"Hive Mind" Design Specification for GENI Security Framework

PI: Sean Peisert, University of California, Davis
peisert@cs.ucdavis.edu

September 17, 2010

1. GENI Introduction

The Global Environment for Network Innovations (GENI) is an NSF-funded, BBN-operated testbed designed to support experimental research in network science and engineering. The goals for GENI are vast: they seek to "understand networks broadly and at multiple layers of abstraction from the physical substrates through the architecture and protocols to networks of people, organizations, and societies," where "the intellectual space surrounding this challenge is highly interdisciplinary, ranging from new research in network and distributed system design to the theoretical underpinnings of network science, network policy and economics, societal values, and the dynamic interactions of the physical and social spheres with communications networks." [GSys] To provide a backbone, or at least a simulation of "the structure, behavior, and dynamics of our most complex systems – networks of networks," GENI seeks "to support a wide range of experimental protocols, and data dissemination techniques running over facilities such as fiber optics with next- generation optical switches, novel high-speed routers, city-wide experimental urban radio networks, high-end computational clusters, and sensor grids." [Gsys] Finally, "GENI suite is envisioned to be shared among a large number of individual, simultaneous experiments with extensive instrumentation that makes it easy to collect, analyze, and share real measurements." [Gsys]

2. GENI Architecture

The core concepts for the suite of GENI infrastructure are programmability, virtualization, federation, and experimentation. Different parts of the GENI suite are owned and/or operated by different organizations, principally including NSF. Those organizations are connected to the GENI infrastructure through a process of *federation*. GENI experiments are an interconnected set of reserved resources, or *slices,* on platforms in diverse locations [Gsol2].

Ultimately, GENI will be highly distributed. That is, though an NSF *clearinghouse* can demand, and possibly enforce globally federated policies, and collections of operations centers around the world (for international partners, countries such as Japan or the European Union) can enforce those policies after they have been negotiated. Currently, GENI exists as a set of enhancements either one of several control frameworks, including Emulab (called ProtoGENI), PlanetLab, ORCA, and ORBIT.

In order for GENI to function as a usable testbed, interconnected with the Internet, and with virtual "slices" running over or near production systems and networks, GENI needs to be both secure and reliable [Gspi1,Gsys]. There need to be facilities to protect GENI experiments from the outside world, to protect the outside world from GENI, and to protect the systems and networks running virtual GENI hosts and slices from attack by GENI experiments themselves [Bis09], despite the increased privileges that such virtual hosts implicitly give.

3. GENI Security

GENI faces significant challenges not faced by typical enterprise networks [Gsec2]. "The scale of GENI, in terms of its number of resources and their geographic spread, makes it an attractive launch pad for large-scale attacks. The national and international attention garnered by the GENI project makes it an attractive target for attacks, for bragging rights if not anything else. The use of GENI for virtually any kind of networking experiment, including long-lived experiments, makes it an attractive platform for hiding and distributing illicit content." [Gsec2]

Our goal is to take steps toward providing a framework for securing such a system. Given its current incarnation of four control frameworks, our current focus is on the *ProtoGENI* architecture and implementation. GENI nodes are composed of hosts, virtual hosts, network devices, virtual network devices, and even cyber/physical devices. Each device has its own security policies, vulnerabilities, and means of enforcing the policies and ameliorating the vulnerabilities. An initial key task is to focus on a framework that will be common to many such devices and the means of implementing them. The purpose of this is to communicate data on a broad scale between devices. The following outline conveys our plan for that architecture, in addition to numerous open questions, many of which we will seek to answer in the course of implementing our prototypes.

Security Needs

Any proposed security architecture must meet the needs of protected organization. This must consider security, operations and any regulatory requirements. The distributed, ad hoc, heterogeneous nature of the collaborative systems that will participate in the GENI system present a unique set of concerns. In some cases, (guest) nodes will be running as virtual machines (VMs) on a host. The host may have multiple VMs, each of which may be assigned to one or more different researchers, from different organizations, from different countries. In other cases the actual physical (non-virtual) host will be assigned as a node for a single user. There may also exist on the network cyber-physical devices (e.g., SCADA devices, sensors, radio telescopes, etc.) that may have no concept of user access privileges at all.

For GENI, operational needs relate to not adversely affecting research and allowing resource contributors to do so without excessive difficulty. In general GENI has few regulatory requirements; however international rules on encryption vary and would likely need to be considered.

The security needs of GENI follow.

- Each resource provider site is independently maintained, and the quality/effectiveness of local security cannot be controlled (particularly physical access to the machines). The security of GENI as a whole must not rely on the security of each site.
- GENI infrastructure systems must be protected GENI Infrastructure systems include the resource providers' systems and those provided at the top Clearinghouse/Registry level (e.g. NSF). Both must be protected against attack coming from Researchers' Slices.
- A key security concern is to provide sufficient assurance to GENI providers that their systems as part of the GENI infrastructure will remain secure according to their security

expectations when they participate in the GENI system. This includes assurances that their hosts are not at risk from attack from GENI Researchers or malicious parties acting through the Researchers, whether through nodes provided by the Resource provider or across the GENI infrastructure from other sites. Also, that the GENI system does not expose them to attacks directly from the Internet.

