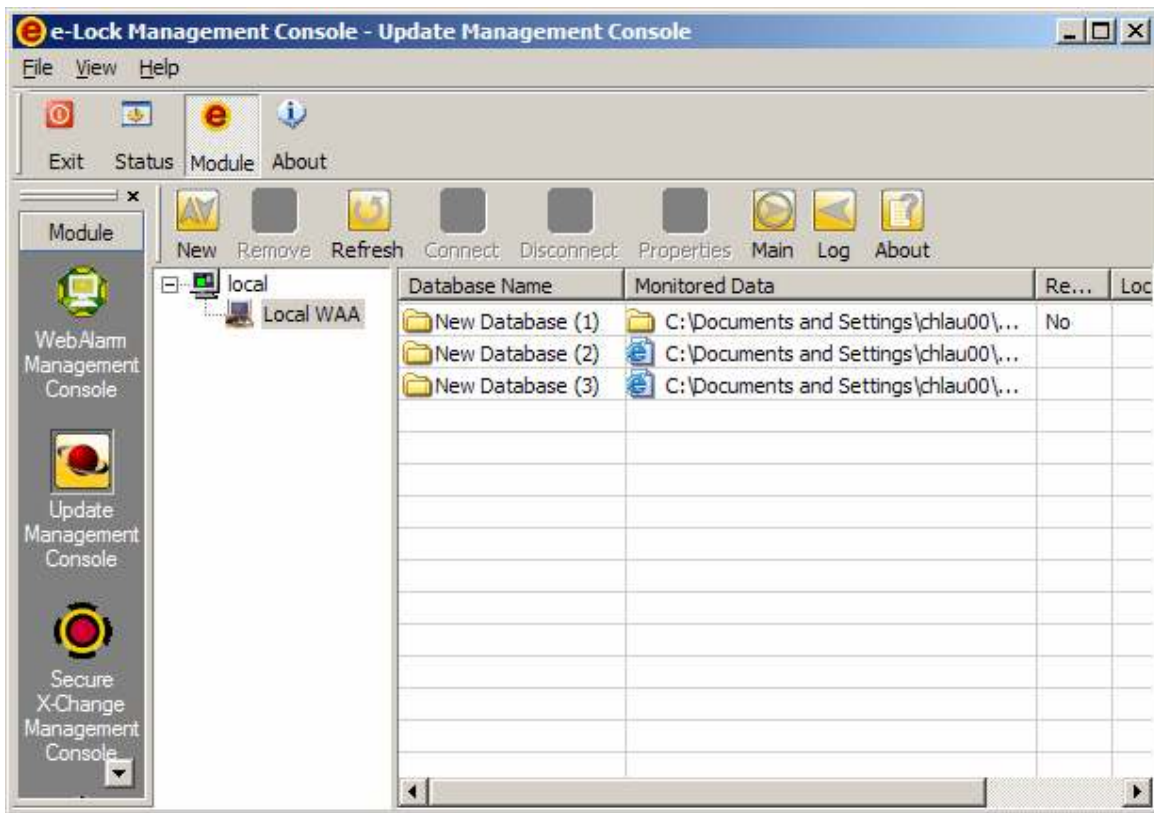


Figure 8 : Update Management Console - Graphical User Interface

Update Management Agent

Update Management Agent (UMA) is configured using the Update Management Console (UMC) and all setting information would be stored on the UMA. The UMA maintains the WebAlarm agent information comprising tcp connection as well as data mapping configuration. It also stores the log configuration and data, administrator account details, and email configuration.

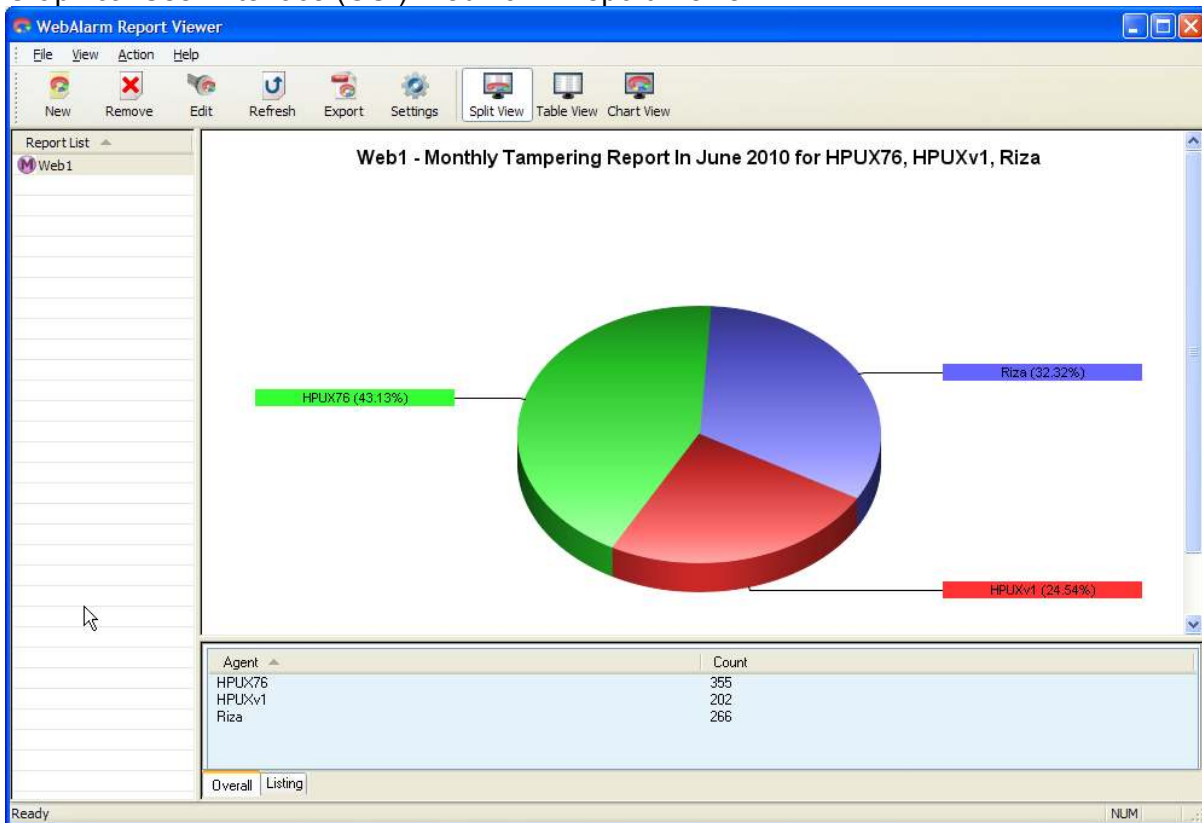
The UMA and UMC can be deployed on the same machine or in a Console/Agent configuration. For a list of supported platforms, see Appendix A “Supported Platforms” on page.

