A Non-Secure Information Systems and the Isolation Solution

Dr. Tai-hoon Kim

Abstract— In this paper, we define Intrusion Confinement through isolation to address such security issue, its importance and finally present an isolation protocol. Security has emerged as the biggest threat to information systems. System protection mechanisms such as access controls can be fooled by authorized but malicious users, masqueraders, and trespassers. As a result, serious damage can be caused either because many intrusions are never detected or because the average detection latency is too long.

Keywords—Intrusion detection, Isolation, Information Systems, Shared Collision, Queuing

I. Introduction

INTRUSION detection is a type of security management system for computer and networks. An ID gathers and analyzes information from various areas within a computer or a network to identify possible security breaches, which includes both intrusion (attack from outside the organization) and misuse (attack from within the organization)[1].The latter case includes seemingly authorized users, such as masqueraders operating under another user's identification (ID) and password, or outside attackers who successfully gained systems access but eluded detection of the method of entry. The methodology of intrusion can be roughly classed as being either based on statistical profile or known patterns of attacks, called signatures or another classification, the anomaly-based. In Anomaly-based, system detects computer intrusion and misuse by monitoring system activity and classifying it as either normal or anomalous.

In this paper we solely concentrate in the Statistical profile-based system. In the following section we define further the Statistical profile-based system, Intrusion confinement thru isolation and the importance. We also present an isolation protocol in the file system.

Importance of intrusion confinement :

Statistical profile-based system compare relevant data by statistical or other methods to representative profiles of normal, expected activity on the system or network [2]. Deviations indicate suspicious behavior. In these systems, there are stringent requirements on not only reporting an intrusion accurately (this is necessary because abnormal behavior is not always an intrusion) but also detecting as many intrusions as possible (usually, not all intrusions can be detected. Based on the assumption the more significant the deviation, the larger the possibility that the behavior of a user is an intrusion, in order to ensure a high degree of intrusion reporting, significant anomaly is required to raise a warning. Moreover, when the anomaly of an intrusion is accumulated, detecting it can still cause a long latency even if it is characterized by significant anomaly. As a result, substantial damage can be caused by an intruder within the latency.

II. SUSPICIOUS BEHAVIOR

Suspicious behavior is the behavior that may have already caused some damage, or may cause some damage later on, but was not reported as an intrusion when it happened. Suspicious behavior emerges in several situations:

(1) In statistical profile-based detection:

(a) in order to get a high degree of soundness of intrusion reporting, some intrusions characterized by gradual deviations may stay undetected. The corresponding behaviors can be reported as suspicious.

(b) For a detection with a long latency, the corresponding behavior can be reported as suspicious in the middle of the latency.

(c) Legitimate behavior can be reported as suspicious if it is sufficiently unlike the corresponding profile.

(2) In signature-based detection, partial matching of a signature can trigger a report of suspicious behavior.

III. RELATED STUDIES

A. Motivation

Photo organizer. Consider an application that scans specified directories for image files and generates photo album les that are written to the same directories. It also generates thumbnail pictures from these files (for creating index files) and has the ability to modify/resize these files. Similar applications that modify images and other media such as audio files are available as freeware on the Internet, e.g., the picture pages [14] package. Safe execution of such applications poses two challenges for sandboxing approaches. Policy selection: Users have to anticipate the resource access requirements of a program prior to its execution, which is often difficult. To overcome this problem, some sandboxing approaches allow changes to policies through runtime prompts to the user: when the sandboxed application violates the initially specified policy, the user is informed and queried whether he/she wants to permit this access. Unfortunately, such repeated prompts lead to .click-fatigue,. as a result of which the user simply grants (or refuses) all subsequent prompts without reviewing them.

fi policy granularity: Users need to develop policies that permit an application to access the resources that it needs, while ensuring that these resources are not corrupted or deleted. For the photo organizer example, such a policy would have to permit .legitimate. changes to image files, as needed for resizing images or including previews, while disallowing other changes. Development of a policy that can capture such legitimate transformations is likely to be hard. Even if such policies can be expressed, enforcement of such policies is likely to be inefficient, if not impossible [18].

