Journal of Information Assurance and Security.
ISSN 1554-1010 Volume 7 (2012) pp. 284-295

Topology Adaptive Computation of Distributed IDS Set for Detecting Attacks on STP

Suchetana Chakraborty, Ferdous A Barbhuiya, Ankush Rai, Arijit Sur, Santosh Biswas and Sukumar Nandi

Dept of CSE, Indian Institute of Technology Guwahati,
Guwahati, Assam - 781039, India

Abstract: **Spanning tree protocol (STP) is a layer-2 network protocol and offers link management with the assurance of loop-free topology for bridged LAN. Despite of its wide applicability, some weak characteristics of STP have made it prone to several attacks. An attacker, exploiting the STP bridge protocol data units (BPDU) messages, can pretend to be a new bridge or populate a false topology change notification over the STP domain for mounting various types of attacks. Distributed IDS is well-known technique used for STP based attack detection. In this paper, a Connected Dominating Set (CDS) based scheme has been introduced to find out a set of IDSs, sufficient to cover the whole STP network. Although every bridge in the network is installed with a small module of IDS, all IDSs are not active at a time. A CDS out of all bridges in the network is computed dynamically to identify the active set of IDSs such that every bridge is directly connected to at least one IDS-activated bridge. An IDS can detect all the exploits mounted to any bridge in its 1-hop neighborhood. Also IDSs communicate with each other in a cooperative distributed environment to detect and verify any changes in the topology which is global with respect to an IDS. The modification or re-computation of CDS set is transparent and done on the fly with the verified changes in underlying topology. The experimental results show that the proposed scheme is able to detect all the STP based attacks.**
Keywords: about six key words separated by commas.

I. Introduction

The recent advances in computer science has made its way to practically every step of our lives that include different industries, education, research, medical, banking, military and many other fields. So the computer network to facilitate communication and sharing of resources globally is a popular field of research for many years. Due to this wide applicability of computer network, the security issues have become a major concern. Communications over the computer network is based on the underling network architecture, designed as a stack of layers. If a service implementation in a lower layer exploits some vulnerability, all the layers above it get compromised too. Therefore the lower layer needs greater attention in terms of security issues. *Layer-2* network infrastructure provides the connectivity and switching functionality, through *layer-2* switch or bridge, in local area network (LAN). The data link layer provides the means for communication between two $hosts$ in a LAN (1; 2). Physical addressing and address mapping are the major duties of data link layer. Traditionally, the *layer-2* protocols have been considered trusted because physical access to the network is under control of a single organization. However, new applications extend the range of *layer-2* networks beyond physical control of a single organization. As more and more broadband service providers deploy their networks based on *layer-2* infrastructure (3; 4), the attacks like MAC flooding, VLAN attacks and Spanning Tee Protocol(STP) attacks, have become more feasible.

A. Background

Spanning Tree Protocol(STP) is a *layer-2* protocol that builds a logical spanning tree to avoid the formation of loops in *layer-2* network. STP dynamically discovers a subset of topology that is loop-free, while simultaneously offering full *layer-2* connectivity for all $hosts$ in the network. Additionally it provides fault-tolerance by automatic reconfiguration of spanning tree topology in case of link or bridge failure in the network (3; 4). Every bridge, participating in the STP topology, is identified by bridge ID (BID, a unique number consisting 16 bit priority and 48 bit MAC address of a bridge). Bridges exchange special Bridge Protocol Data Units(BPDU) messages with each other that allow them to collaboratively compute a spanning tree by running the distributed spanning tree algorithm(STA) as follows.

1) Working of STP

- A bridge with the minimum ID is selected as the root.
- All other bridges calculate the shortest paths to the root.
- For each bridge, the root port (a port of a bridge through which, the bridge has shortest path to the root bridge) is selected. In a tie condition a port with smallest ID is selected as root port.

- For each LAN segment, the designated bridge (the closest bridge from the LAN segment to the root bridge) is selected. In a tie condition, the bridge with the smallest ID is the designated bridge. The designated bridge for a LAN segment is unique and will forwards frames from that LAN segment toward the root bridge. The port that connects this LAN segment to the designated bridge is called the designated port of a bridge.
- The root port and designated port are put into the forwarding state.
- All other ports are blocked.

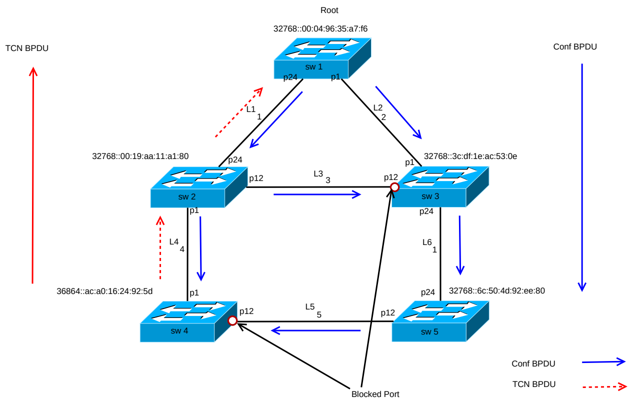

Figure. 1: Conf and TCN BPDU message Flow Direction

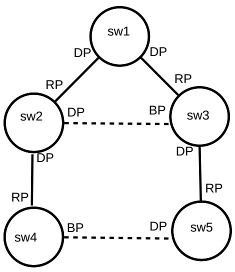

Figure. 2: Spanning Tree

Fig. 1 describes the flow of BPDU messages for the given STP network and Fig. 2 is the corresponding spanning tree, where DP, RP and BP denote $DesignatedPort$, $RootPort$ and $BlockedPort$ respectively. Each configuration BPDU message, *CONF_BPDU*, consists of mainly three fields: root ID, root path cost and bridge ID. Initially every bridge assumes itself to be the believed root in the topology and generates *CONF_BPDU* on each of its port, in every *Hello_Time*. A *CONF_BPDU* that carries root configuration sets both RootID and BridgeID to its ID and the message age to 0. Fig. 3 gives the BPDU frame format. Whenever a bridge receives a *CONF_BPDU* from its neighbor having the superior RootID than its believed root, $(BPDU.RootID < believeRootID)$, the database is updated. The bridge starts forwarding the superior BPDU adding the cost associated with the received port in the root path cost field and increasing the message age. In the event of topology change (such as a link going down or coming up), a bridge that experiences the topology change, generates Topology Change Notification BPDU (*TCN_BPDU*), and forwards to the root bridge in order to notify the root about the change. The root will then notify the topology change event to all other bridges in the network, by setting the Topology Change flag (TC_flag) in the subsequent *CONF_BPDU* messages. The set TC_flag in the *CONF_BPDU*, indicates all the bridges to age out their forwarding tables and if necessary compute the spanning tree algorithm. In a stable STP topology, only the root bridge generates the *CONF_BPDU* in each *Hello_Time*. All other bridges that receive the *CONF_BPDU* in the root port, forward it to the designated port. The bridge ports during the STP configuration go through a number of states (blocking, listening, and learning), before going to the forwarding state. STP operation is the port state transition operation, to avoid the loop formation by transiting some port of a bridge into the blocking state. The state transition of a port from one state to other, during the configuration, is dependent on the STP wide timers. The timers specified by the current root in *CONF_BPDU*s, are shown in Table 1. Additionally, each *CONF_BPDU* contains another time related parameter, *Message Age*, which is basically the hop-count since the root initially originated the BPDU. The root sends the BPDUs with a *Message Age* value *0*, to which all subsequent switches add *1*. In stable state, all bridges keep the current configuration for a period of *Max Age* timer. With every received superior $CONF\_BPDU$, the bridge updates the stored configuration. If within *Max Age* time, no $CONF\_BPDU$ is received, then bridge decides that either the Root Bridge is down or the link to the root is broken. So, the information is aged out for that port and STA is restarted; otherwise the *Max Age* timer is reset. The STP timers basics and the rules to tune the timers are given in (5).

