Visually Assessing Possible Courses of Action for a Computer Network Incursion

Grant Vandenberghe
Visually Assessing Possible Courses of Action for a Computer Network Incursion

GCIH Gold Certification

Author: Grant Vandenberghe
Adviser: Rick Wanner

Accepted: February 26th, 2007
Abstract

When a computer is compromised a standard incident handling process is followed to mitigate damage, expunge the attack, and recover the system. In order to prevent possible spread of an attack, the incident handler will try to isolate the victimized system. Isolation may involve disabling the asset or blocking the attacker’s access. This report presents a tool that allows the security analyst to visually evaluate various containment options to minimize operational impacts.
Executive Summary

Background: Within any computer network there are vulnerabilities both known and undiscovered that can be exploited by a skilled attacker. Network defensive measures such as an intrusion detection system (IDS) are not reliable enough to block some types of attacks and only record possible incursions in the event logs. When an analyst reviews the event logs and verifies an incursion a security incident is declared. Upon declaration a series of containment and recovery steps are taken. This study proposes that the standard incident handling process be extended to include a course of action step. Often there is more than one possible course of action and this study presents a tool that is able to service this step.

Principal Results: The course of action tool is intended to predict the consequences of a containment decision by relating its effects to ongoing commercial/military operations. The tool supports three types of actions by simulating the effect of disabling a link, device or service. The tool uses its underlying knowledge of the network, routing infrastructure and service provisioning to recognize potential operational impacts. The results of the simulations can be presented either as text or in a logical network diagram form.

Significance of Results: Embedded within the tool is a four-stage algorithm that quickly and efficiently determines the impact of an outage or planned change. The combined algorithm integrates well into a large organization that has separate networking, service provisioning, mission planning and incident response teams. When the four algorithms are combined the result is a new
approach that is faster than a simulation but provides less granular results. These results are presented visually in a unique form that makes it easy for an analyst to interpret.

Two of the four algorithms deserve special mention for their uniqueness and applicability to a broader range of graph theory problems. The network fragmentation algorithm, called the “Rock in Pond” algorithm, determines how a network becomes fragmented if a node is removed. The “Product of Primes” approach for testing graph hierarchy quickly resolves dependencies in a single database query.

Future Application: This research was initially developed to help meet the requirements for an Applied Research Program proposal involving development of a Course of Action simulation for the Department of National Defence in Canada. The results will be made available to that effort.
Outline

1. Background .................................................................................................... 7
2. Purpose ............................................................................................................ 14
3. Course of Action ….......................................................................................... 14
4. Course Of Action Methodology ........................................................................ 17
   4.1 Network Fragmentation Algorithm .............................................................. 18
   4.2 Service Transaction Hierarchy Algorithm ......................................................... 19
   4.3 Operations Diagram / Operational Disruption Algorithm ............................. 24
   4.4 Quantifying a Course of Action ...................................................................... 28
5. Implementation ................................................................................................. 30
6. Hypothetical Demonstration ............................................................................. 32
7. Discussion and Conclusions ............................................................................... 40
8. References ...........................................................................................................
Acknowledgements

I would like to thank Craig Burrell, Vincent Taylor, and Douglas Vandenberghe for the editorial feedback on the report. Their input is greatly appreciated and this work has been substantively improved through their efforts.
1. **Background**

The keys to preventing and addressing computer crime are found on three principles: means, motive, and opportunity. As shown in Figure 1 security analysts attempting to prevent a security incident can remove the vulnerabilities or the means of attacking the system. They can also change the network's availability to the attacker through firewall rules and other defensive measures. The analyst can also indirectly affect the motivation of an attacker by a strong reputation for prosecuting transgressors.

![Means, motive, and opportunity (MMO)](image_url)

*Figure 1. Means, motive, and opportunity (MMO)*

The role of preventing a security incident is typically assigned to a network
and vulnerability assessment group. Such a group will follow a risk evaluation process such as that laid out in NIST 800-30 to assess vulnerabilities and prepare defences [1]. This process, shown in Figure 2, is not being well addressed by the current generation of risk assessment tools [2]. One of the more advanced tools is Skybox Assure [3] that is able to combine automated risk assessment, network policy compliance, and firewall auditing. Another tool is Alcatel's 8950 [4] series risk management product line which is able to relate vulnerabilities to threats and relate those risks to operational impacts. There are also smaller niche tools like MulVAL [5] that help predict the effect of exploiting a series of vulnerabilities to achieve a desired effect and RiskWatch [6] that helps assign costs to a risk.