Due to these difficulties, sandboxing policies tend to be conservative and often disallow a large class of useful programs such as the picture pages program. In contrast, our proposed approach will permit execution of programs as long as they don't make system changes other than file modification operations. Most applications observe this constraint, and hence they can be run safely using isolation. Moreover, users need not develop safety policies ahead of time. Finally, they have the opportunity to examine the system state resulting due to the execution of the untrusted program, and then decide whether to .keep. or .rollback. These changes. They can use standard system utilities such as find and Diff, as well as arbitrary helper applications such as image viewers, to examine the system state.

Software installation. Users are all too familiar with poorly packaged software that crashes during its installation, or simply does not function correctly. Even worse, the new package may .break. other applications installed on the system. In all these cases, the users are faced with the daunting task of rolling back the installation.

If the package made use of standard package

Management utilities, this rollback is usually not burdensome. However, if the package came as a self-installing executable or as a source package, rollbacks are almost always very difficult. The package may install its files into standard directories such as /usr/local/bin and /lib. It may also modify system configuration files such as /etc/passwd, /etc/mime.types or user profile files such as ./.bashrc. Identifying the exact set of files that were modified is cumbersome. It is also prone to errors as the user does not know the directories where the package installed files, and hence has to search the entire file system. This may result in identifying many files that may have been modified by applications other than the installer. Even if the modified files are identified correctly, rollback is still a hard problem: it is possible only if the user had backed up modified files, but unfortunately, the user did not know ahead of time which files would be modified by the installation.

Using our isolation approach, all of the above problems can be tackled easily. Users simply install the package in isolation. Within this isolation environment, users can then try out the package. They can also examine the files modified by the package, and see if it includes security-critical files, or files that may be used by other packages. (System configuration databases, such as the Red hat Package Manager database, can help in identifying files used by other packages.) If so, they can examine these files to identify the changes made. Alternatively, they can try out the applications that depend on these modified files to ensure that they are not broken. If the users are convinced, after making all these checks, that the new package has been installed correctly and is functioning properly, they can commit the installation.

Otherwise they can discard the installation at this point, the file system state will be as if the installation never took place.

The idea that a network should support concurrent operation for multiple protocols is not new in the Internet. TCP slows down its data generation when it encounters a packet loss. This property is one of the keys to Internet scalability.

However, a protocol can disrupt TCP by not slowing down in response to losses: as TCP sources slow down this TCP unfriendly protocol will saturate the network. To prevent this, non-TCP protocols are expected equip a TCP-friendly feature [16, 23]; the data generation rate must depend on the packet loss rate as TCP does. This property aims to provide a network where multiple protocols can coexist.

This congestion control feature exists at layer 4 because the narrow waist of the Internet is layer 3. That is, there exist multiple layer-4 protocols such as TCP and UDP but most of them use a single layer 3 protocol, IP. Meanwhile, in sensor nets, there exist multiple layer 3 protocols while most systems use CSMA for the MAC protocol. Therefore, the isolation layer provides a shared mechanism above layer 2 that layer 3 protocols can share. Unlike congestion control, which operates along an end-to-end path and manages queue occupancy along flows, an isolation layer operates on single hop wireless communication and manages medium access.

RTS/CTS is another widely studied mechanism that can be used for collision avoidance at the isolation layer. In sensor nets, S-MAC [25] utilizes RTS/CTS exchanges for uni-cast transmissions. These RTS/CTS mechanisms could provide better collision avoidance performance than GTS since they prevent collisions at the current receiver, while GTS prevents collisions at the previous transmitter. In practice, however, RTS/CTS is rarely used. When the interference and communication ranges differ, RTC/CTS is no longer an effective collision avoidance mechanism [25]. In addition, in sensor networks, the control overhead associated with RTS/ CTS becomes significant due to small datagram sizes. More importantly, RTS/CTS cannot easily provide a collision-free environment for the broadcast packets that many wireless protocols depend on. For example, Deluge [17] uses broadcasts for data packets rather than unicasts since one burst of data packets can update all the neighbors of a node. In fact, many smart sensor net protocols exploit this broadcasting nature of the channel. CTP avoids congestion by making nodes overhear their parent's data packets where information on queue depth is piggybacked. Typically, these broadcast packets are sent without control packets to avoid CTS explosions as in S-MAC. Recently, ZigZag Decoding [15] has shown that packet can be recovered from collisions using signal samples. Ideally, this can settle the issue of inter-protocol collision, because perceived collisions will become scarce. However, the need for high processing power and large memory requirements makes it hard to be applied to sensor nets.