Protocol ID	Version	Message Type	Flags	Root ID	Root Path Cost	Bridge ID	Port ID	Message Age	Max Age	Hello Time	Forward Delay

Figure. 3: STP BPDU Frame Format

Table 1: STP Wide Timers

Timer	Description	Default(sec)
Hello Time	CONF_BPDU time interval	2
Forward Delay	Listening & Learning time duration	15
Max Age	Max waiting time for fresh BPDU	20

2) Attacks on STP

In spite of the broad applicability of STP, some of the properties like lack of authentication in the BPDU messages, stateless nature of STP, slow convergence, timer-dependent port transition of STP etc, made it vulnerable to various attacks (6; 7; 8; 9). An attacker who is intimately familiar with STPs inner workings can impersonate himself as a bridge and can claim an active role in STP topology, with a partially stateful implementation of STP. The attacker exploits lack of BPDU authentication and STA characteristics to select root bridge with the smallest ID. Attacker connected to an active topology, generates well-formed configuration BPDUs per hello time, with bridge ID lower than the current root ID in active topology. This will force STA re-calculation and the attacker bridge will be elected as the root bridge in new active topology with ability to perform all unauthorized activities (10). Moreover once the attacker becomes root, can further accomplish different attacks like *snoop traffic attack*, *refuse to respond to TCN_BPDU*, *generate CONF_BPDU with TC flag set*, *timer changing attacks*, *MiTM Attack*, *priority changing attack* etc. Another set of attacks are possible that compromise network resources. The attacker exploits the limitation of bridge processing power and limited buffer size by generating thousands of unwanted BPDUs per second, creating a DoS condition on the computational power of bridges. There are several versions of this attack, like *flood of configuration BPDUs*, *flood of TCN BPDUs*, *flood of superior configuration BPDUs* etc. Re-designing the existing protocol or mitigating with patches will not be an optimum solution to this problem; because it might affect the standards that are being followed by every network across the Internet. However, a system can be designed that offers continuous monitoring of the network activities and raises alarm if any malicious activity encounters.

B. Related Work

BPDU Guard, Root guard (10; 11), known as *CISCO Solutions*, successfully avoid the STP related attacks. A *BPDU guard* enabled port does not receive any BPDU, hence guards against any STP based attack. Although these techniques might be effective, but certainly oppose the design spirit of STP and clearly restrict the STP network. The dynamic feature or flexibility in *layer-2* network is lost. Moreover, these features impose administrative burdens and complicate the administrative stuffs, because these features must be manually enabled on a port and must need a proper understanding of STP. An improper configuration may leave loops in the network.

Yeung *et al.* have proposed a *partition based switched network* (12), using the special switches, for the modified STP protocol. The problem is addressed by partitioning a STP network into two tier of switching networks, the network infrastructure NI and non-network infrastructure NNI. The partitioning hides the STP operation of the NI from the switching network NNI, that connects end computers, and successfully stops all STP attacks launched from the NNI. The switches that connect two networks NI and NNI, are running two kinds of STP, the normal STP and the modified STP. The special switches provide mapping between these two kind of STP networks. Although the technique successfully stops the STP related attacks launched from the NNI and do not oppose the design spirit of STP (the new switch can be added freely in both NI and NNI). However, the implementation of this technique requires specially designed switches, running the modified STP. Moreover, the technique forces the adaptation of new STP protocol.

Davis *et al.* proposed a mechanism for the authentication of bridge protocol data units (BPDUs) (13), by adding three fields: a *4-byte* nonce, *1-byte* key index, and *20-byte* $SHA-1$ digest. This proposal modifies the STP BPDU, which is clearly against the design spirit of layer-2 network. Moreover, the signing time of a BPDU, is time consuming *(15-47 sec)* that can cause serious impact on the normal STP operation and the convergence time of STP topology.

Intrusion Detection Systems (IDS) is a well-known tool for anomaly or exploit detection based on a set of specified rules (14; 15). IDS is software module that automates the intrusion detection process and detects possible intrusions by monitoring network traffic in promiscuous mode. Distributed IDS (DIDS) consists of multiple IDSs over a large network, all of which communicate with each other, or with a central server that facilitates advanced network monitoring, incident analysis, and instant attack on data or resources. By having these co-operative agents distributed across a network, incident analysts and security personnels are able to get a broader view of the occurrences in their network as a whole. Jieke *et al.* have proposed a *specification based* IDS (16) to model the STP specification as a state machine for detecting the unexpected behaviors of neighbors. Each network element (NE) in the network is extended to keep track of its neighbor's specification and detect an attack if the expected specification does not meet. This scheme is not efficient due to high overhead as every NE runs an individual detection system in addition to keeping the whole state information of all other NEs in its neighborhood. Here the assumption is only the NEs can be malicious. The detection scheme does not account the case where an attacker impersonates a newly added NE claiming the root role. Also no coordination exists among NEs such that an exploit, not direct to any of the NEs, can be detected through joint verification. Furthermore, the scheme is not adaptive to changes in topology as the topology change information is neither updated in local database nor exchanged among all NEs to be reflected globally.

The concept of distributed IDS for STP related attack detection with offline IDS set selection scheme has been reported in (17). In this paper, an extension of that work based on the concept of CDS, with additional IDS set maintenance properties has been reported. In this work an effective scheme has been proposed to compute a set of bridges sufficient to cover the whole STP network. All such bridges form a connected dominating set(CDS) so that every bridge in the network is directly connected to at least one bridge that belongs to the CDS. The IDS modules are activated for only those bridges that are included in the CDS. So each active IDS is able to monitor its $host$ bridge as well as all the bridges in 1-hop neighborhood. Additionally the connectivity among all active IDSs ensures that they can communicate with each other through distributed message-passing environment such that any exploit global with respect to an active IDS can also be detected through cooperative detections. The sufficient number of active IDSs to cover the whole network can be found by computing a minimum CDS. Additionally if a link goes down or comes up, the changes in topology is reflected to the set of active IDSs as the CDS is modified through local adjustment or re-computed on the fly as per the necessity. The term *switch* and *bridge*, are used with identical meaning in this paper.