![NIST 800-30: Risk assessment process](image)

> **Figure 2. NIST 800-30: Risk assessment process**

Although prevention is key to the defence of a computer network it does not make the network impervious to attack even with constant patching. Unknown vulnerabilities may exist for many years before their discovery [7]. As a result, it is foreseeable that a skilled adversary could compromise a computer network. Once a system has been attacked the security professional's role changes from
prevention to protection and eradication. The means, motive and opportunity (MMO) principles still apply with the motive now being inferred from the actions of the attacker and categorized as follows:

- Obtain Information - downloading sensitive information
- Withhold Information - denial of service, ransomware
- Disseminate Information - botnet, covert channels, spam
- Misinform - rootkits, phishing, fraud
- Sabotage – delete/corrupt data, disable/destroy equipment.

Recognition of how the attacker’s actions fulfill his motivation is very important in determining an appropriate response to an attack. From a risk perspective due to the homogenous nature of many computer environments, a single vulnerability can result in the compromise of a vast number of machines. Often the desire, not the opportunity, is the determining factor. To deal with the attack an incident handler is required. Unlike a Network Vulnerability Assessment Team (NVAT) that is responsible for assessing risks, the incident handler assesses damage and tries to efficiently recover from it. The tools used in risk assessment tend to prioritize patches and maintenance work. Vulnerability tools are not well suited to exploring the ramifications of a single event because they overlay what might happen in the future on top of what has happened in the present.

There are a number of decision-making processes available and in a military sphere one that is widely accepted is a decision-making process known as the Observation, Orientation, Decision, and Action (OODA) loop [8]. The OODA loop
was developed by Col. John Boyd, USAF (Ret) for decision-making by pilots during the Korean War and its principles are still routinely used in many command and control environments. The OODA loop is shown in Figure 3 and involves gathering information, fusing the information, deciding upon a course of action, and then responding.

![Figure 3. OODA loop](image)

There is also a number of other published incident handling processes. Of the 13 processes listed in a survey by Alberts [9] the majority closely resemble the process shown in Figure 4. This incident handling process was initially developed by the United States Department of Energy, and subsequently refined by the United States Navy and SANS organization [10].
A comparison between the incident handling process and OODA loop is shown in Figure 5. As can be seen there is a step missing between incident recognition and taking action to solve the problem. To its credit, the standard incident handling process does decide that an incident exists but it stops short of devising a course of action to deal with the problem. There are incident handling forms available through the SANS organization [11, 12]. These procedural checklists fall short of being a forward planning tool.
This study proposes that an additional Course of Action step be added to the incident handling process as shown in Figure 6. The purpose of this step is to quickly plan a strategy to contain and restore a compromised system and ensures that the analyst obtains an agreement with stakeholders before taking action. In any organization the latter agreement should be present in some form so that security analysts would have some type of tacit or formal approval before taking action. Forward planning allows the incident handler to better project the image of composed professionalism rather than cowboy-like responsiveness.
The impact of taking a defensive action within a complex computer network environment can be assessed manually or with the help of a decision support tool. The advantage of a tool is that it would ensure quick and efficient action planning. At present there are no established tools for determining a course of action. However, there are several types of tools that could be adapted to support this activity. For example, there are several network simulation packages available such as Opnet-Modeller [13], Qualnet [14], or Shunra [15]. These simulators require considerable effort and expertise to setup and maintain and do not conveniently associate a device with one or more missions. This capability is needed to compare the importance of assets impacted by a given course of action.

There are some containment and remediation tools such as Radar-ICX [16] for containing a security incident. These tools use a set of user-defined thresholds.
to drive a quarantine action and can be very useful in addressing highly aggressive network attacks. However a system based on thresholds is not equipped with the understanding of how a quarantine action will impact ongoing missions.

In a similar vein, there are a number of ongoing research activities looking into the creation of an automated containment tool for blocking the propagation of an internet based worm [17, 18]. But again their intent is directed towards containment and not general incident response planning.

Outside of computer networks in other emergency response domains there are some situational awareness tools such as Black Coral [19] that can predict the impact of a disaster, such as a chemical gas leak, and display it in a geospatial environment. Tools like this allow a team of users to respond in a coordinated manner to an event and help ensure the rapid containment of a situation. Unfortunately Black Coral is not intended for assessing isolation options in a computer network.