Discussing the whole subject of fairness is beyond the scope of this paper. Rather, we focus on the unique properties of protocol fairness. Demers et al. [10] showed that bit-by-bit round-robin fair queuing can be approximated by a packet-by-packet mechanism in wired network. In wired, because there exist only one transmitter per link, the fair queuing mechanism directly chooses which packet to be transmitted. Meanwhile, in wireless LAN, a distributed mechanism is needed. MACAW [5] proposed a fair allocation scheme that controls the channel access using a modified CSMA back off algorithm. Vaidya et al. [26] extended this approachby implementing fair scheduling on multiple flows with different priorities. Protocol fairness requires a combination of intra-node fair queuing and inter-node fair scheduling.

The per-flow fairness is further extended to multichip wireless network. Typically in this case, a flow is defined to be a unique source, destination pair in the link layer [25, 28, 23]. However, clearly this does not fit for protocol fairness. That is, it is hard to divide traffic of protocols to flows of every source, destination pair, because flows are correlated with one another. Alternatively, an end-to-end source-destination pair as the definition of a flow is also considered [13, 19]. However, this is also not a feasible solution for protocol fairness because all network protocols have unique performance metrics. Therefore, we define the "flow" for protocol fairness as a single-hop term, allowing multiple sources to be the source nodes for each protocol. As discussed earlier, this caused the inconsistency of the state to be more persistent, requiring an explicit mechanism that can cure the inconsistency.

The queuing scheme presented differs from common tag-based approaches because it does not update the start-tag when packets are not queued back-to-back. Normally, updating the start-tag serves as a time-window in which the fairness between flows is achieved. However, TinyOS prevents this because the link-layer packet queues are single-depth for each protocol. Thus, although it is possible to enforce immediate re-queuing, this cannot be applied for general protocols.

For example, when CTP packets are delayed by the transmit timer, the link layer has no knowledge whether CTP has more packets. Therefore, we adjust the time window using the decay period: frequent decay results in a short-term fairness, and longer decay period improves long-term fairness.

IV. PROFITABLE-BASED DETECTION

A statistical profile-based detection system, a user Ui accesses the system through sessions. A session of Ui begins when Ui logs in and ends when Ui logs out. A behavior of Ui is a sequence of actions that can last across the boundaries of sessions. A short term behavior of Ui is a behavior that is composed of a sequence of Ui's most recent actions. In contrast, a long term behavior of Ui is also a sequence of Ui's most recent actions but it is much longer than the short term behavior. We assume the intrusion detector is triggered in every m actions (or m audit records), that is, after m new actions are executed, both the current short term behavior and long term behavior of Ui will be upgraded and the deviation of the new short term behavior from the new long term behavior will be computed. When a short term behavior is upgraded, its oldest m actions will be discarded and the newest m actions will be added.

A. Signature-based detection

A sequence of signature-based events leading from an initial to a final compromised state are specify [3].

Each event causes a state transition from one state to another state. We identify a signature with length n, denoted sig(n), as sig(n) = s0E1s1…En , where Ei is an event and si is a state, and Ei causes the state transition from si-1 to si. For simplicity, intra-event conditions are not explicitly shown in sig(n), although they are usually part of a signature.

A partial matching of a signature sig(n) is a sequence of events that matches a prefix of sig(n) , A partial matching is not an intrusion, however, it can predict that an intrusion specified by sig(n) may occur. The accuracy of the prediction of a partial matching, denoted s0E1s1…Emsm, can be measured by the following parameter: Pm, the probability that the partial matching can lead to an intrusion later. Assume the number of the behaviors that match the prefix is Np and the number of the intrusions that match the prefix is Ni, then Pm = Ni / Np

In signature-based detection, the set of actions that should be isolated is defined as follows. Isolating suspicious behavior can surely confine damage in signature-based detection because the behavior that is actually an intrusion will, with a high probability, be prevented from doing harm to the system.

In signature-based detection, a behavior is suspicious if it matches the prefix of a signature but not the whole signature, and Pm of the prefix is greater than or equal to a threshold that is determined by the SSO.