II. Topology Adaptive Distributed IDS Set Computation

STP is prone to various attacks due to unavailability of security features in STA as well as in STP BPDU packets. This makes STP based attack detection difficult as by exploiting the available bridge ID, an attacker might impersonate himself as a new bridge in the STP topology. The proposed approach is intended to detect this new bridge ID (boot up of a new network device). In order to detect a new booted bridge, the IDSs must need to have the information of all existing network devices (bridges IDs). Moreover, as the STP is adaptive and self-learning protocol, the IDSs must need to be adaptive and self-learning to synchronize with the STP. The simple approach for the detection of root take-over attack is to install an IDS in the STP network that will examine the incoming BPDU and generate the alert, each time the root change incidence occurs. Although this approach results in the root change alert but unable to differentiate between the genuine root and the attacker as the root. One option could be maintenance of a list of all participating bridges in the STP network in IDS by the administrator. The IDS generates the alarm if a bridge, not belonging to this list, becomes the root in the topology. This option is not efficient due to high cost of manual list maintenance effort. Another option could be installing IDSs to all the bridges in the network. This is again not feasible for a large network due to scalability and cumulative operational costs of large number of IDSs.

To limit the number of IDSs and still able to detect attacks effectively, a set of IDSs can be installed in the network such that all the bridges in the network are covered. That means, each IDS is able to receive the direct BPDUs generated by all bridges that are 1-hop neighbors of the $host$ bridge for that IDS. The complete covering of STP network ensures that at least one IDS in the STP network will directly receive the BPDUs generated by the root (the root ID is same as sender ID and message age is zero) and will be able to identify any changes in the network. If there is no IDS in the network that is receiving the direct BPDU from the root, will be considered and detected as the root take-over activity. The cover of the STP network can be found by mapping the problem to one that searches for a dominating set for a specified graph. For a graph $G(V, E)$, where V be the set of all vertices and E be the set of edges, a dominating set D, is defined as the set of vertices $D \subset V$ such that for every vertex $w \in \{V \setminus D\}$, there exists at least one vertex $u \in D$ such that $(u, w) \in E$. Connected Dominating Set (CDS) (18; 19; 20; 21) comes up with additional constraint that the vertices in D forms a spanning tree. This means any node in D can reach any other node in D by a path that stays entirely within D. A minimum CDS ensures that the minimum number of connected vertices required to cover the whole graph is defined by the cardinality of the set. To find a MCDS is known to be NP-hard problem. Although there exist a few approximation algorithms or greedy heuristics that can compute a dominating set (22; 23; 21) for a given graph, none of them can assure the optimal result for the set-cardinality to be minimum. Let $G(V, E)$ denotes a STP topology where V be the set of bridges and E be the set of LAN segments that connect those bridges. A set of bridges that form a CDS cover for the graph G can be computed off-line by installing one IDS to every bridge that belongs to the cover. The connectivity among the bridges within the cover set ensures the efficient and trusted exchange of information among them. If a link goes down or comes up, or for any changes in the topology, a new cover has to be computed with additional cost for installation of a new cover set as well as halt in underlying application. This extra cost can be avoided if the cover is computed dynamically as per the requirement without affecting the running application. The proposed scheme is described in three parts, (a) initialization of IDSs through CDS computation which is similar to the method proposed by Wu *et al.*(18); (b) detection of attacks (either direct or cooperative); and (c) maintenance of CDS cover adapting to the topology changes; in the following subsections.

A. Initialization of IDSs through CDS Computation

The proposed scheme assumes that every bridge in the network is installed with one IDS module, but all are not in operational state. Only the IDSs of the bridges that belong to the CDS cover D, are in operational mode. This reduces the overhead of the running cost of all the IDSs as well as suffices the condition of placing enough number of IDSs to detect an attack. During the initialization of the systems in the STP domain, each bridge u exchanges its open neighbor set information $N(u)$, that contains the bridge ID (BID) of all its 1-hop neighbors, with all its neighbors. Thus each bridge maintains two-hop neighborhood information which is necessary for calculating a CDS. Algorithm 1 gives the steps for computing a CDS cover (18). All the bridges marked as T at the end form a CDS cover and corresponding IDSs are activated. Pruning rules are applied, as per the requirement to minimize the cardinality of the CDS set.

Algorithm 1 IDS Initialization by Construction of CDS, where v is a bridge

```
1. Initially mark every bridge v ∈ V as F
2. if ∃x, y ∈ N(v)|(x, y) ∉ E then
3.       Mark v as T ∀v
4. end if
5. Set status ← Active ∀v marked as T
```

Pruning Rules

1. For any two bridges u and v, if $N[v] \subseteq N[u]$ and $ID(v) < ID(u)$, then change the marking of v to F. Note that $N[v] = N(v) \cup \{v\}$.

2. Let $u, w \in N(v)$, such that u, v, w all are marked T. If $N(v) \subseteq N(u) \cup N(w)$ and $ID(v) = min\{ID(u), ID(v), ID(w)\}$; then mark v to F.

A sample STP topology with 9 bridges has been shown in Fig. 4. STP computes a spanning tree with $BID\ 1$ being the root. The computed CDS cover contains three bridges

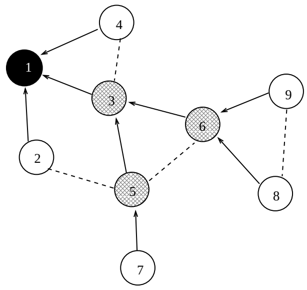

Figure. 4: STP Topology of 9 bridges. $CDS\_Cover = \{3, 5, 6\}$

with ID *3, 5* and *6*. The IDS of these bridges belong to CDS cover are activated. From the analysis of Wu *et al.*(18), it can be observed that the algorithm calculates a CDS in $O(\Delta^2)$ time where Δ is the maximum degree of a bridge in the graph. Also the message complexity is constant. Although this algorithm does not assure to give the smallest CDS, the cardinality of produced CDS is proved to be small. The following Lemmas show that the installation of IDSs through the formation of CDS will cover the whole network of STP domain. Although the proofs are not trivial, can be computed with little effort.

Lemma II.1. *Let the CDS of bridges be denoted by U and for the STP graph $G(V, E)$, the set of all bridges be V. Also let the set of IDSs I_a corresponds to the bridges in U and another set of IDSs I_i corresponds to the bridges in $\{V \setminus U\}$. Then $I_a \cup I_i = V$ and $I_a \cap I_i = \phi$ at any instance of time.*

Lemma II.2. *For every bridge $v \in V, \exists w \in U$, such that $w \in N(v)$.*

B. Detection of various Attacks on STP through DIDS

The set of active IDSs that form a CDS cover, divides the whole STP domain into several overlapping logical zones, each covered by one IDS; called IDS_{Local} corresponding to its local zone. Every bridge that are part of the CDS cover and running an active IDS is called *host* bridge. Remaining bridges are called *peer* bridges. As every bridge (either *host* or a 1-hop neighbors of the *host* bridge), in the STP network is covered by at least one active IDS (IDS_{Local}); at least one IDS receives *CONF_BPDU*s directly from the root. If a bridge is covered by more than one IDSs, then the list of IDS_{Local} is stored in IDS_List. The administrator sets the max priority (default is 32768) for a bridge and the maximum number of bridges (maxNODE, default is 7), in the STP network. Each IDS continuously listens to the incoming *BPDU*s and learns the current STP configuration, like *RootID, SenderID (Bridge ID), believeRootDistance (Message Age)* and STP timers (hello, forward delay and max age), for IDS_init_timer period in a stable STP topology. It is assumed that no attack would occur during IDS initialization period. After the expiry of IDS_init_timer, based on the learned STP configuration an IDS calculates various thresholds and timers. It also then creates an entry in its *LOCAL* table with BID, if the $believeRootID$ is same as $SenderID$ and $beliveRootDistance$ is zero where the entry is not already

