The Behavioral Adversary Modeling System - Console Based Generator

M. Boddy and **H. Shackleton**
Adventium Labs
100 Mill Place 111 Third Avenue South
Minneapolis, MN 55401 USA
{mark.boddy,hazel.shackleton}@adventiumlabs.org

Description

This document describes an automated system intended to aid computer network administrators in analyzing the vulnerabilities of their systems against various kinds of potential attackers. In our Behavioral Adversary Modeling System (BAMS), the security analyst selects the attributes and objectives of a potential adversary and then invokes a Course Of Action (COA) generator to hypothesize attacks that such an adversary is likely to choose in attempting to subvert the system. The analyst can then use this information to evaluate the susceptibility of his system to attacks by a particular type of adversary, and to select the most reasonable defensive measures.

Our work was motivated by a particularly challenging problem: analyzing the threat posed by malicious insiders, adversaries who are legitimate users of the system, with the ability to mount physical, cyber, and social engineering exploits to achieve their goals. Our BAMS prototype generates hypothetical insider attacks against a simple but realistic model of a web-based Document Management System. BAMS can be used to generate a sequence of attacks against a given network, exploiting different vulnerabilities as old ones are blocked through configuration changes. The plans generated vary from twenty to fifty atomic steps, and are produced rapidly enough (typically less than a second) to make the interactive use of the tool feasible.

The BAMS (Behavioral Adversary Modeling System) package is a set of tools used to model computer and social networks in a enterprise business environment to find potential security vulnerabilities. Plans generated using the BAMS domain description have been evaluated by cybersecurity experts and have been found to contain the appropriate structure for discovering security vulnerabilities. For more information, see the ICAPS-05 paper on BAMS found at http://www.adventiumlabs.org/ under Publications and is titled "Course of Action Generation for Cyber Security Using Classical Planning."

The BAMS simulation

The process we support is the construction and manipulation of adversary models, specifically in reasoning from an adversarys goals, capabilities, and knowledge to what they may attempt to do in order to achieve those goals. A simple schematic model of this process is presented in Figure 1. The value here is in the predictive behavioral model of the enemy, not in the collection of enemy characteristics and preferences. Unlike most previous work in the area, our system makes predictions of specific adversary behaviors, so that network defenders can concentrate on defending against specific attacks identified as especially likely or especially costly, rather than simply configuring their defenses to protect against the latest published attacks.

Given this Behavioral Adversary Model, analysts can make intelligent trades among different security architectures and counter-measures. They can also use these models to assess the risk associated with trusting the system to protect mission critical data or services. Red Teams can use behavioral models to develop more realistic attack scenarios than the ones they use today, ideally capturing the preferences and tendencies of particular classes of adversaries in question, rather than the skill set or preferences of the red team. To be of any real use, these models must address the full range of physical and social engineering exploits as well as the cyber exploits that an adversary might employ.

The BAMS simulation provided for use by the International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS) is meant to help competitors model the adversarial agent. That is, the simulation provides a means of executing COAs for exploiting security risks in a given environment, and assuming different capabilities on the part of attackers. Competitors do not need to automatically generate changes to the network to preclude attacks. However, plan quality functions for the domain emphasize the search for plans that will hilight risks inherent in the environment. For example, plans that are inherently risky for the attacker will have low quality, plans that make use of capabilities the attacker is unfamiliar with will have low quality, and so on.

The rest of the document describes the classes of knowledge available, and the actions that can be executed in the domain.

The BAMS Domain

The sections below describe the BAMS domain in more detail. The descriptions provided have four parts:

- Definitions are informal text descriptions that assist in understanding the formalizations that are provided.
- Types are basic types as in PDDL.
- Relationships are propositions as in PDDL.

- Modes in the BAMS domain are used to describe processes that have a regular, state-machine like structure to them.

- Actions are similar to PDDL actions, with additional information denoting mode transitions.

The descriptions are further broken down into sections as follows:

- Basics describes types and relations common to all BAMS problems.

- People describes the people who are the principal actors in all BAMS problems and their knowledge and abilities.

- Machines describes the machines used in BAMS problems and their capabilities.

- Networks describes two different network contents and topologies used in BAMS problems.

- Programs describes the type and capabilities of programs used in BAMS problems.

- Offices describes two different office layouts used in BAMS problems.

- Types, modes, relations and actions specific to different problems are then described in subsequent sections. These are: Email, Encryption, Keydrop, Malware, DMS,Physical and Process.

Basics

The following types can be used in every domain:

Type	Description
c_host	A physical computer system.
c_human	Yer basic bloke.
c_info	Information ie a specific message.
c_instruction	An order from somebody.
c_skill_level	A skill level.

The following relationships can be used in every domain:

knows Description: Human knows some information Info. Parameters: Human (c_human), Info (c_info)
insider Description: Human is a treacherous insider. Parameters: Human (c_human)
duped Description: Human has been fooled into doing something. Parameters: Human (c_human)
paranoid Description: Human is paranoid about security. Parameters: Human (c_human)
inst_by Description: Instruction is given by Human. Parameters: Instruction (c_instruction), Human (c_human)
inst_update_file Description: Instruction tells to update File on Host. Parameters: Instruction (c_instruction), Host (c_host), File (c_file)
inst_to Description: Instruction is for Human. Parameters: Instruction (c_instruction), Human (c_human)
trusts_instruction Description: Human trusts Instruction. Parameters: Instruction (c_instruction), Human (c_human)
trusts_instructions_by Description: Human trusts instructions from Human. Parameters: Human (c_human), Human (c_human)
knows_instruction Description: Human knows Instruction. Parameters: Human (c_human), Instruction (c_instruction)
tech_skill Description: Human has Skill_level tech skills. Parameters: Human (c_human), Skill_level (c_skill_level)
social_skill Description: Human has Skill_level social skills. Parameters: Human (c_human), Skill_level (c_skill_level)

People

Stuff about admin ADAM

Adam is in his office. Adam has a uid. There exists an nes admin password for the everest host. Adam knows his general password, dms password and the nes admin password. Adam is a dms admin.

Adam's password can be used to log in his uid on the bigfoot host. Adam's uid is in the admin gid group. Adam's uid is in the everyone gid group. Adam's dms password can be used as a dms login for his uid on the everest host.

Stuff about bad boy Bob.

Bob is an insider. Bob's tech skill lever is high. Bob's social engineering skill level is low. Bob knows his password and dms password. Bob has a electronic pgp key and a pgp password which he knows, both of which can be used for authorization. Bob has a uid. Bob is in his office.

Bob's password can be used to log in his uid on the everest host. Bob's uid is in the east gid group. Bob's uid is in the everyone gid group.

Stuff about good guy Greg.