There are two crucial folders installed along with the UMA and a description of the folders and their contents are as shown :

Folder Content
Log Log of events

Waa Stores the WebAlarm agent tcp connection and data mapping configuration.

Graphical User Interface (GUI) WebAlarm Report Viewer



The WebAlarm Report Viewer is used to generate comprehensive charts, summary and detailed listing of the tampering activities capture by WebAlarm Agents. It is a standalone module and it can be freely installed on any Windows machine as long as it has network connectivity to the MySQL database server which houses all the tampering data.

WebAlarm Agent Configuration

A variety of options can be set via the GUI console. Some settings can only be configured by the administrative account holder.

Alerts

The agent can be configured to alert the users either via SNMP trap, email or execution of a program

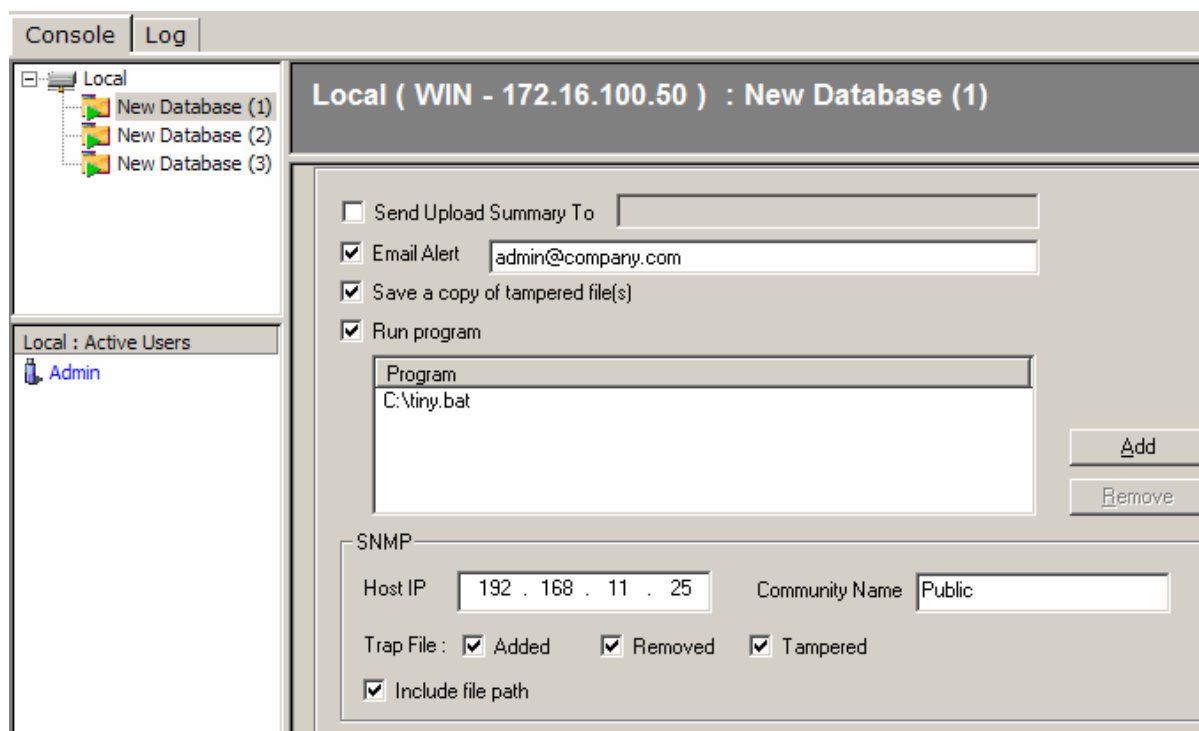


Figure 8 : Alert details are filled in.

Email configuration is easy and consists of simply filling in the relevant information. Multiple emails may be sent out by separating each email address with a semicolon. The software limits the number of email addresses to be sent out not by the number of email addresses but by a 256 character limit.

Alert emails are sent out as soon as an alarm has been detected. An alarm is fired when file alteration has been confirmed. In the case of a Windows platform, the alarm is fired when the OS trigger has been verified by performing a hash check.

An option of running a program is also provided. The configuration here is also easy consisting only of the selection of the program to be run. Programs can consist of either executables or batch files and in Unix – shell script

SNMP trap can also be configured by just filling in the NMS Host IP and the type of alert to be sent. Each trap message has the option to include the file path as well.

Polling Configuration

As explained later in the document, polling is referred to as the process of obtaining a hash value of the monitored file and comparing it with that of the original. The hash value of the original file is calculated upon file selection and upload and stored in the database on the agent.