Table 2: Timer Value Initialization

$Timer$	$Value$
IDS_init_timer	5 to 10 sec
$TC\_BPDU\_count\_timer$	$3 \times (MaxAge + ForwardDelay)$
$CONF\_BPDU\_count\_timer$	$hello\_time + 1$
$TCN\_BPDU\_count\_timer$	1 sec
Th_CONF_BPDU	$maxNODE * maxNODE$
Th_TCN_BPDU	100
$Th\_HelloTime$	3 sec
$Root\_election\_timer$	60 to 100 sec

recorded. The $BID(P : MAC)$ is a combination of priority P and bridge MAC address MAC. Additionally every active IDS also maintains neighborhood information $N(v)$, where v is the $host$ for that IDS, as received from the corresponding bridge v during the initialization period. It is assumed safely that every IDS has a secret shared key to enable encrypted and authenticated message transmissions among IDSs. Timers specifications are given in Table 2 and Algorithm 2 and Algorithm 3 are the main modules of attack detection.

Algorithm 2 Detection Module

```
if status = Active then
    Listen to each incoming packet P and read the packet header
    if P.type = BPDU then
        Call BPDU_Handler()
    end if
    if P.type = root_verify then
        Call Verifier()
    end if
    if alert = True then
        Raise Alarm
    end if
end if
```

Now an attacker, connected to the STP domain, can send superior BPDUs claiming the root role to mount a root-take-over attack. The detection is possible under three different scenarios depending on the position of the attacker connected to the STP domain.

Scenario 1 (The attacker is connected to a host) : On receiving a superior *CONF_BPDU* from an attacker, the $BID(attacker)$ exists in $LOCAL$ database but not in $N(v)$, where v is the $host$. The IDS_{Local} sets alert for *new_root* on the expiry of $Root\_election\_timer$ as per Algorithm 4, Algorithm 5. It then informs other IDSs through *root_verify* message and sends a *TCN_BPDU* to the newly elected root as a probe. If the attacker is not able to reply to that probe within TCN_probe_timer period, it is confirmed as root-take-over attack according to Algorithm 5. In Fig. 5(i), IDS1 is the IDS_{Local} for the bridges with id *1*, *2* and *3*. IDS1 receives the superior *BPDU* sent by the attacker connected to bridge *2* and finds that although $x \in LOCAL(IDS1)$, but $x \notin N(2)$. So it is reported as root take-over attack detected by direct detection technique.

Scenario 2 (The attacker is connected to a peer) : On receiving a superior *CONF_BPDU* from an attacker, the $BID(attacker)$ exists neither in $LOCAL$ database nor in

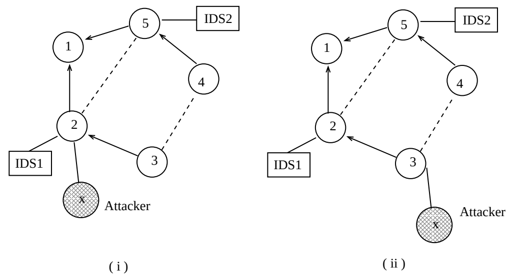

Figure. 5: Root take-over attack: (i) Direct detection by IDS1, (ii) Cooperative detection by both IDS1 and IDS2

Algorithm 3 BPDU_Handler

```
if BPDU.type = 0 then
    conf_BPDU ← True
    if (BPDU.flag).topology_change_bit = 1 then
        TC_flag ← True
        if rootID ≠ currentRoot then
            Update believeRootID, senderID, believeRootDistance
            Call Root_Change_Handler ( believeRootID, senderID, believeRootDistance)
        else
            TC_BPDU ← True
            Start TC_BPDU_count_timer
            if TC_BPDU_count_timer expires then
                if TC_BPDU = True then
                    alert ← True; status ← TC_BPDU_flooding
                    TC_BPDU ← False
                end if
            end if
        end if
    end if
else
    if BPDU.type = 1 then
        tcn_BPDU ← True; TC_flag ← False
        Start TCN_BPDU_count_timer
        tcnBPDU_count + +
        if TCN_BPDU_count_timer expires then
            if tcnBPDU_count > Th_TCN_BPDU then
                alert ← True; status ← tcnBPDU_flooding
                tcnBPDU_count ← 0
            end if
        end if
    else
        alert = True; status ← malformedBPDU
    end if
end if
```

$N(v)$, where v is the *host*. The IDS_{Local} sets alert for *new_root* on the expiry of $Root\_election\_timer$ as per Algorithm 4, Algorithm 5 and Algorithm 6. After the expiry of $Root\_verify\_timer$, if no *root_verify* message is received from any other IDSs verifying the root change, it is confirmed as root-take-over attack according to Algorithm 4 and Algorithm 6. In Fig. 5(ii), the attacker with id x, generates superior ***BPDU*** as connected to the bridge *3*. Now as IDS1 receives superior ***BPDU*** and finds that $x \notin N(2)$; it contacts IDS2 through a $root\_verify$ multicast message. IDS2 does not reply as it is not verified locally. So after the expiry of $Root\_verify\_timer$ the cooperative detection mechanism of all the IDSs conclude and raise an alarm for the root-take-over attack.

Scenario 3 (An existing bridge (either host or peer) is compromised) : In this case, an existing bridge, either $host$ or $peer$, itself is compromised. It generates superior *CONF_BPDU*s which are verified by the IDS_{Local} as the $BID(attacker)$ remains in both $N(v)$ and $LOCAL$, where v is the *host* bridge for IDS_{Local} of attacker. An alarm would be raised only if the attacker does not have the stateful implementation of STP and thus fails to reply to TCN_BPDU sent as a probe by the IDS_{Local}.

The proposed scheme can detect an exploit either local to a bridge or through the cooperative detection schemes of all the IDSs in the STP network. Depending on the specification of several timers, a set of attacks that exploit STP timers can also be detected and avoided in the proposed scheme.