Greg has and knows a general password and a dms password. Greg has a pgp electronic key. Greg only trusts email

that is digitally signed. Greg has a uid. Greg is in his office. Greg trusts instructions by Adam. Greg trusts hosts sherpa and everest. Greg has a digital certificate for his uid. Greg's dms password can be used as a dms login for his uid on the everest host. Greg is in the south gid and the everyone gid. Greg's general password logs his uid into the sherpa host.

Programs

The following programs exist in the domain.

UNIX programs

C Shell, a login shell - can execute new programs. At&t unix mail utility - can compose and read email. Trojan master - controls the ie_virus - can receive documents from trojan transmitters. Mozilla browser - can be a dms client.

WINDOWS programs

Internet explorer - worlds most popular browser - can be a dms client, can be infected by the IE virus. Windows explorer shell - can execute new programs. MS Outlook - mail client on windows - can compose and read email. IE virus - has yet to be written - can transmit documents from programs it infects. Framebuffer virus - forwards framebuffers - can transmit documents from programs it infects. MS Exchange - email transfer application. IIS - the nominal web server for the dms at the moment - dms server. NES - can be a dms server. NES admin - a NES admin server. Ethereal is a net sniffer - can sniff packets.

Machines

Bigfoot is a windows box that adam the admin uses.

Bigfoot has file system type ntfs. Bigfoot is in Adam's office. Adam trusts bigfoot. Adam's password works for his uid on bigfoot. Nobody is currently at or logged into bigfoot.

MS Outlook is accessible from bigfoot and gives Adam execute permission. Windows explorer is accessible from bigfoot, contains a login shell, and gives Adam execute permission. Internet explorer is accessible from bigfoot and gives Adam read, write and execute permission.

Host everest runs the DMS.

There are two file systems, ntfs and unix. There exists some secret info which is contained in a secret document on everest. The dms uid has read access to the secret document. The south gid has dms read access to the secret document.

Everest is in Adam's office. Everest has file system type ntfs. Nobody is currently at everest. Everest has a root uid and a dms uid. Everest has an everyone gid, and east gid, a south gid and an admin gid.

Process 0 on everest server is NES web server running as the dms uid. Process 1 on everest server is NES admin server running as the root uid. Process 2 on everest server is Exchange email running as the root uid.

Host sherpa which is nominally Gregs workstation.

Sherpa has file system type ntfs. Sherpa is in Greg's office. Nobody is currently at or logged into Sherpa.

MS Outlook is accessible from sherpa and gives greg execute permission. A file on sherpa contains Greg's password. The root uid has read access on this file. Windows explorer is accessible from sherpa, contains a login shell, and gives everybody execute permission. Internet explorer is accessible from sherpa and gives Greg read, write and execute permission. A trojan plugin is accessible from sherpa, although Greg will not run it, and gives everyone read and execute permission.

Yeti as a Windows(TM) Machine used by Bob.

Yeti has file system type ntfs. Yeti is in Bob's office. Bob trusts yeti. Bob's password works for his uid on yeti. Nobody is currently at or logged into yeti.

MS Outlook is accessible from yeti and gives bob read and execute permission. It is owned by the root uid. Windows explorer is accessible from yeti, contains a login shell, and gives bob read and execute permission. It is owned by the root uid. Internet explorer is accessible from yeti and gives Bob read, write and execute permission. It is owned by the root uid. Ethereal is accessible from yeti and gives Bob execute permission. It is owned by Bob's uid.

Yeti is insider Bob's node.

Yeti has file system type unix. Yeti is in Bob's office. Bob trusts yeti. Bob's password works for his uid on yeti. Nobody is currently at or logged into yeti.

A login shell is accessible from yeti. It is owned by the root uid, gives everyone execute permission and is assigned to the group 'everyone'. A mail client is accessible from yeti. It is owned by the root uid and gives bob execute permission. A virus is accessible from yeti. It gives everyone read, write and execute permission. A trojan master is accessible from yeti. It is owned by Bob's uid and gives him execute permission. Mozilla is accessible from yeti. It is owned by the root uid and gives Bob's uid execute permission.

Offices

Offices in Office 1 are arranged as shown. Doors d0 and d1 both have a lock, which can be locked/unlocked from both sides. All locks start out locked.

Offices in Office 2 are arranged as shown. All three doors start out closed. All three doors have a lock, which can be locked/unlocked from both sides. All locks start out locked.

Each lock has one key. The key for lock 1 starts in the closet. The key for lock 2 starts in Adam's office. The key for lock 3 starts in Bob's office.

Networks

Network 1 is a very simple network topolgy with just a hub. The firewall is irrelevant since everest is inside.

Hosts yeti, sherpa and bigfoot are all connected to hub1. Host yeti can reach host everest directly, and vice versa. Host yeti can reach host sherpa directly, and vice versa. Host yeti can reach host bigfoot directly, and vice versa. Host sherpa can reach host everest directly. Host sherpa can reach host bigfoot directly. Host bigfoot can reach host sherpa directly. Host bigfoot can reach host everest directly.

A firewall is connected to hub1.

Network 2 is a more complex network topology with switch and firewall.

Hosts yeti, sherpa and bigfoot are all connected to hub1. Firewall1 is connected to hub1. Firewall1 is connected to switch1. The firewall forwards smtp packets and https packets. Host yeti can reach host everest through the firewall1. Host sherpa can reach host everest through the firewall1. Host bigfoot can reach host everest through the firewall1. Host yeti can reach host sherpa directly, and vice versa. Host

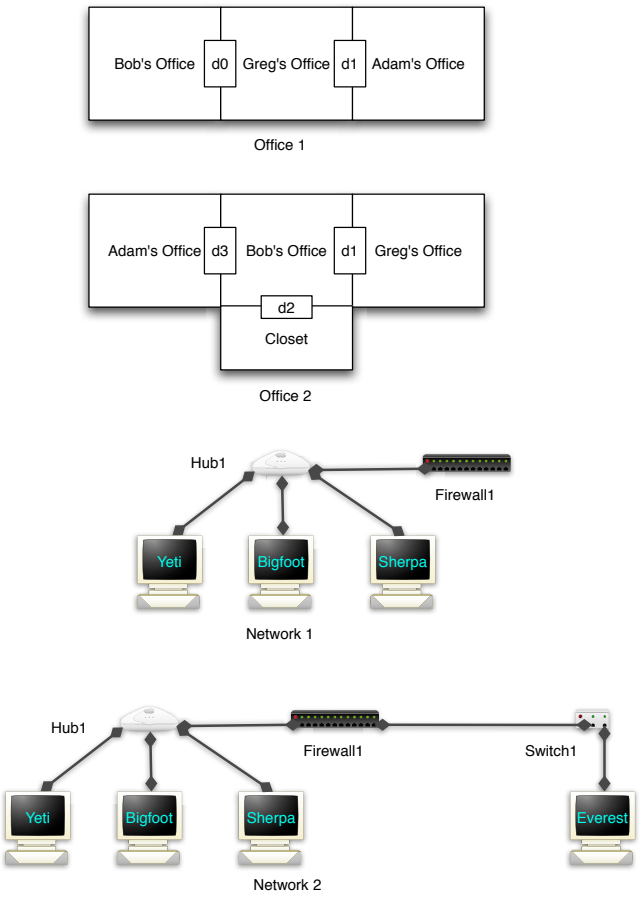

Figure 1: Office and network topologies for BAMS domain.

yeti can reach host bigfoot directly, and vice versa. Host sherpa can reach host bigfoot directly, and vice versa. Host everest is connected to switch1.