- Researchers should be provided with assurance that their experiments will be reasonably secure against interference or malicious activity from the actions of others on or through the GENI system. Researchers' data should be protected from loss or disclosure of research data. Additionally, Researchers should be assured that the GENI system will not make the systems they use to access GENI less secure than they would be if conducted on private testbeds.
- We also must ensure that experiments adhere to acceptable use policies, and that they not be used, intentionally or accidentally, to attack hosts on the Internet.
- Performance/quality-of-service (QoS) requirements: Even though other GENI components have potentially conflicting goals, we should consider how the security architecture might adversely affect GENI experiments. E.g., we can allow longer load times, but must be mindful of what chosen method does to experiments and real-time operations.
- Because the GENI architecture is "open," attackers have ready information about what resources are available to exploit. They may not directly be able to learn what information is available to steal, but by obtaining illicit access, they may be able to determine this as well.
- The design lifetime of GENI is 15-20 years. Security changes for this future must be considered. It is logical to predict that the scale (number of providers and provided resources) may grow dramatically. Portions of the GENI system are likely to be "cloud resources". The speed, data storage and transfer capacity will also grow. A highly scalable, decentralized and heterogeneous solution will be needed.
- Any security problem could have strong negative effects, e.g. bad publicity against the system operators and NSF. Security problems should be detected and responded to as quickly and efficiently as possible. Mitigating an attack's effects will be important.
- Providing accurate security alert information to resource providers, for significant activity with low false alarm rate will be importance for acceptance. Alert messages that occur too frequently or are too often incorrect must be strictly limited.

In summary, the security needs include protecting the GENI infrastructure, protecting Researchers experiments on GENI Slices, protecting against attacks targeting the Internet or other connected systems (e.g. systems at Researchers' sites), assuring acceptable use.

Threat Model

To be effective, a security solution must be able to handle a variety of attacks on the system, particularly those more likely to occur or with the highest resulting damage. The table below highlights those of particular importance for the GENI project. A detailed discussion follows.

Table 1: Threat Model

From/To	Own Slice	Another Slice	GENI Infrastructure	Internet
Own Slice	Spread of viral code, etc., will be through Infrastructure.	Ex: data or intellectual property theft	Attempt to hijack resources	Ex: host a spam server or phishing site from GENI.
Another Slice	Similar to Own to Other, but incoming. Ex: data theft, experiment disruption	--	(same as own slice to GENI)	(same as own slice to Internet)
GENI Infrastructure	Compromised infrastructure attempts to disrupt or exploit experiment.	(same as GENI to own slice)	Spread of attack within or between components, across Aggregates, or to Clearinghouse	Misuse of GENI resources to attack Internet sites
Internet	Exploit vulnerability in Experiment's OS and/or applications.	(same as Internet to own slice)	Pass through from Internet to Own Slice	--

- GENI Infrastructure Attacks
    - From Slices
        - Attacks against GENI infrastructure
            - A malicious user attempts to cause a denial-of-service attack against the system. This user selects components at provider sites he wishes to attack, then exploits vulnerabilities in the operating system or GENI software of these to disrupt operations. The user may also simply use GENI to allocate large amounts of resources and then determine a way to reserve it on their own, creating a denial of service. This is a form of *insider attack* where *insider* is defined as someone who has elevated access to a particular resource and *insiderness* is defined by how close that access is [BEF+10].
            - Similarly, a malicious user attacks systems related to GENI's Central Clearing House at NSF
        - Attacks from one slice to another
            - A malicious user with nodes in one Slice, will attempt to gain access to nodes of another researcher's slice by bypassing restrictions against this provided by the *Aggregate managers* and other systems connected to the control network. (e.g., network linking Emulab hosts for management of components). This is possible because it is possible for a user to direct packets over this network (instead of the slice-defined network).
            - Alternatively, a malicious user uses an ARP spoofing attack to redirect packets from another researcher's Slice to steal the others research data.
    - From Internet
        - For researchers to configure experiments, they must communicate with GENI infrastructure systems over the Internet. This makes them subject to the same attacks as any other Internet-connected hosts. Whether an attack was

intentionally targeted against GENI or just a GENI-connected host discovered by chance, any vulnerability in these systems could be exploited. The central GENI systems are likely to be well secured. However, the security patch and configuration level of individual resource providers will vary. These may have been configured by a graduate student who has left, yet the Aggregate was left running, unmaintained for a long time.

    - From Providers
        - A malicious user with privileges at a resource provider's site has access to the sites GENI infrastructure resources could affect experiments using nodes provided by the site or affect the systems of other interconnected GENI resource providers.

- Slice Attacks
    - Unlike Infrastructure attacks, these only affect the nodes in an experiment's Slice or to hosts connected to the experiment (e.g., opt-in users, or discovered public IP address of a node). An extension of this would be to then attack GENI control system hosts as discussed previously. We discuss Slice attacks separately because the exploit mechanisms, dynamics, and effects are different from Infrastructure attacks.
    - Types of attacks
        - Experiment gone bad.
            - Particularly when experimenter is working with distributed processes, worms, etc.
            - Problematic for researcher if unintended, but priority is to detect if it is going to spread outside of the Slice, that is to the Internet or infrastructure.
        - Vulnerable software exploited from external source
            - Here one of the experiment's hosts has an exploitable vulnerability in a publicly exposed service. If attacked and exploited, the Experiment may be affected, or the node and others in the Slice may be used for other malicious purposes.
        - Against Internet targets
            - A malicious user with access to a slice could use it to target Internet sites. If this was a large slice with many nodes from across the globe, a distributed denial-of-service attack against an Internet site could be damaging, to the site and to the reputation of the NSF and associated organizations.
        - Slices used for inappropriate purposes
            - The most obvious example would be if a slice was compromised and used to store illicit material. This would be saved on GENI storage services while the experiment was not in use, and made available when the experiment was reloaded. Clearly this is an unwanted use of NSF's and other partner organizations' resources.
            - The GENI project was intended for networking research, not distributed computing *per se*. While some projects may be allowable, some are likely to be prohibited. Providing a large scale anonymizer network, for example, or running distributed password cracking or cryptanalysis is probably not desired.