B. Application of Intrusion Confinement support

In signature-based detection, the decision of whether to enforce intrusion confinement on a known attack that is specified by a signature is dependent on the seriousness of the damage that will be caused by the attack and the value of Pm for each prefix of the signature

In statistical profile-based detection, however, it can be tricky to make the decision since degrading the requirement on Re usually can improve Rd, the SSO may want to find a tradeoff between Rd and Re; thus, the cost of isolation would be avoided. However, a satisfactory tradeoff may not be achievable in some systems since the relationship between Rd And Re can dramatically differ from one system to another.

1. Architecture support: The Policy Enforcement Manager enforces the access controls in accordance with the system security policy on every access request [4]. We assume no data access can bypass it. We further assume that users' accesses will be audited in the audit trail.

The Intrusion Detection and Confinement Manager applies either statistical profile-based detection techniques or signature-based detection techniques, or both to identify suspicious behavior as well as intrusions. The detection is typically processed based on the information provided by the audit trail.

Architecture of an intrusion confinement system in information welfare perspective is showed in Fig 1.

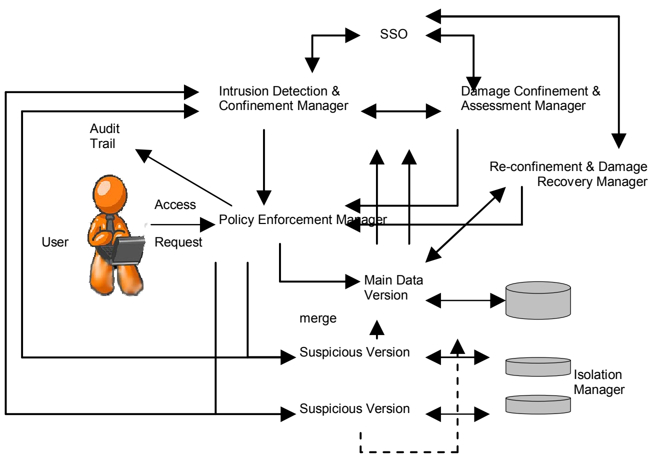

SSO
Intrusion Detection & Confinement Manager
Damage Confinement & Assessment Manager
Audit Trail
Re-confinement & Damage Recovery Manager
Access
Policy Enforcement Manager
User
Request
Main Data Version
merge
Suspicious Version
Isolation Manager
Suspicious Version

Fig 1 Architecture of the Intrusion Confinement System

When a suspicious behavior is detected, the corresponding user is marked suspicious. At this point, first we need to deal with the effects that the user has already made on the Main Data Version because these effects may have already caused some damage. In signature-based detection systems, we can accept these effects because a partial matching is not an intrusion. In statistical profile-based detection systems, if the SSO does not think these effects can cause any serious damage, we can accept these effects; if the SSO thinks these effects can cause intolerable damage, we can isolate and move these effects from the main data version to a separate Suspicious Data Version, which is created to isolate the user. The process of isolation may need to roll back some trustworthy actions that are dependent on the suspicious actions. At this point, we can apply another strategy that moves the effects of these suspicious actions as well as the affected trustworthy actions to the suspicious data version.

Second, the Intrusion Detection and Confinement Manager notify the Policy Enforcement Manager to direct the subsequent suspicious actions of the user to the separate data version. Since we focus on the isolation itself, we can simply assume that when a suspicious behavior starts to be isolated, no damage has been caused by the behavior. Note that there can be several different suspicious users, e.g., S1,…., Sn, being isolated at the same time. Therefore, multiple suspicious data versions can exist at the same time.

When a suspicious user turns out to be malicious, that is, his/her behavior has led to an intrusion; the corresponding suspicious data version can be discarded to protect the main data version from harm. On the other hand, when the user turns out to be innocent, the corresponding suspicious data version is merged into the main data version.

A suspicious behavior can be malicious in several ways:

(1) In signature-based detection, a complete matching can change a suspicious behavior to malicious.

(2) Some statistics of gradual anomaly, such as frequency and total number, can make the SSO believe that a suspicious behavior is malicious.

(3) The SSO can find that a suspicious behavior is malicious based on some nontechnical evidences.

A suspicious behavior can be innocent in several ways:

(1) In signature-based detection, when no signatures can be matched, the behavior proves innocent.

(2) The SSO can prove it to be innocent by some nontechnical evidence. For example, the SSO can investigate the user directly.