Email

The following types are used in the email domain:

Type	Description
c_email	An email message, possibly with attachments.

The following relationships are used in the email domain:

my_email Description: Email is from Human. Parameters: Human (c_human), Email (c_email)
apparent_sender Description: Email seems to be sent by Human. Parameters: Email (c_email), Human (c_human)
attached Description: File is an attachment to Email. Parameters: Email (c_email), File (c_file)
can_compose_email Description: program Prog can compose email. Parameters: Prog (c_program)
can_read_email Description: program Prog is an email reader. Parameters: Prog (c_program)
can_transfer_email Description: program Prog is a mail transfer app. Parameters: Prog (c_program)
email_contents Description: Email contains Info. Parameters: Email (c_email), Info (c_info)
email_contents_instruction Description: Email contains Instruction. Parameters: Email (c_email), Instruction (c_instruction)
in_inbox Description: Email is in Uid's inbox. Parameters: Email (c_email), Uid (c_uid)
recipient Description: uid Ruid is in an intended recipient of Email. Parameters: Email (c_email), Ruid (c_uid)
transmitted Description: Email has been sent. Parameters: Email (c_email)
trusts_recipient Description: Truster trusts recipient of information Trusted. Parameters: Truster (c_human), Trusted (c_human)
trusts_enclosures Description: human Truster will open human Trusted's email attachments. Parameters: Truster (c_human), Trusted (c_human)
viewing_email Description: Process on Host is viewing Email. Parameters: Host (c_host), Process (c_process), Email (c_email)
will_open_attachment Description: Email will open on Host. Parameters: Email (c_email), Host (c_host)
writing_email Description: Human, host or UID is writing email Email. Parameters: H (c_human, c_host or c_uid), Email (c_email)

The following modes are used in the email domain:

Type	Description
m_email_started	Email message is started.
m_email_senderset	Email message has sender set, adding recipients.
m_email_writing	Email message body in progress.
m_email_attaching	Email message attachments being added.
m_attachment_opening	Email attachment opening in progress.

Sending an email is comprised of the following steps:

- Start email.
- Set the legitimate sender or spoof a sender.
- Add recipients.
- Write email containing info, instructions or both.
- Sign email (optional).
- Add attachments (optional).
- Send email.

The details for each step are given below.

- Start email.
Action name: START_EMAIL
Modes: m_free, m_email_started
Parameters: Human (c_human)
Host (c_host)
Uid (c_uid)
Proc (c_process)
Prog (c_program)
Email (c_email)
Preconditions: The email has not been sent.
The program is running on the host as the process by the uid.
The human is at the host.
The email is from the human.
The program can compose email.
Effects: The human is writing the email on the host as the uid.

- Set legitimate sender.
Action name: SET_SENDER
Modes: m_email_started, m_email_senderset
Parameters: Drafter (c_human)
Email (c_email)
Effects: The email seems to be sent by the human.

- Spoof sender.
Action name: CHANGE_SENDER
Modes: m_email_started, m_email_senderset
Parameters: Drafter (c_human; actual sender)
Email (c_email)
Human (c_human; spoofed sender)
Preconditions: The actual sender is writing the email.
The actual sender is an insider.
Effects: The email seems to be sent by the spoofed sender.

- Add recipient.
Action name: ADD_RECIP
Modes: m_email_senderset, m_email_senderset
Parameters: Sender (c_human)
Email (c_email)
Ruid (c_uid)
Recip (c_human)
Preconditions: The human is writing the email.
The sender trusts the recipient of the email or the sender is an insider.
The uid belongs to the recipient.
Effects: The uid is in an intended recipient of the email.

- Write email containing info.
Action name: WRITE_EMAIL
Modes: m_email_writing, m_email_writing
Parameters: Person (c_human)
Email (c_email)
Info (c_info)
Preconditions: The human is writing the email.
The human knows the info.
Effects: The email contains the info.

- Write email containing instructions.
Action name: WRITE_EMAIL_INSTRUCTION
Modes: m_email_writing, m_email_writing
Parameters: Person (c_human)
Email (c_email)
In (c_instruction)
Preconditions: The human is writing the email.
The human knows the instructions.
Effects: The email contains the instructions.

- Sign email.
Action name: SIGN_EMAIL
Modes: m_email_writing, m_email_attaching
Parameters: Human (c_human; sender)
Signer (c_human)
Email (c_email)
Ekey (c_ekey)
Info (c_info)
Preconditions: The sender is writing the email.
The sender allows the use of the key for signing the email.
Effects: The email is digitally signed with the key.

- Add attachment.
Action name: ATTACH_TO_EMAIL
Modes: m_email_attaching, m_email_attaching
Parameters: Host (c_host)
Email (c_email)
Suid (c_uid)
Gid (c_gid)
File (c_file)
Preconditions: The email is being written at the host.
The email is being written by the uid.
The uid or gid has read permission on the file.
Effects: The file is attached to the email.
The detection risk has increased by 1.

- Send email.
Action name: SEND_EMAIL
Modes: m_email_attaching, m_free
Parameters: Host (c_host)
Human (c_human)
Nd (c_nd)

Prog (c_program)
Mta_proc (c_process)
Email (c_email)
Mta_host (c_host)
Preconditions: The email is being written at the host.
The human is writing the email.
The program is a mail transfer application.
The mta host is running the program as the mta process.
The host is connected to the switch/hub.
The mta host is reachable by the host.
Any firewalls in the path forward smtp packets.
Effects: The email is sent.
The email shows up in the inbox of all recipients.
All sniffers on the network attempt to sniff the email.

Other email actions are below.

- Ruid opens email enclosure of Email containing Attachment.
Action name: OPEN_EMAIL_ENCLOSURE
Modes: m_attachment_opening, m_free
Parameters: Ruid (c_uid; recipient)
Gid (c_gid; sender)
Host (c_host)
Email (c_email)
Target_file (c_file)
Attachment (c_file)
Preconditions: The attached file is attached to the email.
Host will open attacment on Email.
Effects: If the attached file contains an unknown virus, or
the host does not scan for viruses, the virus
will infect every file on the host that the reader of
the email as write access to.