Monitoring and Detection

Primary focus is on protecting GENI and its infrastructure. Next would be protecting the Internet from GENI based attacks. While important, protecting slices from attack is less important. Monitoring and detection should keep this in mind.

Monitoring can be either host or network based. However, because of the extensive use of VPNs or other encrypted communication in GENI, network monitoring may not be feasible. Because the focus of this project is on distributed security and the ability of a host monitor to observe network communications in/out of a host, focusing on host based methods is justifiable, particularly when the method employs communication between the distributed monitors. As a distributed set of communicating host monitors, the HiveMind model (see below) will detect the *effects* of network-borne attacks rather than monitoring the network itself. Network monitoring (where feasible), firewalls, DMZs, etc., are all part of the GENI system. We focus on a different, more adaptable method. However, it may be useful to extend distributed monitoring with supplemental information from these localized sources.

Sensor placement.

Monitors must be placed on each host in the system. For infrastructure monitoring this includes all hosts involved in setting up, maintaining, monitoring, and providing resources to a slice. Also those hosts responsible for central operations (Central Clearinghouse, Slice registries, Principal registries, etc).

For within-experiment monitoring, we need to be able to monitor each node in a Slice. These may be actual or virtual hosts. We need to know the set of running processes, used resources, and communications paths in use at any given time.

Sensors can be placed in each node in a Slice, but this provides weak security. Researchers have root access to the node. This would allow them or someone who has gained access to the Slice to disable or modify the sensor. Additional measures may be added to monitor for tampering, however this may not be completely successful. Neighbors can monitor network activity and may under some circumstances detect that the node is misbehaving. But ultimately, this is not a substitute for a secure, protected sensor.

Given this, for virtualized hosts, it will be desirable to modify the hypervisor to provide protected host monitoring. If this is an open system such as XEN, this will be feasible. However for sites wishing to use other virtualization solutions (e.g., VMware) which are closed, commercial products, we would not be able to provide secure monitoring of hosts.

Although intriguing, the use of add-in, hardware-based monitoring solutions are also possible. However, they are likely to be expensive and not well adopted by the community. If the security level of a site's equipment is part of how it is advertised to the user, some researchers may accept minimal monitoring levels; others may prefer more comprehensive security. A sliding scale of protection would be good to provide.

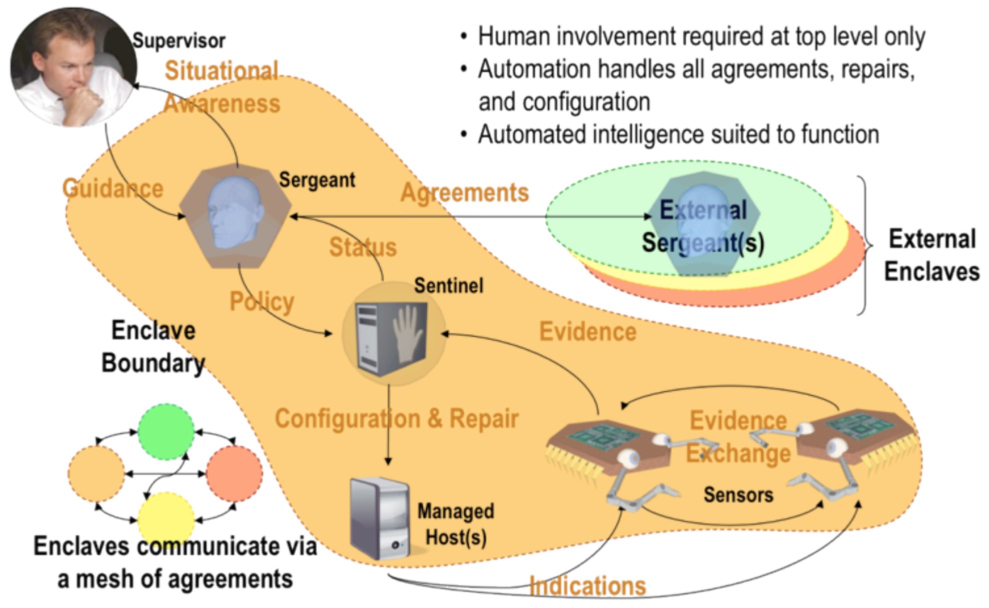
4. HiveMind

Unlike traditional network or host based intrusion detection methods, this project explores a distributed method based on mobile code concepts and swarm intelligence. The goal is to provide a lightweight, distributed detection method that is adaptable to changing threats and that allows suspicious activity to be communicated across hierarchical layers and to humans-in-the-loop who can direct actions when needed.