(3) Some statistics of gradual anomaly can also make the SSO believe that a behavior is innocent.

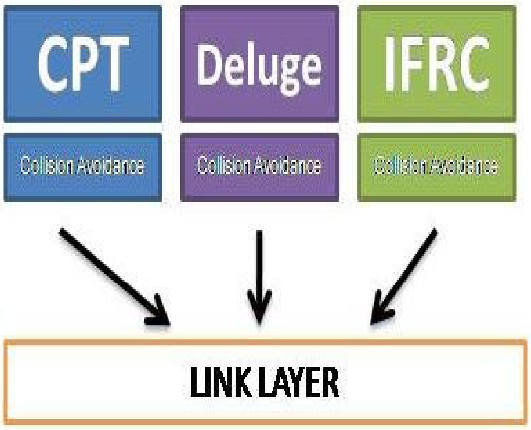
After the damage is assessed, the Reconfiguration Manager reconfigures the system to allow access to continue in a degraded mode while repair is being done by the Damage Recovery Manager. In many situations damage assessment and recovery are coupled with each other closely. For example, recovery from damage can occur during the process of identifying and assessing damage. Also, the system can be continuously reconfigured to reject accesses to newly identified, damaged data objects and to allow access to newly recovered data objects. Interested readers can refer to [5] for more details on damage confinement, damage assessment, system reconfiguration, and damage recovery mechanisms in the database context.

V. ISOLATION PROTOCOL

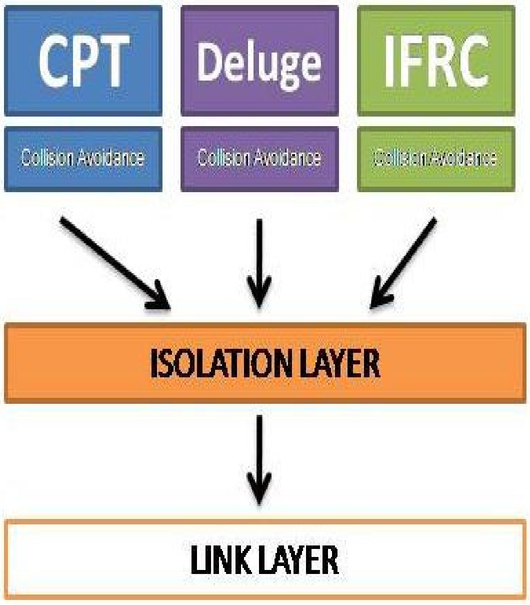
The isolation protocol which is specified as follows is adapted from [6], where a protocol is proposed to detect and resolve mutual inconsistency in distributed file systems. In this protocol, the isolation is processed in terms of each file. When a file fi is modified by a suspicious user Si, the modification and the possible following modifications of Si on fi will be isolated until Si proves to be malicious or innocent. To identify the conflicts between the modifications of Si on fi and the modifications of trustworthy users on fi we associate a version vector with the main version and every isolated version of fi.

To identify and resolve the conflicts between these two versions, we need to first pad the suspicious version vector such that the two vectors have the same set of dimensions. The padding is done by inserting each missed dimension with the value 0 into the suspicious version vector.

The ability to use multiple layer 3 protocols concurrently is a common requirement in large wireless sensor systems.



(a) Current network architecture with individual collision/congestion avoidance mechanisms



(b) Network architecture with isolation layer of shared mechanisms

Figure 2. Individual collision avoidance mechanisms in the current architecture do not work across protocols; a new isolation layer can organize and share collision information. [20]

Rather than use a single layer 3 protocol, such as IP, these networks improve their energy efficiency by using a variety MintRoute [16], CTP [2], MultihopLQI [1], IFRC [17] and other collection protocols build minimum-cost trees to data sinks such as gateways. These collection protocols establish flows up a tree to pull data out of a network. Dissemination protocols, such as Deluge [17] and MNP [18], use leader elections and flurries of broadcasts to push data – e.g. new programs – into a network. Applications often use additional protocols beyond these two basic data flows, such as time synchronization [19] for data time-stamping and geographic [20] or coordinate [6, 12, 21] routing protocols for in-network storage [11, 20].