- View email.
Action name: VIEW_EMAIL
Parameters: Ruid (c_uid; recipient uid)
Human (c_human; recipient)
Author (c_human; sender)
Host (c_host)
Email (c_email)
Mua_prog (c_program)
Mua_proc (c_process)
Preconditions: The email is in the inbox of the recipient uid.
The host is running an email program as the process as the uid.
The human is at the host.
The author is the apparent sender of the email.
The recipient does not require emails to be signed.
Effects: The process on the host is viewing the email.
The recipient learns the info/instructions in the email.

- View signed email.
Action name: VIEW_EMAIL_SIGNED
Parameters: Ruid (c_uid; recipient uid)
Human (c_human; recipient)
Author (c_human; sender)
Key (c_ekey)
Host (c_host)
Email (c_email)
Mua_prog (c_program)
Mua_proc (c_process)
Preconditions: The email is in the inbox of the recipient uid.
The host is running an email program as the process as the uid.
The human is at the host.
The author is the apparent sender of the email.
The email is signed and the author has the key.
Effects: The process on the host is viewing the email.
The recipient learns the info/instructions in the email.

- Person checks the enclosure of Email aparently from Author.
Action name: CHECK_EMAIL_ENCLOSURE
Modes: m_free, m_attachment_opening
Parameters: Person (c_human)
Author (c_human)
Host (c_host)
Mua_proc (c_process)
Email (c_email)
Preconditions: Host is viewing Email on Host.
Author is apparent sender of Email.
Person trusts enclosures from Author.
Effects: Host opens attachement from Email

- Email has been addressed.
Action name: DONE_RECIP
Modes: m_email_senderset, m_email_writing
Parameters: Email (c_email)
Ruid (c_uid)
Preconditions: Ruid is recipient of Email.
Effects: None.

- The body of Email from Human is completed.
Action name: DONE_WRITE
Modes: m_email_writing, m_email_attaching
Parameters: Human (c_human)
Email (c_email)
Preconditions: Human is writing Email.
Effects: None.

Encryption

The following definitions are used in the encrypt domain:

- A human or a piece of info can be used for authentication.

The following types are used in the encrypt domain:

Type	Description
c_ekey	An electronic encryption key.
c_cert	A secure digital certificate.

The following relationships are used in the encrypt domain:

requires_signatures Description: Human only trusts email that is digitally signed. Parameters: Human (c_human)
encrypted Description: Info is encrypted and decryptable by key Ekey. Parameters: Info (c_info), Ekey (c_ekey)
file_encrypted Description: File is encrypted and decryptable by key Ekey. Parameters: File (c_file), Ekey (c_ekey)
signed Description: Email is digitally signed with key Ekey. Parameters: Email (c_email), Ekey (c_ekey)
authorization Description: Auth_method allows use of key Ekey. Parameters: Ekey (c_ekey), Auth_method (c_auth_method)
has_ekey Description: Human has electronic key Ekey. Parameters: Human (c_human), Ekey (c_ekey)
has_cert Description: Cert is a digital certificate for Uid. Parameters: Uid (c_uid), Cert (c_cert)
cert_installed Description: Cert is installed on Host. Parameters: Host (c_host), Cert (c_cert)

The following actions occur in the encrypt domain.

- Encrypt file.
Action name: encrypt_file
Parameters: Human (c_human)
File (c_file)
Host (c_host)
Uid (c_uid)
Gid (c_gid)
Key (c_ekey)
Preconditions: The human is at the host.
The uid is authenticated on the host.
The uid is a member in the gid.
The file as accessible on the host.
The uid or gid has write access to the file.
Either the file is not encrypted or if it is encrypted,
the human is able to decrypt it.
Effects: The file and all of its info is encrypted with the key.

Keydrop

The following definitions are used in the keydrop domain:

- A keylogger can be installed on a host. When a keylogger is installed on a host, it can record information that is entered at the keyboard.

The following types are used in the keydrop domain:

Type	Description
c_keylogger	Physical keylogger.

The following relationships are used in the keydrop domain:

has_keylogger Description: Human is in possession of Keylogger. Parameters: Human (c_human), Keylogger (c_keylogger)
is_keylogged Description: Host has Keylogger attached. Parameters: Host (c_host), Keylogger (c_keylogger)
keylogger_contents Description: Keylogger contains Info. Parameters: Keylogger (c_keylogger), Info (c_info)

The following actions occur in the keydrop domain.

- Install Keylogger.
Action name: install_keylogger
Parameters: Human (c_human)
Host (c_host)
Room (c_room)
Keylogger (c_keylogger)
Preconditions: The human is in the room.
The host is in the room.
The human has the keylogger.
Effects: The human no longer has the keylogger.
The host is keylogged.
The risk of detection is increased one level
for each person who is in the room.

- Retrieve Keylogger.
Action name: retrieve_keylogger
Parameters: Human (c_human)
Host (c_host)
Room (c_room)
Keylogger (c_keylogger)
Preconditions: The human is in the room.
The host is in the room.
The host is keylogged.
Effects: The host is no longer keylogged.
The human has the keylogger.
The risk of detection is increased one level
for each person who is in the room.

- Read Keylogger.
Action name: read_keylogger
Parameters: Human (c_human)
Keylogger (c_keylogger)
Info (c_info)
Preconditions: The keylogger contains the info.
The human has the keylogger.
Effects: The human knows the info.

Malware

The following relationships are used in the malware domain:

file_infector Description: Malware can infect files with itself. Parameters: Malware (c_program)
can_infect Description: file Prog can be infected by Malware. Parameters: Prog (c_program), Malware (c_program)
can_transmit_documents Description: trojan Program can transmit documents from programs it infects. Parameters: Program (c_program)
can_receive_documents Description: trojan Program can receive documents from trojan transmitters. Parameters: Program (c_program)
code_injector Description: Prog is a code injector. Parameters: Prog (c_program)
can_inject Description: Vic can have Code injected by Att. Parameters: Att (c_program), Code (c_program), Vic (c_program)
known_virus Description: Prog has a known viral signature. Parameters: Prog (c_program)
virus_scanner Description: Prog is a virus scanner. Parameters: Prog (c_program)
scanned_host Description: Host is under protection of a virus scanner. Parameters: Host (c_host)

The following actions occur in the malware domain.

- Download file injector.
Action name: download_file_injector
Parameters: Host (c_host)
File (c_file)
Prog (c_program)
Human (c_human)
Uid (c_uid)
Gid (c_gid)
Preconditions: The human is an insider.
The human is at the host.
The uid is authenticated on the host.
The file is empty.
The uid or gid has permission to write to the file.
The tech skill level of the human is not low.
Effects: The file now contains the injector program.