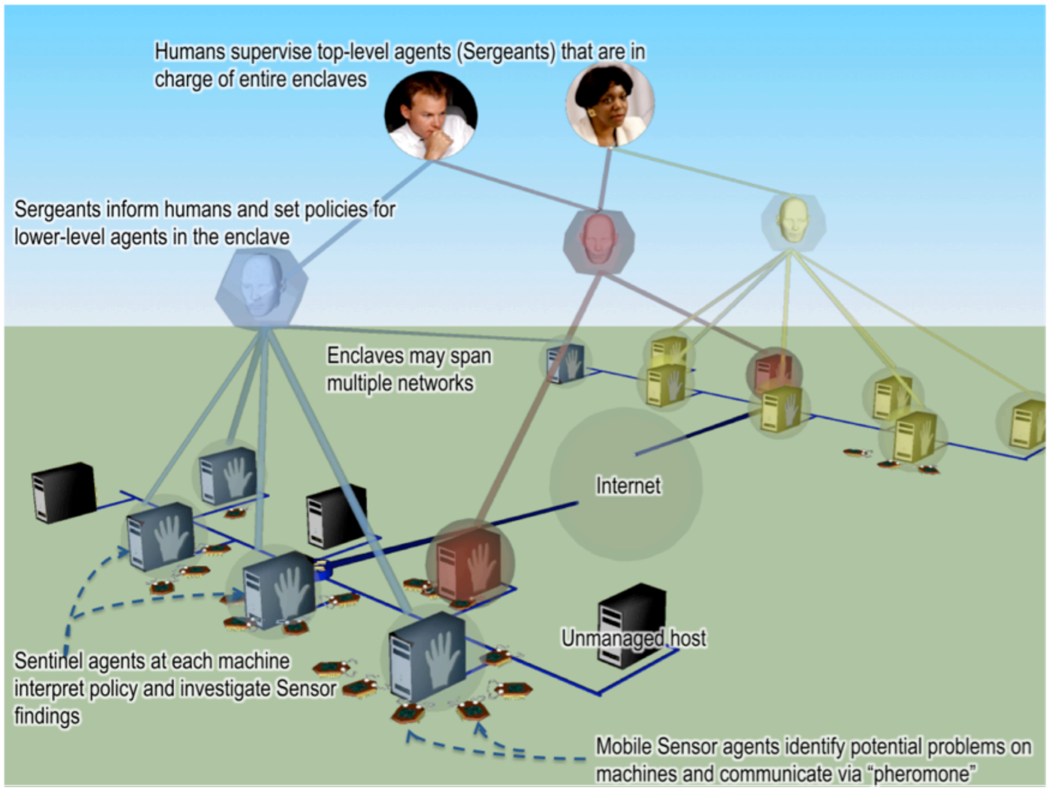
In HiveMind, humans and various types of software agents share the responsibilities of securing an infrastructure comprised of enclaves that belong to member organizations. Figure 1 shows how one human can supervise a multi-*enclave* system with a few enclave-level agents, host-level agents at each monitored machine or group of similar machines, and a large *swarm* of simple mobile agents. Our terminology is as follows:

- Humans function as *Supervisors*. They provide guidance to and receive feedback from one or more enclaves. Action is required of them only when the lower-level agents encounter a problem that requires human involvement.
- Enclave-level agents called *Sergeants* are each responsible for the security state of an entire enclave.
- Host-level agents called *Sentinels* protect and configure a single host, node, or a collection of similarly configured hosts and nodes such as a cluster or storage network.
- Swarming agents, called *Sensors*, are the "ants" that roam from device to device within their enclave searching for problems and reporting to the appropriate Sentinel.

HiveMind's hierarchical structure was inspired by the success of previous research [PNBA06, Smie96, Self88], which suggests hierarchical arrangements of heterogeneous agents. Interposing logic-based rational agents between the humans and the swarm provides a basis for



communication, interaction, and shared initiative. The hierarchical arrangement gives humans a single point of influence that allows multiple points of effect. Figure 2 shows the relationships among HiveMind actors in a conceptual IT system, and the subsequent sections describe the roles of the actors.


Humans supervise top-level agents (Sergeants) that are in charge of entire enclaves
Sergeants inform humans and set policies for lower-level agents in the enclave
Enclaves may span multiple networks
Internet
Unmanaged host
Sentinel agents at each machine interpret policy and investigate Sensor findings
Mobile Sensor agents identify potential problems on machines and communicate via "pheromone"

5. Mapping Digital Ants paradigm to GENI infrastructure

The Digital Ants paradigm fits well with the GENI system, both for monitoring the GENI infrastructure and the Slices independently.

As described earlier, the Digital Ants paradigm uses the concept of an enclave to limit scope and control the flow of information. This will restrict the amount of inter-host communication (ants only range in the enclave), and provide privacy. For example, when vulnerabilities are found at one organization, that organization may not wish to have those disclosed, but reported and mitigated locally. This type of enclave, an Infrastructure Enclave, is the most important to protecting the GENI system.

For this type of Enclave each Resource Provider will have an independent set of Sentinels, one for each host they use either as components or to support the local infrastructure. If these are VM then each VM will have its own Sentinel. Each Sentinel will manage the ants visiting its hosts and communicate with the Sergeant Process associated with the Providers site. Generally, this will correspond to an Aggregate, but may be multiple Aggregates if they are administered together.

The Sergeant will be located at the Provider's site, running on a separate computer. It will oversee the enclave, and report to a Supervisor process which provides command and control functions for local administrators. This makes sense because if an attack is the result of a local misconfiguration in the Provider's system, the problem would be handled locally. It is not necessary (or even desired) to tell all Providers that one host on some Provider's system had file permissions set improperly

Another type of enclave will be Central Clearinghouses or similar sites. These are the highest level GENI systems. They provide functions to validate Researchers and to allocate and maintain Slices. These sites include several registries to Sentinels will run on each involved host and Sergeant host will manage the enclave. As no components are provided, no component monitoring Sentinels will exist.