Polling is used as means of verifying the OS trigger on Windows platforms and occurs directly after an OS trigger has been detected irregardless of the poll settings. The poll settings are to allow polling to be used not only as verification of an OS trigger but also as means of backup in the event that an OS trigger should fail.

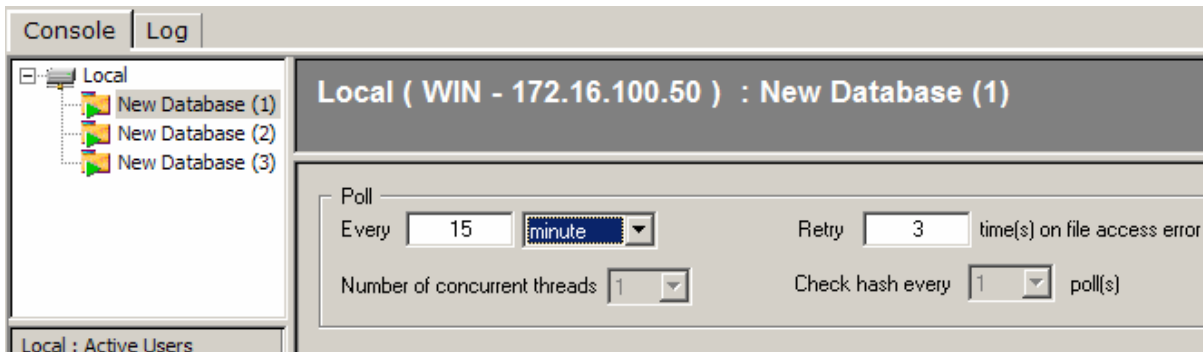


Figure 9 : Customizable polling settings

The poll settings here determine how often polling should occur in the event that an OS trigger fails. If for example, the poll interval is set to one hour then if the OS triggering mechanism fails, polling will only occur every hour. As a reminder, polling occurs directly after an OS trigger irregardless of the poll settings.

In Unix, polling is referred to as the i-node monitoring process. This will be explained in detail later in the document. Hash checks in Unix are performed after a change in the i- node is detected or after x number of i-node monitoring cycles where x is set by the user.

Polling utilizes the SHA-1 hashing algorithm from the National Institute of Standards and Technology to calculate hash values

Login Procedure

The agent configuration process requires a connection between the agent and the console either locally or remotely. A login process is required in which the user must login via the console by supplying a set of valid credentials. The credentials are checked against valid accounts in each individual agent database and access is granted upon a successful match.

Each agent possesses its own database of valid account holders.

When an agent is first installed, only an admin account is added with a default name and password.

Multiple Users

The Admin user of each agent may add additional normal user accounts with rights and permissions as set by the admin user.

These normal user accounts allow users to :

- View logs
- Configure agent settings such as poll intervals and alert programs.
- Add/Remove files (only if granted explicit permission by the admin user)

Only the admin account is assigned the right to add new users and edit user accounts.

Login Security

When normal user accounts are added, the admin user is able to restrict logon to either a subnet range or a specific IP address to ensure that clients are logging on from a valid console.

Communication between console and agent are via a Secure Sockets Layer (SSL) channel.

* See Appendix D for a description on Secure Sockets Layer (SSL).

The benefits of a SSL channel is to :

Encrypt username, passwords and configuration data to defeat sniffer attempts.

Ensure that connecting clients with valid IP addresses are indeed genuine clients with the use of digital certificates to confirm authenticity.

* IP address restrictions can be circumvented by means of IP spoofing. Hence the use of digital certificates to confirm authenticity.

These certificates are issued by an internal certificate authority (CA) managed by e-Lock. Each certificate contains a unique organization ID ensuring that only consoles from the same organization can connect to WebAlarm agents. This protects against the threat of unauthorized consoles attempt at connecting to a remote agent.

Integrity Monitoring Engine

The most important feature of WebAlarm is its ability to efficiently and accurately detect file modification. Integrity check methods are platform dependent and vary in Windows and Unix. Backup procedures such as hashing are conducted to verify file change detection. A brief explanation on hashing and algorithms used can be found in the Appendix B.

Windows Integrity Monitoring Engine

An inherent feature of Windows NT/2000 is its ability to detect file changes and fire a trigger resulting in the generation of an event. WebAlarm utilizes this particular feature of NT/2000 to detect file changes and thus requires very little processing power.

Once an event has been detected, verification is performed by comparing the hash value of the current file with the original. In this case, the original is confirmed to be tamper free and the hash value of this original file is compared to that which is suspect.

Hash values of the original files are calculated when new files or folder are added to the database or when new files or folders are uploaded to the server by the system administrator.

Although hash checks are performed to verify the authenticity of an OS trigger, they also serve as a backup method in the event of OS trigger failure. Hash checks in Windows are often referred to as polling and the polling interval is the interval in which hash checks are performed. Polling intervals are custom configured by the system administrator and can range from one second to a day.

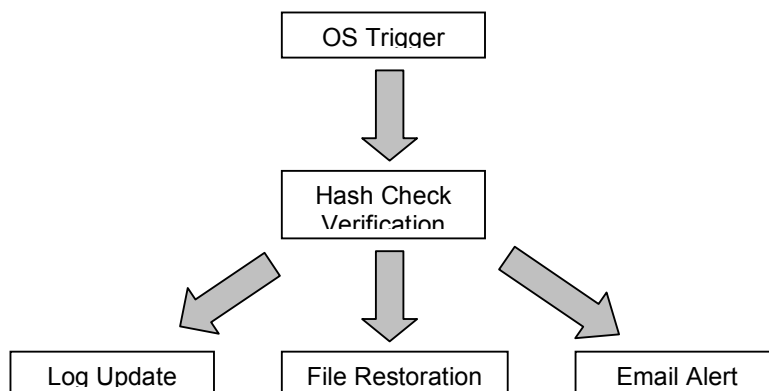


Figure10 : Windows WebAlarm agent procedure in the event of unauthorized file modification.