- Write new file injector.
Action name: write_new_file_injector
Parameters: Host (c_host)
File (c_file)
Prog (c_program)
Human (c_human)
Uid (c_uid)
Gid (c_gid)
Preconditions: The human in an insider.
The human is at the host.
The uid is authenticated on the host.
The program is not yet written.
The file is empty.
The uid or gid has permission to write to the file.
The tech skill level of the human is high.
Effects: The file now contains the injector program.

- Inject code.
Action name: inject_code
Parameters: Chost (c_host; client host)
Cproc (c_process; client process)
Cprog (c_program; client program)
Shost (c_host; server host)
Sproc (c_process; server process)
Sprog (c_program; server program)
Human (c_human)
Code (c_program)
Preconditions: The human is at the client host.
The client host is running the code injector program as the client process.
The server host is running the server program as the server process.
The client program can inject the code into the server program.
The client host can reach the server host directly.
Effects: The server host is now running the code as the server process.

- Relay viewed document.
Action name: relay_viewed_doc
Parameters: Doc (c_file)
Human (c_human)
Src_host (c_host)
S_proc (c_process)
Src_proc (c_process)
Malware (c_program)
Dst_host (c_host)
Dst_proc (c_process)
Controller (c_program)
Preconditions: The human is at the destination host.
The source host is viewing the file with source process 1.
The source host is running the malware program with source process 2.
The malware program can transmit documents.
The destination host is running the controller program as the destination process.
The controller program can receive documents.
Effects: The destination host is viewing the file.

DMS

The following definitions are used in the dms domain:

- A file can be read over the dms if the user has read access to the document, or if the group has read access to the document and the user is a member of the group.

The following types are used in the dms domain:

Type	Description
DMS_session	A dms sessions.

The following relationships are used in the dms domain:

my_session Description: Human can start Dms_session. Parameters: Human (c_human), Dms_session (c_dms_session)
dms_admin Description: Who is a DMS admin. Parameters: Who (c_human)
connected Description: Sess is an active session. Parameters: Sess (c_dms_session)
connecting Description: Sess is an active session. Parameters: Sess (c_dms_session)
can_access_dms Description: Program can be a dms client. Parameters: Program (c_program)
can_serve_dms_docs Description: Program can be a dms server. Parameters: Program (c_program)
nes_admin_prog Description: Program can be a NES admin server. Parameters: Program (c_program)
nes_admin_connected Description: Chost is in an admin session on Shost. Parameters: Chost (c_host), Shost (c_host)
dms_established Description: Dms_session is an authenticated session. Parameters: Dms_session (c_dms_session)
dms_session_real_user Description: Human initiated DMS session. Parameters: Dms_session (c_dms_session), Human (c_human)
dms_session_client Description: DMS session has Host as client host. Parameters: Dms_session (c_dms_session), Host (c_host)
dms_session_cproc Description: DMS session is connected to client process Process. Parameters: Dms_session (c_dms_session), Process (c_process)
dms_session_server Description: DMS session has Host as server host. Parameters: Dms_session (c_dms_session), Host (c_host)
dms_session_sproc Description: DMS session is connected to server process Process. Parameters: Dms_session (c_dms_session), Process (c_process)
dms_session_pwd Description: DMS session authenticated with password Info. Parameters: Dms_session (c_dms_session), Info (c_info)
dms_authenticated Description: Uid has authenticated to DMS session. Parameters: Dms_session (c_dms_session), Uid (c_uid)
dms_requesting Description: Document File requested from DMS session. Parameters: Dms_session (c_dms_session), File (c_file)
dms_request_in_progress Description: DMS session is making a document request. Parameters: Dms_session (c_dms_session)
dms_password Description: Password is a DMS password for Uid on host Server. Parameters: Uid (c_uid), Server (c_host), Password (c_info)
nes_admin_pwd Description: Password is a password to access the NES admin server Host. Parameters: Host (c_host), Password (c_info)
dmsacl_read Description: uid or gid is allowed to read file Doc. Parameters: Doc (c_file), id (c_uid or c_gid)
dmsacl_write Description: uid or gid is allowed to write file Doc. Parameters: Doc (c_file), id (c_uid or c_gid)
dmsacl_exec Description: uid or gid is allowed to exec file Doc. Parameters: Doc (c_file), id (uid or gid)
dmsacl_list Description: uid or gid is allowed to list file Doc. Parameters: Doc (c_file), id (uid or gid)
dmsacl_info Description: uid or gid is allowed to info file Doc. Parameters: Doc (c_file), id (uid or gid)
viewing_doc Description: Host is viewing Doc with Process. Parameters: Host (c_host), Process (c_process), Doc (c_file)

The following modes are used in the dms domain:

Type	Description
c_dms_session	A dms session.
m_dms_sess	Session spawned.
m_dms_client	Wait for client selection.
m_dms_access	Waiting for route to server.
m_dms_connect	Waiting for network connection to server.
m_dms_know	Waiting for user to know password.
m_dms_auth	Waiting for client to authenticate to server.
m_dms_request	Waiting for client to request a document.
m_dms_send	Waiting for servre to send document.

The following actions occur in the dms domain:
Authenticating a dms is comprised of the following steps:
- Begin dms session. - Find the client. - Find the server. - Connect. - Authenticate via either a certificate or password.
The details for each step are given below.

- Begin dms session.
Action name: DMS_BEGIN_SESSION
Modes: m_free, m_dms_sess
Parameters: Sess (c_dms_session)
Preconditions: The dms session is not a connecting active session.
Effects: The dms session is a connecting active session.

- Find the client.
Action name: DMS_CLIENT_FIND
Modes: m_dms_sess, m_dms_access
Parameters: Client_host (c_host)

Client_proc (c_process)
Client_prog (c_program)
Human (c_human)
Sess (c_dms_session)
Preconditions: The dms session is not a connected active session.
The dms session is not an authenticated session.
The dms session is connecting.
The human can start the dms session.
The client host has the client process.
The client host is running the client program as the client process.
The client program can be a dms client.
The human is at the client host.
The human trusts the client host.
Effects: The dms session has the client host as its client host.
The human has initiated the dms session.
The dms session is connected to the client process.

- Find the server.
Action name: DMS_ROUTE
Modes: m_dms_connect, m_dms_know
Parameters: Client_host (c_host)
Server_host (c_host)
Sess (c_dms_session)
Preconditions: The dms session is not a connected active session.
The dms session is not an authenticated session.
The server program can be a dms server.
The server host is running the server program as the server process.
Effects: The dms session is connected to the server process.
The dms session has the server host as its server host.

- Connect.
Action name: DMS_CONNECT
Modes: m_dms_access, m_dms_connect
Parameters: Server_proc (c_process)
Server_host (c_host)
Server_prog (c_program)
Ses (c_dms_session)
Preconditions: The client host is the dms session's client host.
The server host is the dms session's server host.
The dms session is not a connected active session.
The dms session is not an authenticated session.
The client host can reach the server host.
Any firewalls in the network must allow https forwarding.
Effects: The dms session is an active connected session.