Infrastructure Enclave Sergeants not only communicate with their Sentinels and report to Supervisor processes, but need to intercommunicate to share information learned about new threats (e.g., ant rules that have been particularly successful). Because the set of Sergeants and the Geography linking them is not predetermined or static, a method for "neighbor Sergeant discovery" is needed. We envision this as being based on current router discovery protocols.

An alternative type of enclave, a Slice Enclave, will be useful for detecting attacks that span a Slice, but do not affect infrastructure systems. This differs from Infrastructure Enclaves in that the hosts involve may be composed of components from many if not thousands of Resource Provider Sites and may involve devices that span the globe. Here a Sentinel Process will run on each node in the Slice. The Sergeant process will not be tied to a specific Resource Provider, but will be located along with the central registries. Significant problems can be reported to the Researcher and to the Central Authority as needed.

Information may need to flow up to the NSF GENI central clearinghouse level. For example, if a Provider's system has been compromised or known to have significant vulnerabilities, we may wish to exclude them from the pool of available resources until such time that the problem has been corrected.

The events that ants process may differ between the Infrastructure and Slice enclaves. Also, the nature of the Geography the ants navigate will be constructed differently. The Geography for Infrastructure Enclaves will be static. For Slice Enclaves, they will be different for each Slice. The location of the hosts may change between experiment runs. The geographic and network topologic distances will differ between runs.

A method for the Slice Enclave Sergeants to discover the Geography and configure the Sentinels will need to be created. As envisioned in GENI, the nodes in a Slice may be dynamic. That is, nodes may join, leave, or change during the run of an experiment. A method for handling this will in the future need to be determined. A method similar to those used for memory management involving dynamic allocation and reallocation of memory is a likely place to start.

A concern is that Aggregates may have very few hosts. When this is true, the power of the Digital Ants paradigm may not be realized. The Enclave must contain more hosts that at the single site. It may be necessary to create meta-Enclaves that span an arbitrary number of Aggregates, perhaps all Aggregates. Each of these would have its own Sergeant. Where best to locate this Sergeant is to be determined. Alternatives include at an arbitrary provider site, at one of several specialized provider sites or at the GENI clearinghouse.

Evidence

Rather than rely primarily on rules and signatures like traditional intrusion detection systems (IDSs), the digital ants agents incorporate low-level event monitors that individually do not signal a problem, but in aggregate and in combination with other security information indicate something suspicious is occurring. Examples are number of page faults, process restarts, packet rates, ports open, etc. These can be based on immediate state or summaries over time, e.g. number connections to X from Y in the last Z time. Also, GENI specific API calls should be monitored. Specific rules may also be used, but they are not the primary detection mechanism. All these types of events, regardless of what or how detected are referred to as *evidence*. By reviewing the collected evidence we are able to determine the significance of the observed activity.

By providing this evidence rather than just alerts from IDS type rules, the Digital Ants can provide more information. This creates a system that can generalize across different types of attacks. By looking at the evidence in aggregate, the Supervisor can see the whole and determine its significance.

For this project we will create a base set of detection items. These will be selected with consideration for the range of attack types discussed above.

For example, assume an Experiment where the Researcher is using a Slice composed of nodes running a service that is vulnerable to a buffer overflow, and that a worm exploiting this vulnerability has infected the Researchers workstation. Once he connects to his Slice, the Worm starts to propagate across the Slice. Because nodes are connected to both the experiment's private network and the Control Network, the Worm begins to target hosts of the GENI infrastructure. The ants see a number of process crashes on the same port and service on multiple hosts. Also ants detecting unusual connections to the control network succeed. Ants specifically looking for these occur in greater numbers on all hosts. Pheromone draws ants looking for other types of evidence to these affected hosts. The evidence from many Sentinels is reported to the Sergeant, and because of the significance of the activity, it is reported to the Supervisor who may take action to shut down the Slice. For this scenario, we need to determine what evidence will be needed to detect the attack and to determine it is significant. This type of analysis will be done for several example attacks.

Because we have two distinct types of hosts, those delegated to experiments and those responsible for infrastructure, we will need to develop evidence detection mechanisms that are relevant to each and to both.

Because it is left up to the Supervisor to determine what this activity means, it is foreseeable that a Supervisor Analytic Process that provides summarization, labeling and significance assessment information will be useful. Such a process would, with human support, add semantic information to the activity. This could be sent to the Sergeants for dissemination to other Sergeants (and on to other Supervisors). Just as the ants use information sharing to improve detection, information sharing between Sergeants and Supervisors can provide significant value.