Unix Integrity Monitoring Engine

Unix machines do not provide an operating system trigger function and therefore as a quick monitoring method, i-node monitoring was implemented. In a Unix system, each file is represented by an i-node. An i-node contains all the information about the file such as type, permissions and modification time.

Each field in the i-node table is allocated a limited amount of space allowing it to retain its size although data within each field is subject to change.

On a Unix platform, WebAlarm monitors certain relevant i-node fields to detect file modification. A modified file will often be of a different file size and more importantly possess a different modification time. These changes form the basis of the i-node integrity check and allow WebAlarm to detect changes made to any file. Note however that the i-node monitor is means of implementing a fast check system in which any obvious file modification is quickly detected.

i-node monitoring is not thorough since only file attributes are monitored and not the data itself. It is feasible to alter the i-node data to bypass the i-node monitoring engine. Polling is therefore a secondary method of monitoring these files for changes in the event the i-node monitor fails to detect file modification.

Performance however is an issue. Polling requires more processing power than simple i-node checks and as a compromise to thorough file monitoring, the polling can be set to occur every 'x' amounts of i-node checks where 'x' is an integer specified by the admin user.

Assuming an i-node change has been detected, a hash value comparison of the suspect file with the original is performed to verify the i-node change. This happens irregardless of the hash interval settings. Contradicting hash values will confirm data alteration and the original file will be restored from backup including its original i-node values.

Type*	Device Number
Permission*	Access Time
Ownership*	Modification time*
File size*	i-node modification time*
Number of links	

Figure 11 : i-node contents

- The relevant i-node fields monitored by the Unix agent are marked with asterisks

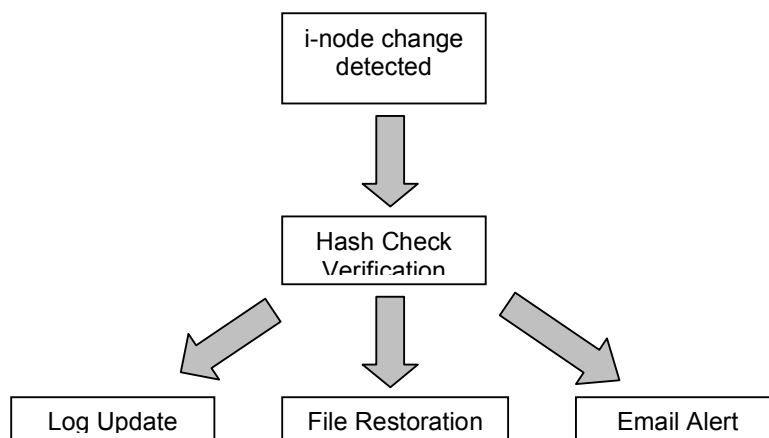


Figure 12 : Unix WebAlarm agent procedure in the event of unauthorized file modification.

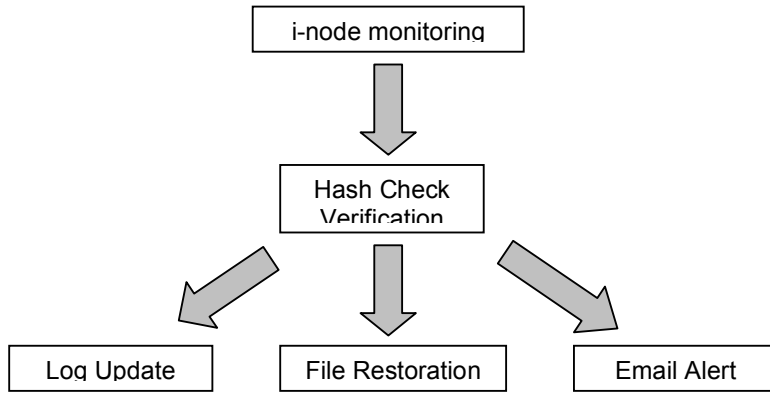


Figure 13 : Hash checks act as a backup in the event that i-node detection fails.

In the event that a skilled hacker is able to modify data in a monitored file as to retain the file size and alter the modification time, the i-node monitor will not detect a change since in i-node monitoring modification detection is based only on file attributes and not the data itself.

Provided the hash checks were set to 1 every poll, a hash check will be performed after each i-node monitoring cycle successfully detecting the file alteration.

Figure 10 above illustrates this.

If hash checks were set to run every 'x' number of polls then in the scenario above, successful detection will only occur after 'x' number of polls. Figure 11 illustrates this.

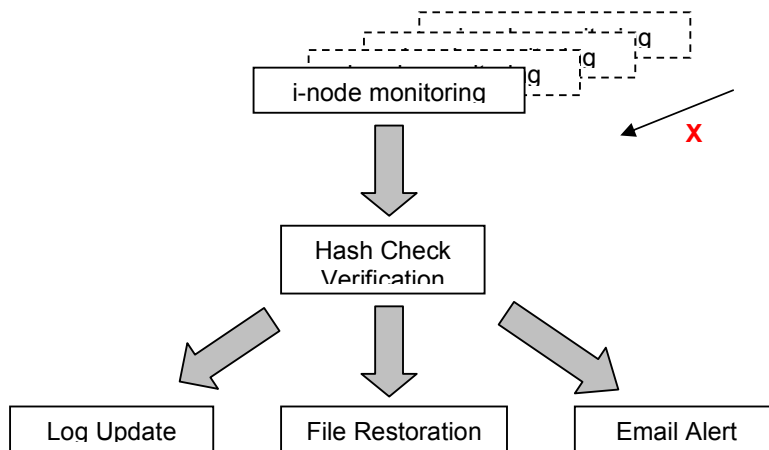


Figure 14 : Hash check is run after **X** number of i-node monitoring cycles

Recovery Options

WebAlarm provides four different recovery options :

- 1) Recovery
- 2) No Recovery
- 3) Alternative Page
- 4) Update
- 5) Log Monitoring

Recovery

By selecting the recovery option, all tampered files will be recovered with an exact copy of the original which is stored in the backup folder.