- Verify certificate.
Action name: DMS_VERIFY_CERTIFICATE
Modes: m_dms_know, m_free
Parameters: Chost (c_host)
Uid (c_uid)
Cert (c_cert)
Sess (c_dms_session)
Preconditions: The dms session has the client host as its client host.
The dms session is not an authenticated session.
The certificate is a digital certificate for the uid.
The certificate is installed on the client host.
The uid has been authenticated on the client host.
Effects: The uid has authenticated to the dms session.
The dms session is an authenticated session.

- Enter password.
Action name: DMS_ENTER_PASSWORD
Modes: m_dms_know, m_dms_auth
Parameters: Human (c_human)
Client_host (c_host)
Pwd (c_info)
Sess (c_dms_session)
Preconditions: The human has initiated the dms session.
The dms session has the client host as its client host.
The dms session is not an authenticated session.
The human knows the password info.
Effects: Any keyloggers attached now know the password info.
The dms session has been authenticated with the password.

- Authenticate to dms.
Action name: AUTH_TO_DMS
Modes: m_dms_auth, m_free,
Parameters: Server_host (c_host)
Pwd (c_info; password info)
User (c_uid)
Ses (c_dms_session)
Preconditions: The dms session is not an authenticated session.
The dms session is an active connected session.
The dms session has been authenticated with the password.
The password info is a dms password for the uid on the server host.
Effects: The uid has authenticated to the dms session.
The dms session is an authenticated session.

Sending a file on a dms is comprised of the following steps:
- Begin request. - Request document. - Send document.
The details for each step are given below.

- Begin request.
Action name: DMS_REQ_BEGIN
Modes: m_free, m_dms_request
Parameters: Sess (c_dms_session)
Human (c_human)
Chost (c_host; client host)
Preconditions: The dms session is an authenticated session.
The dms session has the client host as its client host.
The human is at the host.
The human trusts everyone else in the room.
Effects: The dms session is making a document request.

- Request document.
Action name: DMS_REQUEST
Modes: m_dms_request, m_dms_send
Parameters: Sess (c_dms_session)
Doc (c_file)
Shost (c_host; server host)
Uid (c_uid)

Gid (c_gid)
Preconditions: The dms session is making a
document request.
The dms session has the server host as its server host.
The uid has authenticated to the dms session.
The document is accessible from the server host.
The uid or gid has permission to read the document.
Effects: The file is requested from the dms session.

- Send document.
Action name: DMS_SEND
Modes: m_dms_send, m_free
Parameters: Sess (c_dms_session)
Chost (c_host; client host)
Cproc (c_process; client process)
Doc (c_file)
Preconditions: The dms session is making a
document request.
The file is being requested from the dms session.
The dms session has the client host as its client host.
The dms session is connected to the client process.
Effects: A document request is no longer in process.
The file is no longer being requested from the dms session.
The client host is viewing the file as the client process.

Other dms actions are below.

- Read document.
Action name: READ_DOC
Parameters: Human (c_human)
Host (c_host)
Proc (c_process)
Doc (c_file)
Room (c_room)
Info (c_info)
Preconditions: The human is in the room.
The host is in the room.
The host is not locked.
The host is viewing the file as the process.
The file contains the info.
Effects: The human reads the file and learns the info.

- NES Admin login.
Action name: NES_ADMIN_LOGIN
Parameters: Admin (c_human)
Chost (c_host; client host)
Cproc (c_process; client process)
Cprog (c_program; client program)
Shost (c_host; server host)
Sproc (c_process; server process)
Sprog (c_program; server program)
Nd (c_nd; network hub)
Pwd (c_info; password info)
Preconditions: The human is at the client host.
The client host is running the client process.
The client host is running the client program as the
client process.
The client program can be a dms client.
The server host is running the server process.
The server host is running the server program as the
server process.
The server program can be a NES admin server.
The human knows the password info.
The password info is a password to access the NES admin
server host.
The client host is connected to the network hub/switch.
The client host can reach the server host.
Any filewalls in the path forward NES administration packets.
Effects: The client host is in an admin session on the server host.
Any keyloggers attached to the client host know the
password info.
Any packet sniffers on the hub/switch try to sniff the
password info.

- DMS password change.
Action name: DMS_PWD_CHANGE
Parameters: Admin (c_human)
Chost (c_host)
Shost (c_host)
Uid (c_uid)
Pwd (c_info)
Preconditions: The client host is in an admin session on the
server host.
The human is at the client host.
The human is an insider.
The password info is a DMS password for uid on the
host server.
Effects: The human knows the new admin password.
All other people do not know new admin password.
The detection risk level increases by 10.

- Add DMS group allow.
Action name: DMS_ADD_GROUP_ALLOW
Parameters: Admin (c_human)
Chost (c_host)
Shost (c_host)
Doc (c_file)
Gid (c_gid)
Preconditions: The client host is in an admin session on the
server host.
The human is at the client host.
The human is an insider.
Effects: The gid is allowed to read the file.

Network

The following definitions are used in the network domain:

- A packet sniffer can be installed on a network hub or switch, however it can only gather information off a hub. The sniffer can be used to gather passwords and information in emails. When a hub has a packet sniffer installed, all traffic from all hosts connected to the hub is sniffed.

The following types are used in the network domain:

Type	Description
c_nd	10/100 network hub or switch.
c_firewall	A firewall that can filter packets.
c_packet_type	Type of network traffic.

The following relationships are used in the network domain:

can_sniff_packets Description: program Sp can sniff packets. Parameters: Sp (c_program)
is_sniffed Description: Host is running packet sniffer Sp. Parameters: Host (c_host), Sp (c_program)
sniffer_contents Description: packet sniffer Sp has sniffed Info. Parameters: Sp (c_program), Info (c_info)
nd_connected Description: Host is connected to network device Nd. Parameters: Host (c_host), Nd (c_nd)
fw_connected Description: Firewall is connected to network device Nd. Parameters: Firewall (c_firewall), Nd (c_nd)
is_hub Description: network device Nd is a hub. Parameters: Nd (c_nd)
reachable Description: Host1 can reach Host2 directly. Parameters: Host1 (c_host), Host2 (c_host)
reachable_fw Description: Host1 can reach Host2 going through firewall. Parameters: Host1 (c_host), Host2 (c_host), Fw (c_firewall)
fw_allows_smtp Description: firewall forwards smtp packets. Parameters: Fw (c_firewall)
fw_allows_https Description: firewall forwards https packets. Parameters: Fw (c_firewall)
fw_allows_nesadmin Description: firewall forwards NES administration packets. Parameters: Fw (c_firewall)

The following actions occur in the network domain.