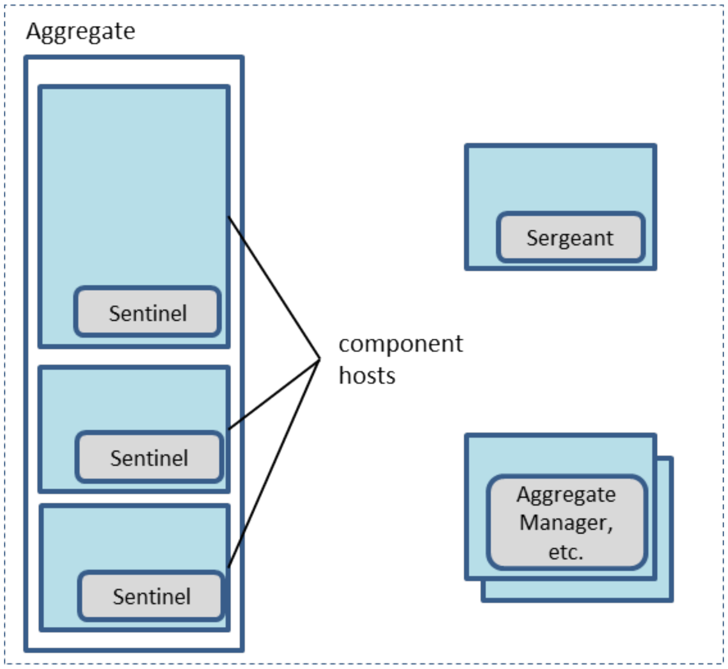
Integration into ProtoGENI

We have chosen to use the ProtoGENI control framework as our implementation target. It is well supported, provides a rich feature set and an infrastructure like that proposed by GENI. We will use it for implementing and evaluating HiveMind for protecting both slices and infrastructure. In this section we will discuss how ProtoGENI could be modified to fully support this security paradigm.

As discussed before there are two ways to monitor the GENI system, Slices and Infrastructure. This is true for a ProtoGENI implementation as well.

Infrastructure Enclaves

As described earlier, we must add a Sentinel Process to each host that provides Control Framework functions, whether real or virtual. For a ProtoGENI system this means one for each Clearinghouse host, one for Aggregate Manager, and one for each Component host provided. Each Aggregate Cluster will have a Sergeant process assigned to it. If too few hosts are present in an Aggregate, it may be necessary to group several clusters and provide a single Sergeant for the group.



The Sentinel process must be able to monitor low-level system information such as system calls, process failures, ARP messages, and IP stack info. Certain evidence may require modifying the OS kernel to provide this. Ultimately, the Sentinel should be a kernel module, or interface with one as a proxy.

The Sergeant process should run on a separate host. It will be receiving information from all Sentinels. For large sites or when the amount of packets exchanged is high, instead of using the primary control network, we may wish to provide a separate Sentinel to Sergeant network.

Question: How many hosts per site are we likely to encounter?
Question: Is this expected increase greatly in the future

Slice Enclaves

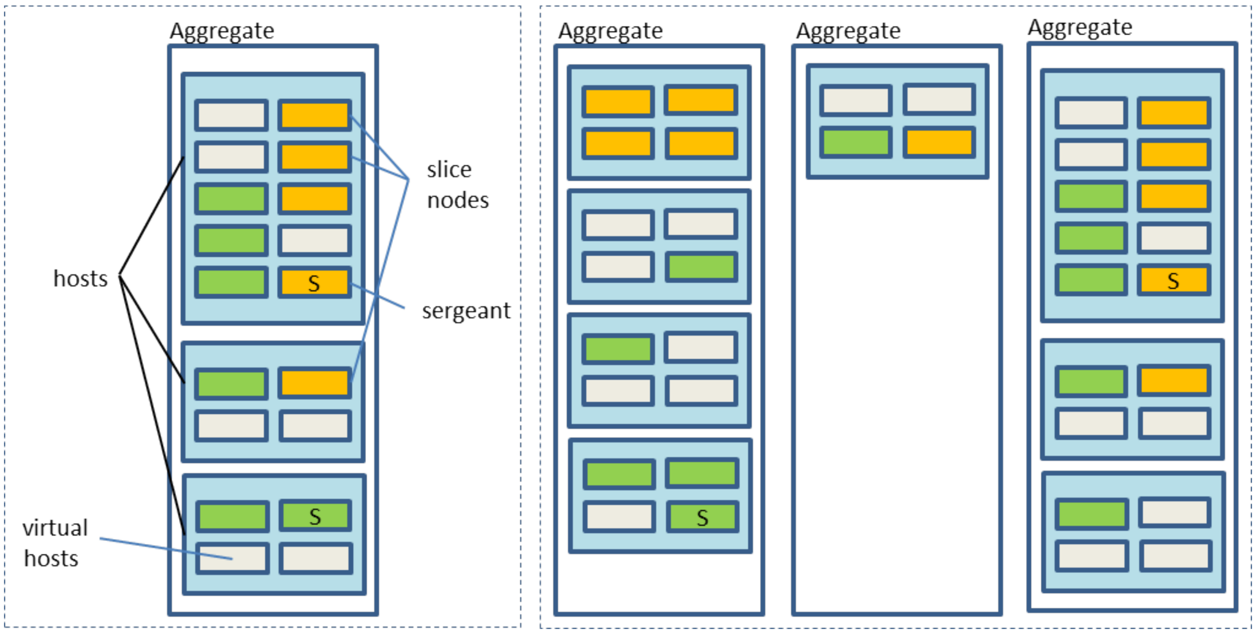
For Slice Enclaves we will need to run a Sentinel Process on each node assigned to the Slice. Because Researchers have root access to their nodes, the Sentinel Process may be removed or tampered with. If the process were to be "root-kitted", it would appear to outside monitoring to be operating correctly, but in fact not be so, and be selectively not providing the correct security information. External health monitoring by the Sergeant Process, or by neighbor Sentinels may help detect this; however

a. Researchers can specify their own arbitrary or custom operating systems to run, perhaps a custom kernel and distribution image; we have no ready method to support a Sentinel Process on an arbitrary set-up. However, a set of executables specific to standard hardware and OS distributions could be created. When the node is created, the appropriate Sentinel version would be loaded. A configuration management table, much like XML DTDs can be associated with each version. The loader would then use this information to determine which Sentinel version to use. For cases where the node is truly unique and incompatible, we will have to rely on monitoring external to the node.