Typically, however, each of these protocols is designed, developed, and evaluated separately, hoping that it operates as well when other protocols are present. Designing a system with multiple protocols as building blocks can easily run into unforeseen interactions and complications. This discrepancy makes the design phase complicated, performance unpredictable, and debugging difficult.

This paper argues that improving the isolation between layer 3 protocols will greatly simplify the design and implementation of efficient wireless sensor systems. Just as an operating system simplifies building complex system on a single node by isolating processes, a network that isolates protocols would enable each one to be developed, tested, and optimized independently.

Given multiple layer 3 sensor net protocols, there are two ways to achieve protocol isolation. The first is to design all layer 3 protocols such that they respect each other channel requirements. This is similar to the way non-TCP protocols in the Internet are required to be TCP-friendly [16]. The second option is to implement a new mechanism that sits between layers 2 and 3 and ensures that no one protocol interacts badly with the rest minimizing the modifications to the layer 3 protocols. Changing all existing sensor net network protocols would be infeasible. Thus, this paper uses the second route and shows that it is an effective way to achieve protocol isolation.

A. Shared Collision Avoidance Mechanism

Layer 3 protocols often have mechanisms to avoid interference as in Figure 2(a). For example, CTP delays the next transmission for a random period (16_31ms) to give time to the previous packet to be forwarded out of interference range by using a transmission timer that prevents back-to-back transmissions. Deluge can suppress transmissions from neighbors while a node is receiving data bursts [21], since data bursts can easily collide with these transmissions.

These mechanisms do not work across protocols: a node

can transmit non-CTP traffic during CTP's backoff or a neighbor can send non-Deluge traffic to a node that is receiving a binary update. MintRoute [22] has been reported to break when Deluge coexists in a deployment [23]. That is, the bursts of data packets cause excessive collisions with the control packets of MintRoute, collapsing the network topology. Section 5 also shows that concurrent operation of CTP and Deluge decreases the efficiency of both protocols, increasing the end-to-end delivery cost by 24% and 72% respectively.

Many other deployments have also reported low data yields [3, 4, 22, 32]. While the causes are mostly uncertain, we believe many of them to be the interactions between network protocols.

To deal with undesired interactions, some protocols simply assume inter-protocol collisions do not exist. Flush [22] assumes it has complete control of the channel and supports a single flow. While such strict partitioning may be acceptable for application-level workloads, it precludes concurrent services such as management, time synchronization, code distribution, or localization.

VI. Shared Collision Avoidance

Grant-to-send (GTS) is a collision-avoidance mechanism proposed by Choi et al. [8] that can be used to provide isolation between multiple network protocols. Every link-layer packet contains a grant duration field, which grants the channel around the transmitter to the receiver of the packet. That is, upon overhearing or transmitting a packet, nodes can only transmit after the grant duration has expired, and only the receiver of the packet can transmit immediately. The local collision avoidance mechanisms of CTP can be expressed using GTS. Section 2 introduced CTP's transmission timer – a simple mechanism that delays the next packet transmission to give time for the previous packet to be forwarded out of interference range. This mechanism can be replaced by GTS by including a grant of one packet time in each sent packet. Since a grant silences its originator, the recipient of the packet will have time to forward the data up the collection tree.

The lack of a layer 3 collision avoidance mechanism in Deluge has been shown to cause collision losses during a burst transmission. Therefore, an improved version, Deluge [27], augments a request packet to silence neighbors while the data exchange takes place. This approach is equivalent to the grants that GTS supports; a network architecture that employs GTS can use regular Deluge where request packets carry a grant for the duration of the data burst.

GTS is a general mechanism that can be effectively applied to protocols that exhibit correlated packet transmissions. Ideally, the grant included in one packet will prevent collisions for the packet or packets that follow. We believe that because of its generality, GTS can act as a common language that different network protocols can use to communicate collision avoidance information with each other. However, the authors of GTS do not explore what happens when multiple protocols use GTS at the same time. The key difference between using V-Deluge and GTS is that the first affects only packets from Deluge but the latter affects packets from every protocol since GTS is a MAC mechanism.