No Recovery

The 'No Recovery' option will not recover tampered files. It will only alert the administrator that a file has been tampered.

Alternative Page

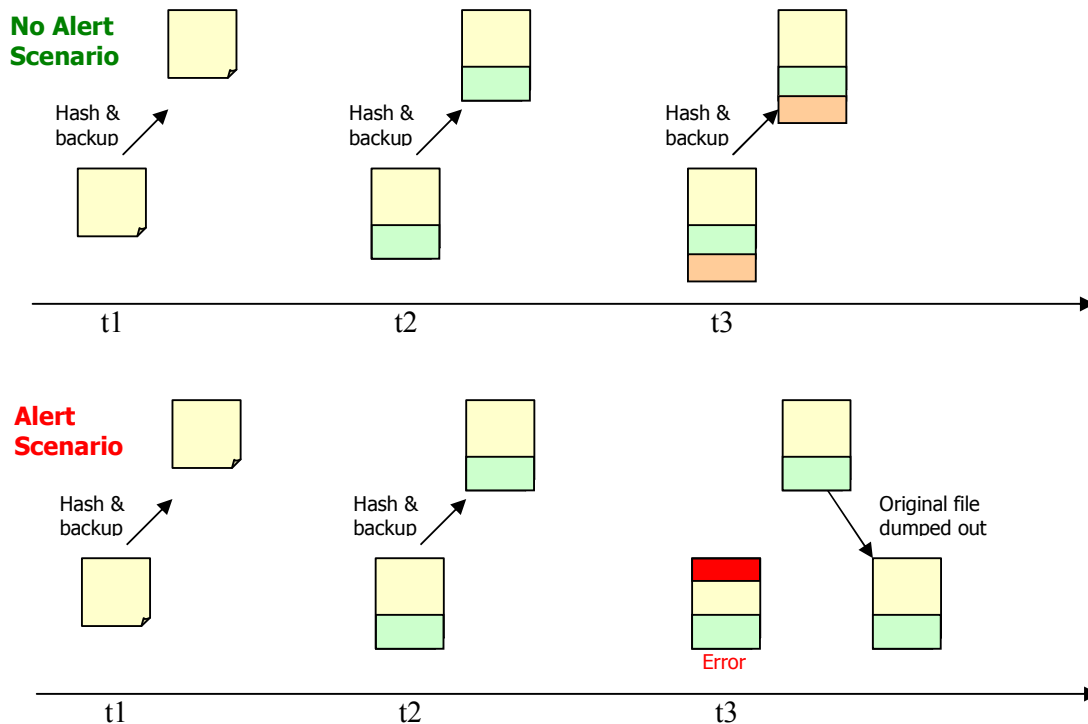
The alternative page option consists of either alternative page or alternative image. When the alternative page option is checked, the user must select an alternative page. When file tamper has been detected, WebAlarm will replace the tampered file with the contents of the alternative page. Alternative Image will replace tampered images with user selected images.

Update

Files and directories are not monitored and protected for this option. Choosing this option ONLY enables Update Management Agent (UMA) to map and update files and directories remotely..

Log Monitoring

The log monitoring detection differs from the normal file and directory monitoring by calculating the hash signature excluding the latest incremental changes to the file. Please see the illustration below:



Restoration Processes

Windows Restoration Process

On a Windows platform, two file integrity methods are employed. The OS trigger is by default the most instant followed by hash checks or polling. As a reminder, the hash checks or polling intervals in Windows can be customized to suit the needs of the user.

Upon file modification (1), providing the particular file has been selected as a monitored file, an OS trigger will be generated instantly (2). WebAlarm detects this trigger and performs a hash check as means of OS trigger verification. The process of the hash check begins with a hash calculation of the monitored file (3).

The newly calculated hash value is then compared with that of the original which was calculated at the time the files were selected or uploaded (4). If the two values differ then WebAlarm pulls the original file from the backup folder and replaces the altered copy with the original (5). Simultaneously the log file is updated with details of the hack attempt and an email sent and/or program run.