b. Because there is a strong push towards virtualization whenever possible we believe we can provide enhanced security by modifying the hypervisor. Because XEN is being promoted for ProtoGENI and it is an open and extensible system, while this will be quite challenging, we believe that it will serve as the best, secure source of node information generally available. Each XEN hypervisor could reserve some portion of the host for the Sentinel and as a landing zone for Sensors. However, how/whether the Sensors might access the private portions of the VMs running on each host is TBD.

c. A still stronger security method, using a separate hardware device on a card could be developed. This would provide near tamper proof monitoring and a separate network path for communications with the Sergeant. While a possible solution, the projected expense and requirement that Resource Providers install these in all hosts providing components, not a viable solution. However, if some providers wished to provide a high-reliability set of components, perhaps that would justify the expense.

d. Given these issues, for initial releases of the product we intend to make Slice Enclave Sentinels and monitoring available as a user option.



6. Implementation and Evaluation

Initial Implementation

The open-source project Java Agent DEvelopment Framework (JADE) was selected as the basis of the prototype system [JADE]. Developed by Telecom Italia, JADE is an Open source project that provides a middleware platform for distributed, agent-based applications (referred to as an enabling technology).

In the JADE framework, agents are implemented as threads and support is provided for common agent activities such as interaction, management of agent sleep and wake, and structuring complex tasks. To manage agent operations JADE uses containers, which can reside on the same system or different systems. Collectively the platforms are then referred to as an agent platform. Each agent platform has a main container, which also contains two specialized agents, the Agent Management System (AMS) agent and the Director Facilitator (DF) agent. The AMS agent can be considered the authority for the platform since it is responsible for the creation and termination of other agents, termination of containers, and the shut down the entire platform. In contrast the DF (Directory Facilitator) agent implements a yellow pages service which advertises the services of agents in the platform so other agents requiring those services can locate them. Beyond these two required agents, additional agents can be created to provide the required platform functionality.

For the current JADE implementation of CID, each host (system to be protected) is a container and a collection of containers (agent platform) is an enclave. At initialization the AMS agent determines the geography of the hosts in the enclave using the Peano-Hilbert space filling curve algorithm described in [Haack09]. This information is then distributed to Sentinel agents that are resident per host. Sensor agents are then created via the AMS agent and begin wandering from container to container (equivalent to moving from host to host). Once at a container the Sensor agent will attempt look for a certain type of evidence and report to the Sentinel agent. For example in the current implementation there are three types of evidence sought: network, process, and disk activity. It is important to note that given the limitations of Java with respect to accessing low-level system information helper scripts resident on the host are required to harvest the required information. Thus the Sensors provide an API to manage the low-level differences that exist among operating systems. Sensor then processes the low level information and provides the findings to the Sentinel. The Sentinel then provides the Sensor geography and pheromone information, which will be used to determine the Sensor's next destination container. This process then repeats as described in [Haack09].

Question: How pheromone is implemented is an open question. Several versions have been tried. Where/who will actually store this information?

Further Implementations

As discussed earlier there are two types of hosts to protect, Slices and Infrastructure. Because we have ready access to Slice nodes, our following implementation will be to protect example Researcher Slice Enclaves. Each node in the Researcher's Slice will have a Sentinel process. We will restrict the Slice to assets from a single Aggregate Manager. A simplified Sergeant process will be run on a separate node associated with the Slice for this purpose. These Sentinels are based on the Java prototype described above. Evaluation will be based on how well this configuration responds to our test suite of activity. Although we have restricted this to a Single Aggregate Manager, the evaluation will be equivalent to that on a Slice that spans many providers' sites.

The difference between this configuration and a full ProtoGENI (or GENI) Slice is that the Sergeant is co-located with the Slice nodes and running on a node associated with the Slice, i.e. not external to the Slice. This allows us to delay implementation of Infrastructure code for stand-alone Sergeant processes, and determining where to place a Sergeant for a Slice distributed across different Aggregate Manager sites. However, this may not be an issue if we determine that an arbitrary additional node for the Sergeant is sufficient.

Initially communication will be over the Slices private network. We will want to determine how much network traffic to expect, and to determine if it would adversely affect operations would it and a projected number of other Experiments be running concurrently. If this is suitable, we will change so that communications use the control network. This will provide isolation and not affect the Researcher's experiment. Also, it will allow us to connect to a stand-alone Sergeant on a host connected to the site's control network (or a direct connection should that be required).

Successive steps will be to create a stand-alone Sergeant and a method for the Slice nodes to report to it. Suggested is to start the Sergeant before the Slice nodes are allocated and provide the Sergeant's location as part of the Sentinel initialization process. This may be implemented using VMs on a dedicated host, or integrated into the Aggregate Manager itself. This is similar to how Sergeants must be implemented for protecting Infrastructure hosts.

Early on we would like to make a version available opt-in for GENI researchers. This would help in evaluating its effectiveness and in determining resource utilization. The security monitor would be run optionally as part of their slice configuration. A node for the Sergeant would also be added to the experiment. For this case, the Supervisor, that is, where the Sergeant will report its findings, will initially be to the Researcher and to a web site dedicated to monitoring this research.

While this could be defeated by a malicious user or attacker who has otherwise compromised the slice, we are assuming that because the researcher has chosen to use security monitoring, their focus is on securing their experiment, not in finding ways to defeat a prototype system. Although if done, reports of their success or failure would be desired.

Next we will focus on Infrastructure Enclaves. Here, we will initially either run a modified version of Emulab in Emulab, and/or create a stand-alone Aggregate Manager that has been instrumented with our Sentinel and modified Reference Control Manager.