This property enables GTS to be used as a shared mechanism for the isolation layer, enforcing the grants on all protocols. For example, if Deluge requires silence while a node receives a burst of data packets, all other protocols are held in the isolation layer, providing silence as required. When GTS is used by multiple network protocols concurrently, it is possible that a node is granted by one protocol but sends on behalf of another. GTS specifies neither the destination of the granted packet, nor the protocol it should come from. This is important since otherwise one protocol could easily starve the rest or two nodes could take over the channel. [29]

At the same time, this under specification still leaves space for unfairness in the way channel time is shared between protocols and nodes. The following section discusses this issue in more detail. Ideally, GTS requires overhearing of all packets in the vicinity of a node. In practice, nodes can have their radios turned off during periods of low contention. In such periods, a node may not overhear packets, and hence not update its GTS timer. However, since a node in low-power state is not contending for the channel, collision avoidance is not crucial in this case.

A. *Fair Queuing*

To explain a major challenge fair queuing encounters in a wireless network, we start with a very simple case: two nodes both send packets from two protocols P1 and P2 as fast as possible. If there are no packet losses, this network will achieve node, channel, and transmit fairness. Each node has

an equal chance of acquiring the channel, and on doing so will transmit the protocol with a smaller channel time. Packet losses, however, complicate this situation. If a node fails to hear a transmission, then the channel occupancy tables on the two nodes become inconsistent: the transmitter has incremented channel occupancy but the receiver has not. Because each node is queuing packets based on its own local view of the channel, an inconsistency can lead to a "ping-pong" effect, where the two nodes disagree on which protocol has used the channel less. It can cause low transmit fairness. . For simplicity, a transmission of each protocol increments the value of OpN by one. Suppose that both protocols have accessed the channel N times but, for some reason, N1 has missed two packets from P2. When N1 accesses the channel again, it sends a packet from P2 trying to equalize the protocols. Upon hearing this packet, N2 increments the channel access history for P2 by one. If N2 accesses the channel next, it will send P1, balancing out its protocol table. However, this packet reverts the effort of N1 trying to achieve equality. Thus, at the next chance N1 will transmit P2 again, and this cycle goes on. Eventually, N1 will be biased for P2, and N2 for P1. This communication schedule has perfect channel fairness but low transmit fairness.

Unfortunately, the ping-pong effect is not an edge case that rarely happens; inconsistency can occur from packet losses, collisions, or even different boot times. Furthermore, in multichip networks, each node typically has a different view of the channel as it hears transmissions from a different set of neighbors. Prior work by Luo et al. [28][30] showed how, in the case of unicast flows, a transmitter can embed the flow's channel occupancy in a data packet. Nodes overhearing a packet can use the information to update their table and restores consistency. But in the case of protocol fairness, virtual tags are not easily synchronizable because there can be many transmitters: doing so would require each node to maintain O(np) space, where n is the number of neighbors and p is the number of protocols.

VII. CONCLUSION

In this paper we have shown that a second level in addition to access control intrusion confinement can dramatically enhance the security especially integrity and availability of a system in many situation. It showed that intrusion confinement can effectively resolve the conflicting design goals of an intrusion detection system by achieving both a high rate of detection and a low rate of errors. Developing a more concrete isolation protocols will further be studied in our future research.

VIII. REFERENCES

[1] Graubart, R., Schlipper, L., and McCollum, C. (1996). Defending database management systems against information warfare attacks. Technical report, The MITRE Corporation.

[2] Ammann, P., Jajodia, S., and Liu, P. Recovery from malicious transactions. Technical report, George Mason University, Fairfax, VA. http://www.isse.gmu.edu/□ pliu/papers/dynamic.ps.

[3] Jajodia, S., Liu, P., and McCollum, C. (1998). Applicationlevel isolation to cope with malicious database users. In Proceedings of the 14th Annual Computer Security Application Conference, pages 73–82, Phoenix, AZ.

[4] S. Northcutt, Network Intrusion Detection, New Riders, Indianapolis, 1999.

[5] Ilgun, K.,Kemmerer, R., and Porras, P. (1995). State transition analysis: A rulebased intrusion detection approach. IEEE Transactions on Software Engineering, 21(3):181–199.