- Start sniffing.
Action name: start_sniffing
Parameters: Host (c_host)
Process (c_process)
Program (c_program)
Preconditions: The program can sniff packets.
The program is running on the host as the process.
Effects: The host is running the packet sniffer.

- Read sniffer.
Action name: read_sniffer
Parameters: Human (c_human)
Host (c_host)
Sp (c_program)
Process (c_process)
Info (c_info)
Preconditions: The packet sniffer has sniffed the info.
The human is at the host.
The host is running the program as the process.
The program can sniff packets.
Effects: The human learns the info.

Physical

The following definitions are used in the physical domain:
- A key, human and host are physical things. - A person will leave a room if they trust the remaining people in the room or they are an insider.

The following types are used in the physical domain:

Type	Description
c_door	Solid door.
c_key	Physical key; bit of metal that goes in a lock.
c_lock	Physical lock.
c_room	Physical room.

The following relationships are used in the physical domain:

can_lock Description: Lock can lock Door from Room. Parameters: Lock (c_lock), Door (c_door), Room (c_room)
can_unlock Description: Key can unlock Lock. Parameters: Key (c_key), Lock (c_lock)
door Description: Room has a Door. Parameters: Room (c_room), Door (c_door)
has_key Description: Human is in possession of Key. Parameters: Human (c_human), Key (c_key)
in_room Description: a physical Thing is in Room. Parameters: Thing (c_phys), Room (c_room)
is_open Description: Door is open. Parameters: Door (c_door)
is_locked Description: Lock is locked. Parameters: Lock (c_lock)
trust Description: Human currently trusts everybody in the room. Parameters: Human (c_human)
trusts_in_room Description: human Truster trusts human Trusted to be in Room. Parameters: Truster (c_human) Trusted (c_human) Room (c_room)
at_host Description: Human is at Host and ready to type. Parameters: Human (c_human), Host (c_host)

The following actions occur in the physical domain.

- Grab key.
Action name: grab_key

Parameters: Human (c_human)
Key (c_key)
Room (c_room)
Preconditions: The human and the key must be in the same room.
Effects: The human has the key and the key is no longer in the room.

- Drop key.
Action name: drop_key
Parameters: Human (c_human)
Key (c_key)
Room (c_room)
Preconditions: The human must have the key and be in the room.
Effects: The key is in the room and no longer in the possession of the human.

- Unlock Door.
Action name: unlock_lock
Parameters: Human (c_human)
Room (c_room)
Door (c_door)
Lock (c_lock)
Key (c_key)
Preconditions: The lock can lock the door in the room.
The lock is locked.
The key can unlock the lock.
The human has the key.
The human is in the room.
Effects: The lock is no longer locked.

- Open Door.
Action name: open_door
Parameters: Human (c_human)
Room (c_room)
Door (c_door)
Preconditions: The human is in the room.
The room has the door.
The door is not open.
The door is unlocked.
Effects: The door is open.

- Go through door
Action name: transit_via_door
Parameters: Human (c_human)
RoomFrom (c_room)
RoomTo (c_room)
Door (c_door)
Preconditions: The human is in room 1.
Room 1 is not room 2.
The door connects room 1 and room 2.
The door is open.
The human is not at a computer system
The human is not paranoid or there is not an unlocked computer system in room 1.
The human is an insider or the human trusts all the other people in room 1.
Effect: The human has moved from room 1 to room 2.

- Close door
Action name: close_door
Parameters: Human (c_human)
Room (c_room)
Door (c_door)
Preconditions: The human is in the room.
The door is in the room.
The door is open.
Effect: The door is closed.

- Sit at host
Action name: sit_at_host
Parameters: Human (c_human)
Host (c_host)
Room (c_room)
Preconditions: The human is in the room.
The host is in the room.
Nobody is at the host.
Effect: The human is at the host.

- Leave host
Action name: leave_host
Parameters: Human (c_human)
Host (c_host)
Preconditions: The human is at the host.
Effect: The human is no longer at the host.

Process

The following definitions are used in the process domain:

- A file can contain either info or a program. - Only a program is executable. - A file is readable if the following holds: Parameters: host, file, uid, gid. - The file is accessible from the host. - The uid is a member in the gid. - The file gives read permission to the uid or the gid. - A file is writeable if the following holds: Parameters: host, file, uid, gid. - The file is accessible from the host. - The uid is a member in the gid. - The file gives write permission to the uid or the gid. - A file is executable if the following holds: Parameters: host, file, uid, gid. - The file is accessible from the host. - The uid is a member in the gid. - The file gives execute permission to the uid or the gid. - A person can log into a host if he/she trusts the host and knows the password info. - A program can execute on a host if there is no virus scanner running. - If there is a virus scanner running on the host, the program can run if it is not a virus.

The following types are used in the process domain:

Type	Description
c_file	A file which contains one or more programs.
c_gid	A Group ID.
c_process	A Process ID.
c_program	A Program; Executable Code.
c_uid	An User ID.
c_fstype	A File System Type.

The following relationships are used in the process domain:

accessible
Description: File is on a mounted drive or otherwise accessible by Host.
Parameters: File (c_file), Host (c_host)

authenticated
Description: Uid has been authenticated on console Host.
Parameters: Uid (c_uid), Host (c_host)

can_exec_programs
Description: Program can execute new programs.
Parameters: Program (c_program)

console_user
Description: Human announced user Uid to the console at Host.
Parameters: Human (c_human), Uid (c_uid), Host (c_host)

empty_file
Description: File is empty.
Parameters: File (c_file)

euid
Description: Process is running on Host as Uid.
Parameters: Host (c_host), Process (c_process), Uid (c_uid)

has_uid
Description: Human has been officially assigned Uid.
Parameters: Human (c_human), Uid (c_uid)

file_read
Description: Id can read File.
Parameters: File (c_file), Id (c_uid or c_gid)

file_write
Description: Id can write to File.
Parameters: File (c_file), Id (cid or gid)

file_exec
Description: Id can execute File.
Parameters: File (c_file), Id (c_uid or c_gid)

file_contents_info
Description: File contains Info.
Parameters: File (c_file), Info (c_info)

file_contents_program
Description: File contains Program.
Parameters: File (c_file), Program (c_program)

file_owner
Description: File is owned by principal Id.
Parameters: File (c_file), Id (c_uid or c_gid)

file_group
Description: File is assigned to unix group Gid.
Parameters: File (c_file), Gid (c_gid)

forking
Description: Uid has forked a new process on Host.
Parameters: Host (c_host), Uid (c_uid)

fstype
Description: Host has file system type Fstype.
Parameters: Host (c_host), Fstype (c_fstype)

has_process
Description: Host has the Process.
Parameters: Host (c_host), Process (c_process)

in_group
Description: Uid is a member of group Gid.
Parameters: Uid (c_uid), Gid (c_gid)

host_locked
Description: Host is locked.
Parameters: Host (c_host)

login_password
Description: Info can be used as a password for Uid on Host.
Parameters: Uid (c_uid), Host (c_host), Info (c_info)

weak_password
Description: Password is weak - eg. dictionary word.
Parameters: Info (c_info)

login_shell
Description: File contains a login shell.
Parameters: File (c_file)

norun
Description: Human will not run File.
Parameters: Human (c_human), File (c_file)

running_prog
Description: Host is running program Executable with process id Process.
Parameters: Host (c_host), Process (c_process), Executable (c_executable)

trusts_host
Description: Human will type at Host.
Parameters: Human (c_human), Host (c_host)

unwritten_code
Description: Program has yet to be written.
Parameters: Program (c_program)

wtmp
Description: audit log of Host contains login for user Uid.
Parameters: Uid (c_uid), Host (c_host)