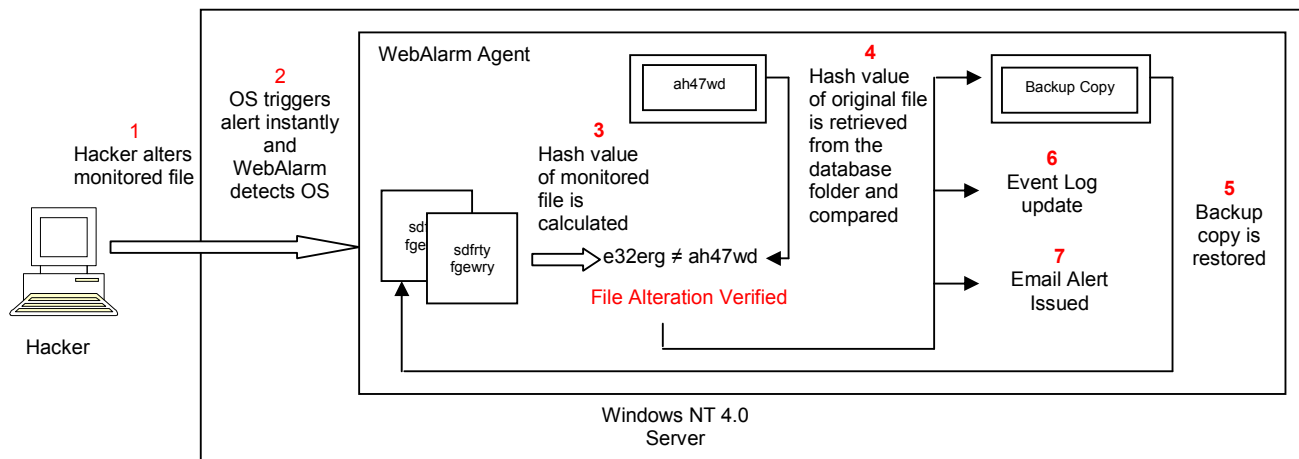


Figure 16 : Overview of the Windows Restoration Process

Unix Restoration Process

The restoration process within Unix is similar to that of Windows. The only difference lies in the integrity check method. The OS trigger method is replaced by the i-node monitoring method in Unix. In Unix, i-node monitoring is referred to as polling.

Upon detection of i-node change, the agent will perform a hash value check comparing the recently calculated hash value of the current file with that of the original version. If these values differ then the original file is restored from back up and alerts are sent out along with log event updates.

Update Management Agent Configuration

The Update Management Agent can only be configured using the Update Management Console GUI application. Among the options available are the configuration of file/directory mapping, email setting, and initiation of a publishing action.

File/Directory Mapping

Mapping between remote data on monitoring agents and the local data on UMA is done through Update Management Console. When this option is selected, a "Folder Browser" will be displayed and any directory/file selected will be set as local data for the monitored database. If it is a directory, any new or

updated file found in the local directory will be captured and transferred to the WebAlarm Agent automatically. If the monitored database/remote directory has 'Recursive' set to 'Yes', any new or updated file in any level or subdirectory will be captured and transferred too.

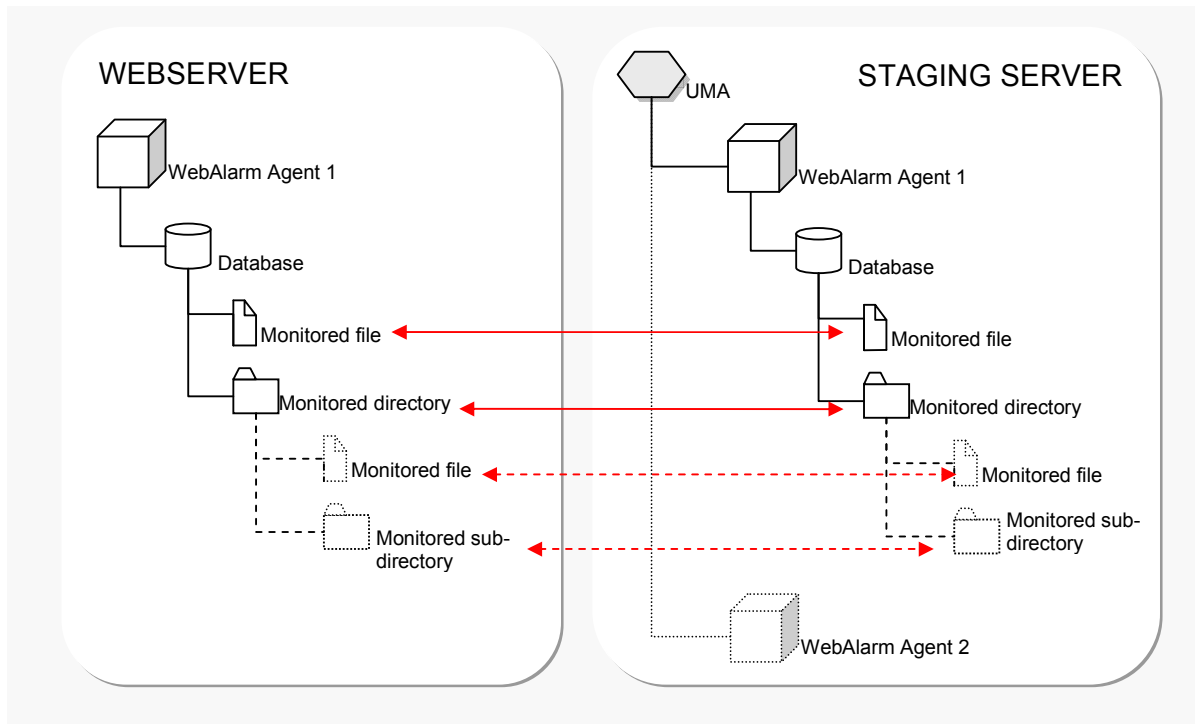


Figure 17 : A illustration showing direct mapping between local and remote data

Email Notification

Email configuration is easy and consists of simply filling in the relevant information. Multiple emails may be sent out by separating each email address with a semicolon. The software limits the number of email addresses to be sent out not by the number of email addresses but by a 256 character limit.

Notification emails are sent out as soon as a data publishing action has been performed. The content of the email would clearly display each file and directory that has either been updated, added, or removed. In the case of the Linux UMA, attribute changes would be recorded as well.