10. COSTIN CEPISCA, HORIA ANDREI, EMIL PETRESCU, CRISTIAN PIRVU, CAMELIA PETRESCU, "Remote Data Acquisition System for Hydro Power Plants", Proceedings of the 6th WSEAS International Conference on Power Systems, Lisbon, Portugal, September 22-24, 2006, pp. 59-64

11.RAMÓN MARTÍNEZ-RODRÍGUEZ-OSORIO, MIGUEL CALVO-RAMÓN, MIGUEL Á. FERNÁNDEZ-OTERO, LUIS CUELLAR NAVARRETE, "Smart control system for LEDs traffic-lights based on PLC", Proceedings of the 6th WSEAS International Conference on Power Systems, Lisbon, Portugal, September 22-24, 2006, pp. 256-260

[12] K. Kato and Y. Oyama. Softwarepot: An encapsulated transferable file system for secure software circulation. In Proc. of Int. Symp. on Software Security, 2003.

[13] P. Liu, S. Jajodia, and C. D. McCollum. Intrusion confinement by isolation in information systems. Journal of Computer Security, 8, 2000.

[14] D. Malkhi and M. K. Reiter. Secure execution of java applets using a remote playground. Software Engineering, 26(12), 2000.

[15] N. Provos. Improving host security with system call policies, 2002.

[16]A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of multihop routing in sensor networks. In Proceedings of the First ACM Conference on Embedded networked sensor systems (SenSys), 2003

[17] S. Rangwala, R. Gummadi, R. Govindan, and K. Psounis. Interference-aware fair rate control in wireless sensor networks. In Proceedings of the international conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM), 2006.

[18] L. Wang. MNP: multihop network reprogramming service for sensor networks. In Proceedings of the Second ACM Conference on Embedded networked sensor systems (SenSys), 2004.

[19] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. Knightly. Distributed multi-hop scheduling and medium access with delay andthroughput constraints. In Proceedings of the 7th annual international conference on Mobile computing and networking (MobiCom), 2001.

[20] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In Proceedings of International Conference on Mobile Computing and Networking (MobiCom), 2000.

[21] Y. Mao, F. Wang, L. Qiu, , S. Lam, and J. Smith. S4: Small state and small stretch routing protocol for large wireless sensor networks. In Proceedings of the Fourth USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2007.

[22] A. Woo and T. Tong. Tinyos mintroute collection protocol. tinyos- 1.x/lib/MintRoute.

[23] E. Kohler, M. Handley, and S. Floyd. Designing DCCP: congestion control without reliability. Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM), 2006.

[24] V. Turau, C. Renner, M. Venzke, S. Waschik, C. Weyer, and M. Witt. The heathland experiment: Results and experiences. In Proceedings of the Workshop on Real-World Wireless Sensor Networks (REALWSN), 2005.

[25] K. Xu, M. Gerla, and S. Bae. How effective is the ieee 802.11 rts/cts handshake in ad hoc networks? In

Proceedings of IEEE GLOBECOM' 02, volume 1, pages 17–21, Nov. 2002.

[26] N. H. Vaidya, P. Bahl, and S. Gupta. Distributed fair scheduling in a wireless lan. In Proceedings of the 6th annual international conference on Mobile computing and networking (MobiCom), 2000.

[27] M. Wachs, J. I. Choi, J. W. Lee, K. Srinivasan, Z. Chen, M. Jain, and P. Levis. Visibility: A new metric for protocol design. In Proceedings of the Fifth ACM Conference on Embedded Networked Sensor Systems (SenSys), 2007.

[28] H. Luo, J. Cheng, and S. Lu. Self-coordinating localized fair queueing in wireless ad hoc networks. Mobile Computing, IEEE Transactions on, 3(1):86–98, Jan-Feb 2004

[29] M. BURGIN, Robustness of Information Systems and Technologies, Proceedings of the 8th WSEAS International Conference on DATA NETWORKS, COMMUNICATIONS, COMPUTERS, , 67-72, 2009

[30] CHUNG-PING CHEN*, YING-WEN BAI** and CHENG-HUNG TSAI, Performance Measurement and Queueing Analysis at Medium-High Blocking Probability of Parallel Connection Servers with Identical Service Rate, Proceedings of the 8th WSEAS International Conference on DATA NETWORKS, COMMUNICATIONS, COMPUTERS, , 173-178, 2009

AUTHOR

Dr. Tai-hoon Kim
He received B.E., M.E., and Ph.D. degrees from Sungkyunkwan University. Now he is a professor, School of Information & Multimedia, Hannam University, Korea. His main research areas are security engineering for IT products, IT systems, development