The following modes are used in the process domain:

Type	Description
m_wait_pass	Wait on password for login.
m_wait_shell	Wait for shell to launch.
m_wait_fork	Wait for process to fork.

The following actions occur in the process domain:
Launching a shell is comprised of the following steps:
- Enter username. - Enter password. - Launch shell.
The details for each step are given below.

- Enter username.
Action name: ENTER_UNAME
Modes: m_free, m_wait_pass
Parameters: Human (c_human)
Uid (c_uid)
Host (c_host)
Preconditions: There is currently no one logged into the host.
The human is at the host.
Effects: The uid is entered at the host.

- Enter password.
Action name: ENTER_PASSWORD
Modes: m_wait_pass, m_wait_shell
Parameters: Human (c_human)
Uid (c_uid)
Host (c_host)
Info (c_info)
Preconditions: The user has not yet been authenticated at the host.

The uid has been entered at the host.
The password info is the correct password for the user at the host.
The human knows the password info.
Effects: Any keylogger attached to the computer knows the password info.
The audit log for the host contains the login info for the user.
The user has been authenticated at the host.

- Launch shell.
Action name: LAUNCH_SHELL
Modes: m_wait_shell, m_free
Parameters: Uid (c_uid)
Host (c_host)
File (c_file)
Process (c_process)
Pbefore (c_process pid, used for process id ordering)
Preconditions: The user has been authenticated at the host.
The file is a login shell.
The file is accessible from the host.
The process is not already in use.
Effects: The process is running on the host as the uid.
All the programs in the file are running.

Executing a file is comprised of the following steps:
- Fork a child process. - Execute the file.
The details for each step are given below.

- Fork child process.
Action name: PROC_FORK
Modes: m_free, m_wait_fork
Parameters: Host (c_host)
Uid (c_uid)
Process (c_process)
Program (c_program)
Preconditions: The host has the process.
The host is running the process as the uid.
The host is running the program with the process id.
The program can execute new programs.
Effects: The uid has forked a new process on the host.

- Execute file.
Action name: EXEC_FILE
Modes: m_wait_fork, m_free
Parameters: Host (c_host)
Human (c_human)
Uid (c_uid)
Gid (c_gid)
Process (c_process)
Pbefore (c_process pid, used by process id ordering)
File (c_file)
Preconditions: The human is at the host.
The human will run the file.
The uid has forked a new process on the host.
The host is not already running the process.
The file is executable.
Effects: The uid is no longer forking a new process.
The file is executing as the uid on the host.

Other process actions are below.

- Lock Screen.
Action name: LOCK_SCREEN
Parameters: Human (c_human)
Host (c_host)
Uid (c_uid)
Preconditions: The host is not locked.
The human is at the host.
The uid has been authenticated on the host.
Effects: The host is locked.

- Unlock Screen.
Action name: UNLOCK_SCREEN
Parameters: Human (c_human)
Host (c_host)
Uid (c_uid)
Info (c_info)
Preconditions: The host is locked.
The human is at the host.
The human trusts everyone in the room.
The uid has been authenticated on the host.
The human knows the info.
The info is the login password for uid on the host.
Effects: The host is unlocked.

- Do File Update.
Action name: DO_FILE_UPDATE
Parameters: Human (c_human)
Uid (c_uid)
Gid (c_gid)
Instruction (c_instruction)
Host (c_host)
Process (c_process pid, used by process id ordering)
Pbefore (c_process)
File (c_file)
Preconditions: The human knows and trusts the instruction.
The instruction is for the human.
The instruction tells to update the file on the host.
The human is at the host.
The uid of the human is authenticated on the host.
The host does not yet have the process.
The file is executable.
Effects: The file update is executed.

- Instruct File Update.
Action name: INSTRUCT_FILE_UPDATE
Parameters: Writer (c_human)
Human (c_human)
Host (c_host)
File (c_file)
Instruction (c_instruction)
Preconditions: Human 1 is an insider.
The instruction is given by nobody.
Effects: The instruction is given by human 2.
The instruction is no longer given by nobody.
Human 1 knows the instruction.
The instruction tells to update the file on the host.

Example

The domain and problem will be input into the simulator using STRIPS-style PDDL2.1 format. Below is a portion of a problem definition:

```
(define (problem DEMO)
  (:domain DMS)
  (:objects
   proc_0 proc_1 proc_2 proc_3  - c_process
   email_1 email_2 email_3 - c_email
   y_winbork - c_file
        ...
  )
  (:init
   (my_session bob dmss1)
   (dmsacl_read e_secret_doc greg_uid)
   (not (is_open door_1))
   (fw_allows_nesadmin fw1)
   (my_email bob email_1)
   (my_session greg dmss2)
   (has_keylogger bob keycap)
        ...
  )
  (:goal (knows bob secret_info))
  (:metric minimize (+ (total-time) (* 10 (detection_risk))))
)
```

The plan should be a set of ordered actions, for example:

```
0: SIT_AT_HOST ADAM BIGFOOT ADAMS_OFFICE
1: ENTER_UNAME ADAM ADAM_UID BIGFOOT
2: ENTER_PASSWORD ADAM ADAM_UID BIGFOOT ADAM_PWD
3: LAUNCH_SHELL ADAM_UID BIGFOOT B_WEXPLORE PROC_3
        ...
27: DMS_SEND DMSS1 YETI PROC_1 E_SECRET_DOC
28: READ_DOC BOB YETI PROC_1 E_SECRET_DOC BOBS_OFFICE SECRET_INFO
```

On successfull fullfillment of the goal, the simulator will return the Metric result if there is one and ”Success” if there is not. If there is an error executing the plan, then the step which contains the error, and as much information as possible about where the failure took place will be displayed.

Acknowledgements

The material in this archive was developed by Adventium Labs, funded by ARDA under contract NBCHC030080.