C. Maintenance of the CDS Cover

To adapt to the self-learning nature of STP, the proposed scheme is able to handle the changes in underlying topology. As every bridge in the STP domain is covered by at least one IDS, any changes in topology, either in terms of bridge or link, is experienced by an IDS. Therefore, it then sends the *TCN_BPDU* towards the root which the root acknowledges in subsequent BPDUs. An alarm is raised by the concerned IDS whenever a new root is boot up in the network. Sometimes a false alarm can be raised by an IDS depending on the STP topology as per the specified rules. So with the conformation of administrator a genuine change in topology can be distinguished from the real attack. For a genuine change in topology the CDS cover becomes wrong as some of the bridges might become uncovered or some IDSs remain activated unnecessarily. Naturally a dynamic maintenance of the CDS cover

Algorithm 4 Root_Change_Handler

if $believeRootID < currentRootID$ **then**
  Start $Root\_election\_timer$
  if $Root\_election\_timer$ expires **then**
    $superiorBPDU\_count \leftarrow 0$
    $currentRootID \leftarrow believeRootID$
    if $currentRootID \in LOCAL$ **then**
      Call $Direct\_Detection\_Module$(currentRootID)
    else
      Start $Root\_verify\_timer$
      if $Root\_verify\_timer$ expires **then**
        if $verified\_flag = False \wedge new\_root\_flag = True$ **then**
          $alert \leftarrow True$; $status \leftarrow root\_take\_over$
        end if
      end if
    end if
  end if
  $superiorBPDU\_count + +$
  if $superiorBPDU\_count > maxNODE$ **then**
    $alert \leftarrow True$; $status \leftarrow superiorBPDU\_flooding$
  end if
  if $BPDU.priority < maxPriority$ **then**
    $alert \leftarrow True$; $status \leftarrow invalid\_priority$
  end if
  if $believeRootID = senderID$ **then**
    if $believeRootDistance = 0$ **then**
      Add entry to $LOCAL$ for $senderID$
    end if
    if $believeRootDistance > 0$ **then**
      $alert \leftarrow True$; $status \leftarrow malformedBPDU$
    end if
  end if
end if

Algorithm 5 Direct_Detection_Module

if $currentRootID \in N(u)$ **then**
  $verified\_flag \leftarrow True$
else
  $new\_root\_flag \leftarrow True$
  $alert \leftarrow True$; $status \leftarrow new\_root$
  Start TCN_probe_timer
  if TCN_probe_timer expires **then**
    $alert \leftarrow true$; $status \leftarrow root\_take\_over$
  end if
  Send TCN_BPDU
  if $currentRootID \in N(u)$ **then**
    $alert \leftarrow true$; $status \leftarrow priority\_change$
  end if
  Send $root\_verify$(currentRootID, verified_flag, new_root_flag) to all other IDSs
end if

Algorithm 6 Verifier(currentRootID, msg_verified_flag, new_root)

if $msg\_verified\_flag = True$ **then**
  $verified\_flag \leftarrow True$
end if
if $new\_root = True$ **then**
  $verified\_flag \leftarrow False; new\_root\_flag \leftarrow True$
end if

would be necessary to provide constant security effectively. Also the modification or re-computation of cover set on the fly avoids the extra cost incurred due to manual activation of a new set of IDSs and halt in running application. The proposed scheme tries to modify the cover set, by activating some IDSs and inactivating some others, through local adjustment, once the topology change has been accepted by the administrator. When an IDS transits from active state to inactive state, it waits enough time (until the expiry of IDS_reset_timer) before closing all operations so that no data is lost or get stale in this process. Algorithms for cover maintenance is given in the following subsections under different topology change conditions.

1) A new bridge comes up

Let a new bridge with ID Z is connected to the STP topology. Whether Z will be added to the existing cover or not depends on the following conditions.
Case 1 (Z is added to a host bridge) : As per Fig. 6, Z will be directly covered by the IDS_{Local}; so remains a peer.
Case 2 (Z is added to a peer bridge) : As per Fig. 7, Z will not be directly covered by the IDS_{Local}; so the bridge to which Z is connected, becomes a $host$. Algorithm 7 defines the actions performed when a new bridge is connected to the network.

Algorithm 7 New Bridge Z Added to a Bridge u

Z broadcasts *Hello(Z)*
$\forall v \in N(Z), N(v) \leftarrow N(v) \cup \{Z\}$
$\forall v \in N(Z)$ broadcasts *Hello(N(v))*
if $u = host$ **then**
  Mark Z as F
end if
if $u = peer$ **then**
  mark u as T
  $Cover \leftarrow Cover \cup \{u\}$ where u is marked T
  $status \leftarrow Active\ \forall u \in Cover$
end if

2) A bridge Z, connected to u, goes down

Let an existing bridge with ID Z and connected to another bridge u, goes down in the STP topology. Now whether u is a $host$ or $peer$ decides the modification of cover set as stated in Algorithm 8.
Case 1 (u is a host bridge) : Bridge u will re-compute its status. If their exist two bridges in its neighborhood such

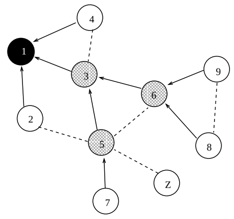

Figure. 6: Bridge Z is added to a host *5*

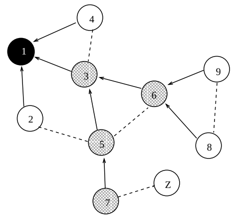

Figure. 7: Bridge Z is added to a peer *7*

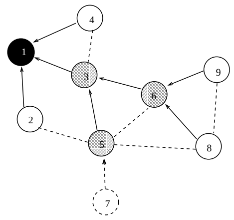

Figure. 8: Bridge *7* goes down, no change in host bridge *5*

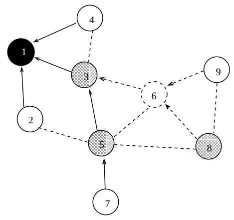

Figure. 9: Bridge *6* goes down, peer bridge *8* becomes host

Algorithm 8 A Bridge Z, Connected to the Bridge u, Goes Down

$N(u) \leftarrow N(u) \setminus \{Z\}$
if $u = host$ **then**
    if $\forall i, j \in N(u); (i, j) \in E$ **then**
        $Counter \leftarrow |N(u)|$
        Send *downAlert* to all $w \in N(u)$ /*On receiving downAlert every bridge sends downAlertACK(IDS_List)*/
    end if
end if
if $u = peer$ **then**
    if $\exists i, j | (i, j) \notin E$ **then**
        mark u as T
        Apply Line 3 and Line 4 of Algorithm 1 to prune
        $Cover \leftarrow Cover \cup \{u\}$ where u is marked T
        $status \leftarrow Active \ \forall u \in Cover$
    end if
end if

Algorithm 9 On receiving *downAlertACK(IDS_List)* from a bridge v by u

$Counter \leftarrow Counter - 1$
if $Counter = 0$ **then**
    if $|IDS\_List| \geq 2 \ \forall w \in N(u)$ **then**
        Start IDS_reset_timer
        if IDS_reset_timer expires **then**
            Mark u as F
            $status \leftarrow Inactive$
            $Cover \leftarrow Cover \setminus \{u\}$
        end if
    end if
end if

that they are not connected, u remains the $host$; otherwise becomes a peer. While going to $Active$ to $Inactive$ state, the IDS on u confirms that all its neighbors are covered through a pair of *downAlert* and *downAlertACK* message as per Algorithm 8 and Algorithm 9. In Fig. 8, the bridge *5* remains the $host$ after bridge *7* goes down as bridge *2* and bridge *3* are not pair-wise connected.