Data Publishing Process

The UMA would transfer the file and directory on the local machine to the linked WebAlarm monitoring agent when any of the following conditions is met:

- i) a new file / directory is added
- ii) an existing file / directory is deleted
- iii) an existing file / directory is modified
- iv) an existing file / directory is renamed
- v) attributes of a file / directory is modified (*Unix only)

The detection mechanism used in UMA is similar to the **Integrity Monitoring Engine** of the Monitoring Agent counterpart. The Windows version makes use of the inherent trigger feature provided by the NT file-system and this produces an instantaneous response to the publishing feature of the Windows UMA. Nonetheless, the changes on the data would again be verified using hash-checking method by the Windows UMA to ensure that the file content is indeed changed. The Linux version of UMA uses a continuous hash-checking method for content changes detection. The publishing response related to the Linux UMA is entirely dependent on the size of the mirrored data. In other words, the more data being mirrored, the longer it takes to be replicated.

Publishing To Multiple Servers

In a multi-servers environment setup, Update Management Agent (UMA) performs content update in a synchronized manner in a way such that it makes sure the same file is updated on all of the servers before continuing on the next file. This allows the servers especially web servers to have consistent content.

Appendix A

Supported Platforms

WebAlarm Agent

- Windows Server 2003 , Server 2008 (runs in 32-bit mode in x64 environment)
- Fedora, CentOS, Red Hat Enterprise Linux
- Solaris 2.6, 2.7, 8, 9, 10 (SPARC & Intel)
- HP-UX 11.0, 11i (PA-RISC), 11i v2 (IPF)
- AIX 5.3/6.1 (PPC)

WebAlarm Console

- Windows Vista and 7
- Windows Server 2003, Server 2008 (runs in 32-bit mode in x64 environment)
- All Windows XP versions

Update Management Agent

- Windows Server 2003 , Server 2008 (runs in 32-bit mode in x64 environment)
- Windows XP
- Fedora, CentOS, Red Hat Enterprise Linux

Update Management Console

- Windows Vista and 7
- Windows Server 2003, Server 2008 (runs in 32-bit mode in x64 environment)
- All Windows XP versions

WebAlarm Report Viewer

- Windows Vista and 7
- Windows Server 2003, Server 2008 (runs in 32-bit mode in x64 environment)
- All Windows XP versions

* Please contact our sales personnel for the latest update on the list.

Appendix B

Hashing

A cryptographic hash is an algorithm that takes an entire message and, through a process of shuffling, manipulating, and processing the bytes using logical operations, generates a small message digest of the data. The output value represents the fingerprint or digest of the message. A cryptographically useful property of a one-way hashing algorithm is that it is infeasible to find two distinct messages that have the same fingerprint.

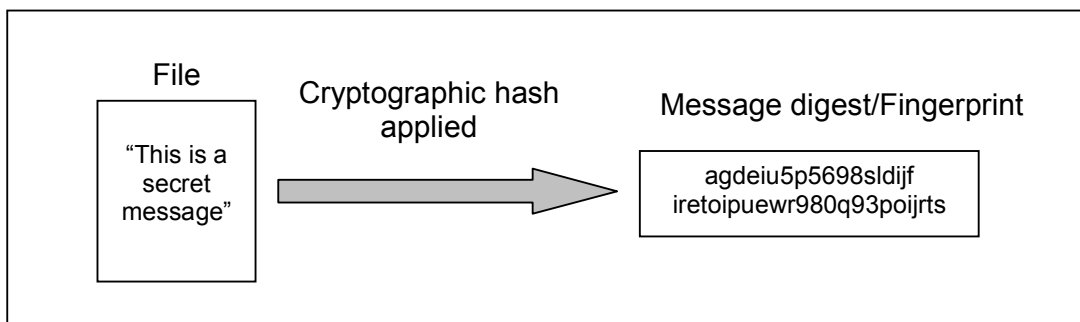


Figure 23 : Brief explanation on hashing

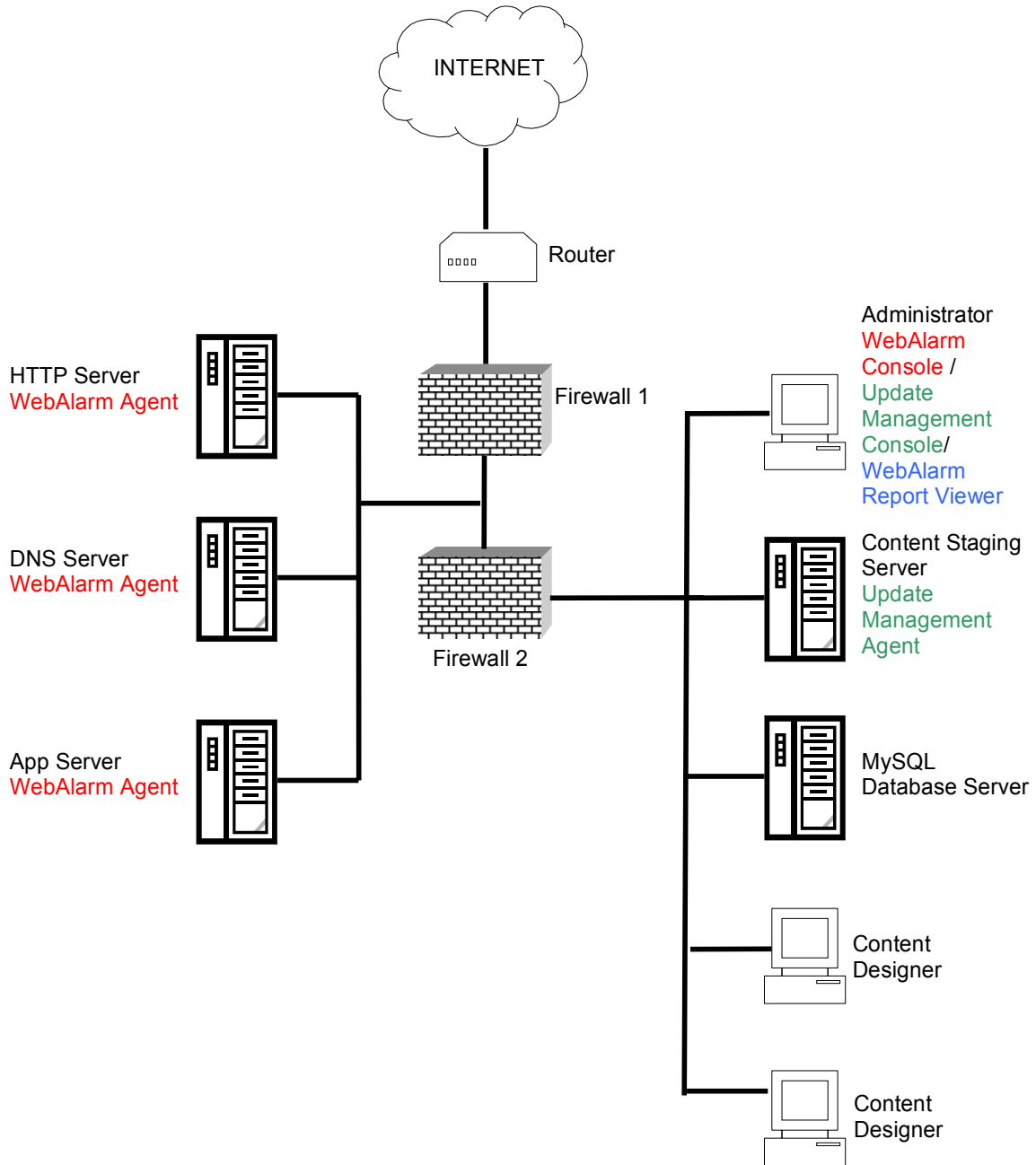
WebAlarm uses this feature of hashing to determine changes made to a file. Since no two files possess the same fingerprint, a file that has been altered will possess a different fingerprint from its original version thus verifying that the file has been tampered with.