As an alternative to modifying an operational ProtoGENI enclave, we are considering building several small ProtoGENI/Emulab enclaves and a mock Central Authority. This would allow us to place Sentinels on infrastructure hosts and construct Slices that federate across enclaves. This would also allow us to test inter-Sergeant communications.

To demonstrate detection of proposed threats, our evaluation plan is to

- Create several simple scenarios that span the attack space from our representative threats.
- Analyze the selected threats and identify what the likely host-based effects will be.
- From these, derive sensors for the evidence of likely unusual conditions that will arise. Add a number of other sensors that we think may have some value, sometime.
- Create a suite of attack scripts to for these scenarios.
- Using these determine the effectiveness of the detection system.
    - Was the evidence reported sufficiently different from normal activity to determine that an attack was occurring?
    - Determine how fast an attack is detected.
    - Was their sufficient evidence to determine what was happening?
    - Do a multitude of alerts cause confusion to the Supervisor?
    - What was the resource use before, during and after the period of the attack that we can attribute to the detection system (not the attacks)?
- Try introducing attacks that are variants of those above or we had not specifically considered when defining sensors.

7. Research Questions and Challenges

- What effect does the HiveMind system have on the control system?

- What effect does the HiveMind system have on the Researcher's experiments?
- How will HiveMind affect consistent behavior of experiments?
- Is there sufficient number of monitoring elements to make Digital Ants model successful? If not how can we modify the system to mitigate this?
- There may be only a few sentinels (or one) from each Provider in a Slice. Could a malicious user select the set of resources to defeat detection?
- Given that the control network is accessible from experiment nodes (in addition to the experiment's created inter-node networking), we must monitor and protect for inappropriate use of the control channel.
- Not all Federations will use the GENI system, but integrate with it over a GENI API. For these non-GENI federations, we will not be providing security monitoring. Is there a Security API that we could use to provide information to GENI Sergeants?
- Attacks that spread within a Slice, e.g. if an experiment went bad or was compromised by malware from the Internet, may cross different enclaves. This will have visibility across multiple Enclaves. Multiple Sergeants may observe the activity. We need to determine how Sergeants will interact/communicate.
- It may be useful to investigate a situation where different Resource Providers provide different levels of both Slice and Infrastructure security. Using this information a Researcher would be able to only run on Slices that provided the needed level of security for their experiments. This is similar to the Researcher selectable firewall level based on the dangerousness of their experiment. Similarly, resource providers may wish to only federate with others who provide sufficiently high levels of security. For highly secure experiments, Slices may only wish to operate on hosts equipped with separate monitoring hardware as mentioned earlier. For special situations there may be sufficient support for this.
- Experiments may require malware, attacks, etc. In these cases we may not want security monitoring intra-slice, only inter-slice, slice-infrastructure, and slice-Internet. Do we need to add a user option to select the security level similar to how ProtoGENI sets experiment firewall levels?
- Support for emergency shutdown is a major competent of GENI security. Can we automatically take action that may disrupt an experiment with higher false positive rate, but require human intervention to take action that would have greater effects?
- What information do we want to communicate w/ Operations and Management (O&M), what and when?
- Can we detect violation of resource allocation and access control policies in component/aggregate managers?
- If a Sentinel or Sergeant process were to be compromised, how could this be detected? Assuming the process were modified to never report that attackers activity, could evidence externally observed by neighbor processes be used to identify this?

8. References

[BEF+10] Matt Bishop, Sophie Engle, Deborah A. Frincke, Carrie Gates, Frank L. Greitzer, and Sean Peisert, "A Risk Management Approach to the 'Insider Threat,'" In *Insider Threats in Cybersecurity*. Springer, 2010.

[Bis09] Matt Bishop et al. "GENI and Security Workshop Final Report," February 2009.

[GSec2] GENI Project Office, "GENI Spiral 2 Security Plan," March 15, 2010.

[GSol2] GENI Project Office, “Solicitation 2 for GENI Development & Prototyping Proposals,” December 15, 2008.

[GSpi1] GENI Project Office, “GENI Spiral 1 Overview,” September 29, 2008.

[GSys] GENI Project Office, “GENI System Overview,” September 29, 2008.

[Haack09] "Mixed-Initiative Cyber Security: Putting Humans in the Right Loop." Jereme N. Haack, Glenn A. Fink, Wendy M. Maiden, David McKinnon, and Errin W. Fulp. *In Proceedings of the Eighth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS~R09)*, 2009.

[JADE] Telecom Italia, 2000. http://jade.tilab.com/

[PNBA06] H. V. D. Parunak, P. Nielsen, S. Brueckner, and R. Alonso, “Hybrid Multi-Agent Systems: Integrating Swarming and BDI Agents.” In: Brueckner S.A.., Hassas, S., Jelasity, M, Yamins, D. (eds.), *Proceedings of the 4th International Workshop on Engineering Self-Organizing Systems (ESOA)*, Lecture Notes in Computer Science vol. 4335, pp 1–14. Springer, Heidelberg, 2007.

[Self88] O.G. Selfridge. Pandemonium: a Paradigm for Learning. In: Anderson, J.A.D., Rosenfeld, E.: *Neurocomputing: Foundations of Research*, pp 115–122. MIT Press, Cambridge, MA, 1988.

[Smie96] F. Smieja. The Pandemonium System of Reflective Agents. *IEEE Transactions on Neural Networks*. 7, 1996.