Case 2 (u is a peer bridge) : Bridge u will become a $host$ depending on the condition stated in Algorithm 1. In Fig. 9, the bridge *8* becomes the $host$ and activates its IDS to provide the complete coverage to STP network.

3) A new link (u, v) comes up

Let a new link (u, v) comes up which is experienced by the bridge u. Then u updates all its neighbors. Any $host$ belongs to both $N(u)$ and $N(v)$ recomputes its status. Irrespective of the status of v, if u is the only neighbor of v and v is a new bridge then u becomes $host$. Algorithm 10 gives the formal steps. According to Fig. 10, bridge *8* experiences a new link $(8, 5)$. Bridge *6*, belongs to neighborhood of both *8* and *5*, re-computes its status and becomes a peer as per Fig. 11.

Algorithm 10 A New Link (u, v) Comes Up and u experiences that

u broadcasts *Hello(u)*
if $\exists x | u, v \in N(x) \land x = host$ **then**
    x Sends *downAlert* to all $w \in N(x)$ /*On receiving downAlert every bridge sends downAlertACK(IDS_List)*/
end if

4) An existing link (u, v) goes down

Let an existing link (u, v) goes down which is experienced by the bridge u. Then u updates all its neighbors. Any bridge w, belongs to $N(u)$, recomputes its status as per the rules of Algorithm 1. According to Fig. 12, the bridge *2* experiences a broken link $(2, 3)$. Bridge *1*, belongs to neighborhood of *2*, re-computes its status and becomes a

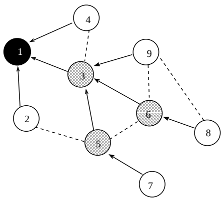

Figure. 10: A new link (8, 5) comes up

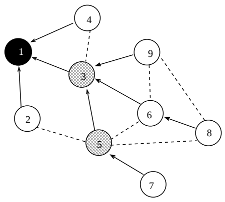

Figure. 11: Bridge *6* re-computes its status and becomes peer

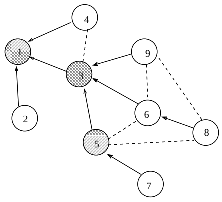

Figure. 12: Link (2,3) goes down, bridge *1* becomes host

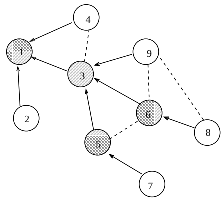

Figure. 13: Link (8,5) goes down, bridge *6* becomes host

host as bridge *2* and bridge *3* are not connected. Same with thecase when the bridge *8* experiences a broken link $(8, 5)$ in Fig. 12 and as a result bridge *6* becomes a *host* as per Fig. 13. Algorithm 11 describes this case.

Algorithm 11 An Existing Link (u, v) Goes Down and u experiences that

1. u broadcasts *Hello(u)*
2. **if** $\exists w | w \in N(u)$ **then**
3. w Sends *downAlert* to all $x \in N(x)$ /*On receiving downAlert every bridge sends downAlertACK(IDS_List)*/
4. **end if**

Therefore, whether a link goes down or new link comes up, or a new bridge added or deleted from the STP network, the change of topology does not affect the CDS cover set. With the help of alarms raised by IDSs and the administrator intervention IDSs can detect an attack. The bridges perform certain local adjustments depending on its present status to continuously provide a cover to the STP domain. It can be proved theoretically that at any instance of time, despite of topology changes, the proposed scheme enables enough IDSs in the network such that all possible attacks on STP can be detected. However, due to increase in space, the detail proofs are not included in this paper.

III. Experimental Result

Different STP attacks and their corresponding proposed detection techniques are implemented and tested using C language. The proposed scheme covers the STP network by activating sufficient number of IDSs. Clearly if STP network is properly covered, the elected root in the topology must belong to the covered network.

A. Testbed

The testbed created for the experiments consists of layer-2 switches running STP. The network architecture is illustrated in Fig. 14. There are 5 switches in the network, labelled

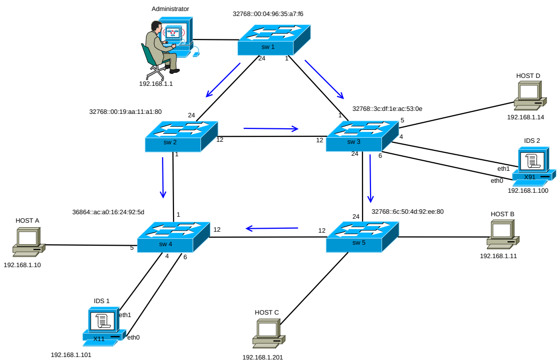

Figure. 14: Network Setup

as *sw1, sw2, sw3, sw4 and sw5*. The attacker's system configuration is pentium 4 processor (1.5 Ghz) with 512 MB RAM and two no.s of Gb Ethernet Network Interfaces Cards, running Linux based OS (Ubuntu 10.10). The attacker can be connected anywhere in the topology. Two IDS (*IDS1* and *IDS2*), form a dominating set and cover all other switches completely in the STP domain. Each IDS has the same configuration as of the attacker, is actively running on Linux systems. Although IDSs were assumed to run as an API module inside every switch in the STP network, due to unavailability of switches with such API, a separate system installed with an IDS API is attached with every switch. Switches are enabled with port mirroring to trace all the transmitted packets. Thus an IDS enabled system can generate STP BPDU packets as well as process all other forwarded packets in neighborhood. Also the IDS systems are capable of sending *TCN probe*, *root_verify* message and alarm signals to the administrator. Depending on the test scenario, the attack generation tool *Yersinia* is deployed on attacker's system and the packet capturing tool *Wireshark* is deployed on the systems connected to the switches. The IDS operations for the detection of STP attacks are as follows.

Table 3: LOCAL Database

IDS	$LOCALDatabase$
IDS1	*32768: 00:04:96:35:a7:f6* *32768: 00:19:aa:11:a1:80* *32768: 3c:df:1e:ac:53:0e*
IDS2	*32768: 00:19:aa:11:a1:80* *32768: 6c:50:4d:92:ee:80* *36864: ac:a0:16:24:92:5d*

Initialization of IDSs Under normal operation of STP, the switch *sw1* with ID *32768:00:04:96:35:a7:f6* will be elected as the root. The direction of *BPDU* message flow is shown in Fig. 14. At the initialization phase, both the IDSs *(IDS1 and IDS2)* learn the current root ID *(32768:00:04:96:35:a7:f6)* and STP wide timers (default, *hello time 2 sec*, *forward delay 15 sec* and *max age 20 sec*). For the above testbed topology, let *maxNODE = 7* and *max_priority = 32768*. The maximum diameter of the network is *dia = 6*. Each IDS calculates the thresholds for STP wide timers as Th_CONF_BPDU and Th_TCN_BPDU are set to $(maxNODE * maxNODE)$ 49 and 100 respectively. The $CONF\_BPDU\_count\_timer$ and $TCN\_BPDU\_count\_timer$ are set to $(hello\_time + 1)$ 3 sec and 1 sec respectively. The TC_flag is set for the duration of $max\ age + forward\ delay$, the TC_BPDU_timer is calculated as $(3 * (max\ age + forward\ dealy))$ 159 sec. Also each IDS initializes its *LOCAL* database by reading the bridge IDs in the incoming *BPDU*s as listed in Table 3. Here *IDS2* does not contain entry for *sw5* (although it is directly connected with *sw3*), because *sw3* is not receiving the *configuration BPDU*s from *sw5*. So *IDS2* is unable to learn *sw5* in its local zone. Based on the connection of attacker to a system with respect to the position of IDS, detection can be either direct or cooperative.