A clear message is processed through a one-way "hashing function". SHA-1 is an example of hashing function options provided by WebAlarm.

SHA-1

SHA stands for 'Secure Hash Algorithm' and was developed by the National Institute of Standards and Technology. SHA is a cryptographic message digest algorithm that encrypts a message and produces a 160-bit message digest.

Appendix C Typical Deployment of WebAlarm



Appendix D

Secure Communication with SSL

Designed by Netscape, Secure Socket Layer (SSL) is a protocol providing data security between application protocols. It is an open, non proprietary protocol and has been submitted to the W3 Consortium (W3C) working group on security for consideration as a standard security approach for World Wide Web browsers and serves on the internet.

Here, Secure Socket Layer is employed as means of secure communication between the agent and the console adding an extra layer of security by encrypting logon passwords and configuration commands. It is also authenticated host pcs with the use of digital certificates.

Digital certificates were designed as means of overcoming inherent problems in private/public key authentication and contain the following:

- The certificate issuers name
- The entity for whom the certificate is being issued
- The public key of subject
- Some time stamps

The certificate is signed using the certificate issuer's private key and is the standard way of binding a public key to a name.

Each console and agent are provided with a digital certificate allowing it to establish a secure channel.

These Digital Certificates are based on the X.509 standard and have a public key length of 1024 bit.

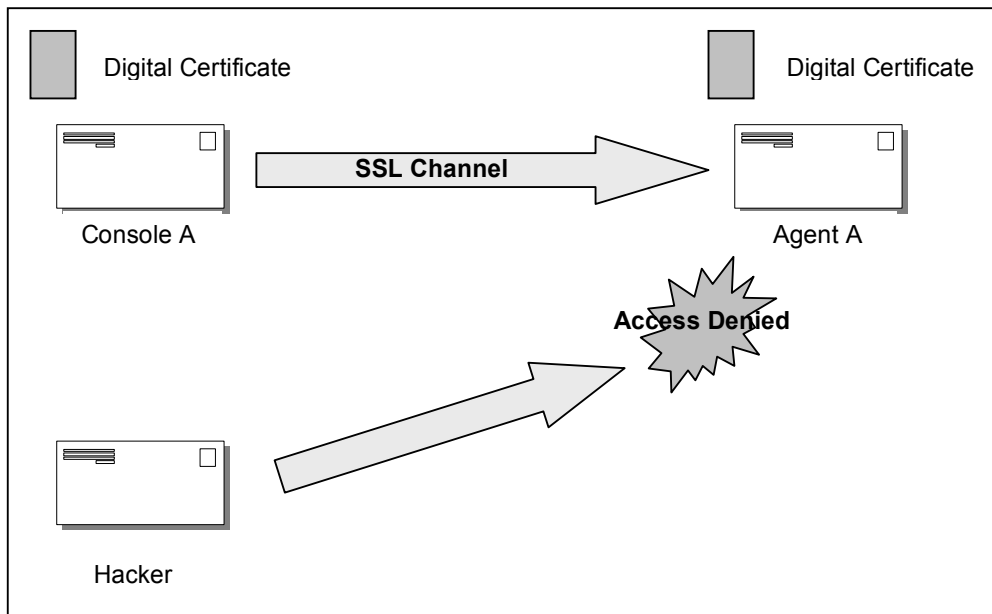


Figure 25: Digital Certificates allow authentication of valid hosts.

Contact Details

For more information please do not hesitate to contact us.

e-Lock Corporation Sdn. Bhd.
Business Suite,
UOA Center,
19A-26-3, Level 26,
No. 19, Jalan Pinang,
50450 Kuala Lumpur,
MALAYSIA

Tel : +603-2166 2981
Fax : +603-2166 2982
Email : info@elock.com.my
Web : <http://www.elock.com.my>