B. Detection of Attacks on STP Topology

An attacker can become the root of the topology by generating the *BPDU*s with root ID less then the current root ID. Additionally it can spoof traffic, refuse to respond *TCN_BPDU*s, set the TC_flag in *CONF_BPDU*s and can change the STP wide timers. The proposed scheme is able to detect all kinds of attacks based on STP. However, only a limited number of attack detection techniques have been shown here with illustration.

1) Root Take-over

The attacker generates *CONF_BPDU* per hello time claiming the root role with superior ID, targeting a single switch.

Scenario 1 Attacker is connected with a $host$ switch *sw4* and launches the root take-over attack with root ID *32768:0:0:c:0:1:8e*. As BPDU from the attacker has the root ID lower than the current root ID *(32768:00:04:96:35:a7:f6)* in Fig. 14, the attacker gets elected as the new root. On receiving superior BPDU, IDSs start their *Root_Change_Handler* module. The *IDS1* will get the BPDUs generated from the attacker with the root ID same as bridge ID and the message age *0* and make an entry in its LOCAL database. On expiry of *Root_election_timer*, the *Root_Verification_Handler* is started. The *IDS2* will wait for the *root_verify* message as, it does not experience any topology change in its local zone. The *IDS1* which experienced a new bridge in its local zone as a root, generates the alert *new_root*. Additionally, it sends a multicast *root_verify* message in IDS domain and send a *TCN_BPDU* as a probe to the selected root. If the attacker does not have the complete stateful implementation of STP protocol, it fails to respond to the TCN probe. The IDS raises alarm for the root take-over attack, if root fails to respond for three consecutive TCN probe, in 1 sec. The deduced spanning tree before and after the root take-over attack is shown in Fig. 15.

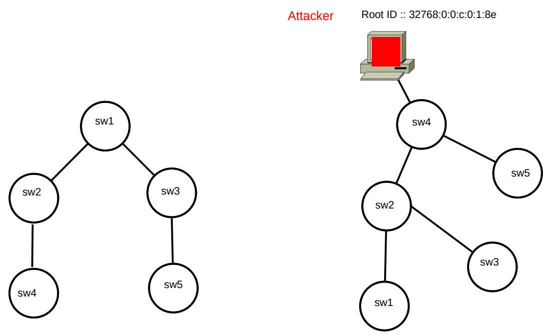

Figure. 15: Spanning Tree before and after Root Take-over Attack

Scenario 2 Attacker is connected with a $peer$ switch *sw5*, which does not belong to the IDS set and launches the root take-over attack with root ID, *32768:0:0:c:0:1:8e*. On receiving the superior BPDU, IDSs start their *Root_Change_Handler* module. In this case both the IDSs will not get the direct BPDUs generated from the attacker and at the end of *Root_Change_Handler* both the IDS will wait for *verify_root* message. As there is no IDS in the network that has experienced the elected root in its local zone, so after the expiry of *Root_verify_timer* all the IDSs will raise an alarm for root-take-over attack.

The *TCN_BPDU* refusing attack will be detected if the root fails to response a *TCN_BPDU*. The *TC_BPDU* (BPDUs with TC flag set) flooding attack will be detected if the TC flag is set for more than *3 * (max age + forward delay)*. And the timer changing attack will be detected if timer specification (calculated based on maximum bridge and network parameters) does not meet. For example, if the attacker sets the forward delay to 60, is detected as crosses the calculated threshold (15). These specification are important to detect the root behavior in case, the current root of the topology is compromised.

2) MiTM Attack

The attacker generates a BPDU per hello time claiming the root role with superior ID, targeting two different switches. Three scenarios can occur depending on the position of attacker with respect to other bridge (whether a *host* or a *peer*).

Scenario 1 The attacker is connected with two switches *sw4*, which is a *host* and *sw5*, which is a *peer*. The attacker launches MiTM attack by impersonating himself as a new root in the topology. The scheme successfully detects the boot up of a new switch as a root in the topology. The *IDS1* will detect the new switch in its local zone and declares the root take-over attack, however it may not conform this attack exactly as MiTM.

Scenario 2 The attacker is connected with two switches *sw1* and *sw2*, where both are *peer*. The new switch boot up will be detected as the root take-over attack, as the attacker does not belong to any of the two local zones. None of the IDSs will experience the elected new root in its zone. Hence each IDS generates the alert after expiry of *Root_verify_timer*. However, in this situation also, the MiTM attack may not be conformed.

Scenario 3 The attacker is connected with two switches *sw3* and *sw4*, where both are *host*. The new switch boot up will be detected as the root take-over attack, as the attacker becomes the root and belongs to both the local zones. Moreover both the IDSs will experience the elected new root in its zone and send a *root_verify* message. If the IDS, that has sent and also received the *root_verify* message, will be able to detect MiTM attack.

IV. Conclusion

In this paper, an IDS based detection techniques have been proposed to detect attacks on STP. The proposed scheme exploits the advantage of Connected Dominating Set (CDS) to compute a cover of bridges for the whole STP domain set. A CDS is computed through distributed message-passing environment and all the IDSs corresponding to the bridges that form the CDS cover are activated for the detection purpose. This avoids the cost of running all the IDSs simultaneously. The cover assures that every bridge is monitored by at least one IDS and thus any attack mounted on a bridge in the STP domain by this scheme through either a single IDS directly or co-operative intervention of all the IDSs. Additionally to adapt to the changes in topology, the proposed scheme also performs certain actions for the maintenance purpose of the IDS cover. new bridges are added or some existing bridges are deleted from the cover on the fly through local adjustment as and when required due to changes in topology. This assures the robustness and effectiveness of the scheme in providing a continuous and correct detection service. Experimental result show how different types of attacks are detected using this scheme.

References

[1] S. T. Dan Hamilton, *Community College LAN Design*, Community College and Vocational Education (CCVE) Deployment Guide ed., Cisco Systems, 2010.

[2] R. Padjen, *Broadcasts in Switched LAN Internetwork*, Internetwork Design Guide Appendix E ed., CISCO System, 2009.

[3] *IEEE Standard for Media Access Control (MAC) Bridges*, ANSI/IEEE Std IEEE 802.1D ed., LAN/MAN Standards Committee of the IEEE Computer Society, 1998.

[4] *IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges*, ANSI/IEEE Std IEEE 802.1D ed., LAN/MAN Standards Committee of the IEEE Computer Society, 2004.

[5] *Understanding and Tuning Spanning Tree Protocol Timers*, Cisco Systems, 2006, document ID: 19120.

[6] David Barroso, Alfredo Andres, *Blackhat EU-Yersinia Framework For Layer 2 Attack*, Blackhat EU, 2005.

[7] E. Vyncke and C. Paggen, *LAN Switch Security What Hackers Know About Your Switches*, 1st ed., ser. Networking Technology: Security. Cisco Systems, 2008.

[8] A. Wong and A. Yeung, *Network Infrastructure Security*. Springer, 2009, ch. Timer Modification Attacks, pp. 143–210.

[9] O. K. Artemjev and V. V. Myasnyankin, "Fun with the spanning tree protocol," *PHRACK MAGAZINE*, 2003.

[10] *Spanning Tree Protocol Root Guard Enhancement*, Cisco Systems, 2007, document ID:10596.

[11] K. Lauerman and J. King, *Spanning Tree Protocol (STP) MiTM Attack and Layer 2 Mitigation Techniques on the Cisco Catalyst 6500*, Cisco Systems, 2010, document ID:10596.

[12] K. H. Yeung, F. Yan, and C. Leung, "Improving Network Infrastructure Security by Partitioning Networks Running Spanning Tree Protocol," *International Conference on Internet Surveillance and Protection*, p. 19, 2006.

[13] S. Whalen and M. Bishop, "Layer 2 Authentication," University of California, Davis (USA), Tech. Rep. XP002387477, March 2009.

[14] J. Mitchell and J. Faust, *InfoSec Reading Room, Understanding Intrusion Detection Systems*, SANS Institute, 2001.

[15] T. M. W, *Information Assurance Tools Report: Intrusion Detection Systems*, 6th ed., Information Assurance Technology Analysis Center (IATAC), 2009.

[16] P. Jieke, J. Redol, and M. Correia, "Specification-based intrusion detection system for carrier ethernet," in *Proceedings of the International Conference on Web Information Systems and Technologies (WEBIST 2007)*, 2007.

[17] A. Rai, F. Barbhuiya, A. Sur, S. Biswas, S. Chakraborty, and S. Nandi, "Exploit detection techniques for stp using distributed ids," dec. 2011, pp. 939 –944.

[18] J. Wu and H. Li, "On calculating connected dominating set for efficient routing in ad hoc wireless networks," in *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*, ser. DIALM '99. ACM, 1999, pp. 7–14.

[19] F. Dai, *Local construction of connected dominating sets in wireless ad hoc networks*, DigitalCommons at Florida Atlantic University, 2005.

[20] R. Misra and C. A. Mandal, "Minimum connected dominating set using a collaborative cover heuristic for ad hoc sensor networks," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. PDS-21, no. 3, pp. 292–302, 2010.

[21] J. Wu, M. Cardei, F. Dai, and S. Yang, "Extended dominating set and its applications in ad hoc networks using cooperative communication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 8, pp. 851–864, 2006.

[22] L. D. Penso and V. C. Barbosa, "A distributed algorithm to find k-dominating sets," *Discrete Applied Mathematics*, vol. 141, pp. 243–253, 2004.

[23] F. Kuhn and R. Wattenhofer, "Constant-time distributed dominating set approximation," *Distributed Computing*, vol. 17, pp. 303–310, May 2005.

Author Biographies

Suchetana Chakraborty has completed her under graduate studies in Computer Science and Engineering from West Bengal University of Technology, India in 2009. She completed her M.Tech from Indian Institute of Technology Guwahati, India in 2011 and is currently pursuing PhD at the same place. Her research interests are broadly in the area of Distributed Algorithms, Wireless and Sensor Networks and Fault-tolerant Computing.

Ferdous Ahmed Barbhuiya received the B.E. Degree in 2001 from Jorhat Engineering College, Jorhat, Assam, India, and the M.Tech. degree in 2007 from the Indian Institute of Technology Guwahati, India. He is a PhD student at Indian Institute of Technology Guwahati doing research in the field of "Intrusion Detection System". He is also working as a Project Fellow in Indian Institute of Technology Guwahati. He has worked as a scientific office in IIT Guwahati during the period of June 2002 to Dec 2007. He has also worked in GlobalLogic Inc from Dec 2007 to July 2009 and also contributed in the conceptualization of products by many startup companies. His current research interests include system and network security, failure diagnosis and discrete event systems, sensor fusion etc.

Ankush Rai received his MTech degree from IIT Guwahati and now working as R & D Engineer at Airvana.

Arijit Sur received his Ph. D. degree in Computer Science and Engineering from Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur. He has received his M. Sc. in Computer and Information Science and M. Tech in Computer Science and Engineering both from Department of Computer Science and Engineering, University of Calcutta. He is currently working as an Assistant Professor at Department of Computer Science and Engineering, Indian Institute of Technology Guwahati. He is a recipient of Infosys Scholarship during his Ph. D. tenure at IIT Kharagpur. He got Microsoft Outstanding Young Faculty Programme Award at Dept. of CSE, IIT Guwahati. His current research interest is Multimedia Security such as Image and Video Watermarking, Steganography and Steganalysis, Reversible Data Hiding and Network Security.

Santosh Biswas received B.E degree from National Institute of Technology, Durgapur, in 2001. He has completed his MS from the department of electrical engineering, Indian Institute of Technology Kharagpur with highest institute CGPA in the year 2004. He has completed PhD from the department of computer science and engineering of the same institute in the year 2008. At present he is an assistant professor in the department of computer science and engineering, IIT Guwahati. His research interests include networking, VLSI testing and design for testability, discrete-event systems and embedded systems. He has published about 50 research papers. He is the recipient of Infineon India best master's thesis award, 2005-2006. His biography has been published in Marquis Who's Who in science and engineering, 2006-2007.

Sukumar Nandi received BSc (Physics), BTech and MTech from Calcutta University in 1984, 1987 and 1989 respectively. He received the PhD degree in Computer Science and Engineering from Indian Institute of Technology Kharagpur in 1995. In 1989-1990 he was a faculty in Birla Institute of Technology, Mesra, Ranchi, India. During 1991-1995, he was a scientific officer in Computer Science and Engineering, Indian Institute of Technology Kharagpur. In 1995 he joined Indian Institute of Technology Guwahati as an Assistant Professor in Computer Science and Engineering. Subsequently, he became Associate Professor in 1998 and Professor in 2002. He was in School of Computer Engineering, Nanyang Technological University, Singapore as Visiting Senior Fellow for one year (2002-2003). He was member of Board of Governor, Indian Institute of Technology Guwahati for 2005 and 2006. He was General Vice-Chair of 8th International Conference on Distributed Computing and Networking 2006. He was General Co-Chair of the 15th International Conference on Advance Computing and Communication 2007. He is also involved in several international conferences as member of advisory board/ Technical Programme Committee. He is reviewer of several international journals and conferences. He is co-author of a book titled "Theory and Application of Cellular Automata" published by IEEE Computer Society. He has published more than 150 Journals/Conferences papers. His research interests are Computer Networks (Traffic Engineering, Wireless Networks), Computer and Network security and Data mining. He is Senior Member of IEEE and Fellow of the Institution of Engineers (India)