In summary, there are a number of tools available to assess, detect, predict and respond to a security incident. Most of these tools lack knowledge of the network, its interrelationships and the operational effects a given change may have. The remainder of the tools are large complex applications whose intended purpose is not directed to the needs of an analyst. It would be useful to have a tool that allows a security analyst to quickly compare potential courses of action from an operational impact point of view.

In place of using a network simulation, this study has chosen to use a client-server dependency hierarchy to model network dependencies. The dependency
hierarchy is similar to a fault tree diagram, commonly used in the failure modes and effects analysis (FMEA) area. Fault trees have been used in the aeronautics and electronic industry to prevent massive system failures by spotting when environmental or operational conditions exceed design restrictions [20-22]. FMEA tools such as Relex [23] are commonly used to prevent massive system failures by tracking large numbers of complex interdependencies. The techniques employed in FMEA, like fault trees, can be applied in a very simplistic form to computer networks. The main advantage of this technique is speed since the problem is solved once, as opposed to iteratively in the case of a simulation.
2. **Purpose**

The research examines the feasibility of adding a course of action step to the standard incident handling process. The results of this analysis will be used to direct a containment or isolation action during a security incident in a way that is minimally disruptive to ongoing commercial/military operations. In support of this additional step a fault tree-like analysis tool is developed to assess a proposed course of action. Naturally, permission to proceed with any defensive measure would require formal administrative approval and a simple form is suggested.

3. **Course of Action**

Computer network incursions vary in virulence and destructive capability. In the case of a particularly nasty incursion attacking a server, the only prudent course of action would be to immediately disable the server. In other cases the attack may be mitigated through a planned response. Regardless of the circumstances the organization's security policy should specify what the analyst can do and the context in which it can be done.

For situations that support a planned response, a proposed course of action form is shown in Figure 7. It describes the key actions to be taken during the incident handling process and requires stakeholder agreement. The first section is called “starting premise” and describes what is known about the system and incident.
The forensic analysis section details the steps to be taken to preserve evidence for civil/criminal prosecution. If a system needs to be forensically secured the security analyst can either remove the system from service to do a bit-by-bit backup or remotely copy the disk.

The containment and isolation section of the form allows the incident handler to choose how the system is to be isolated so that it can be repaired. The handler must first choose whether the isolation action is performed locally on the system or remotely on the pathways that lead to the device. Next, the method of isolation
is chosen which may involve disabling a device, link or service. The preferred option will combine the need to block the attacker's perceived end goal while having a minimal impact on business/military operations.

The eradication section follows the containment section of the form. The 5 ways to eradicate the problem are given by the 5 R's: Reduce, Repair, Restore, Rebuild and Replace. The choice of recovery method is highly dependent on the problem, the system's importance, and the containment measures taken.

Once the problem has been removed, the recovery phase begins. Often if a system has been compromised, returning the system to its state just prior to the attack is insufficient. The system will need to be patched or hardened to prevent a repeated incursion. Unfortunately changing the system from its original setup may also cause problems with the existing applications. To further complicate things there is often a knowledge gap between knowing that a system is compromised and knowing how it occurred. Therefore at this point in the process the course of action form is intended to notify stakeholders of the resources that will be needed to facilitate system testing and recovery.

In summary, the course of action template shown in Figure 7 can be completed on a single page and helps ensure that people and resources are on-side to fix a security incident. Some parts of the form may be driven by a company's security policy while others are situation dependent. Since every action has a consequence our ability to pick the action with the least consequences is very important. The impact assessment tool presented in the next section allows the security analyst to quickly evaluate the consequences of different actions as
they relate to computer network operations.
4. **Course of Action Methodology**

To successfully isolate a compromised device it must either be disabled or the path taken by the attacker to the device blocked. Disabling one or more devices, links, or services can block the network access path. To recognize the operational effects of these changes the architecture shown in Figure 8 is proposed.

![Figure 8. High level architecture](image)

The analysis tool proposed here uses four algorithms to categorize and quantify network fragmentation, service impacts and operational ramifications. The network fragmentation algorithm checks whether a proposed change impacts the network’s ability to communicate. The algorithm receives link or device outage data and determines if the result splits the network into fragments. The network
fragments along with device outage information are then applied to a service transaction algorithm. This algorithm is used to recognize service disruptions where a server loses the ability to communicate with its clients. The disrupted services are then applied to an operational disruption algorithm that relates lower level service impacts to high-level mission outages. The combination of the network, service, and operational impacts gives a relatively complete picture of the ramification associated with the chosen course of action. In a situation where multiple courses of action are being compared, an algorithm for quantifying the severity of the proposed change can be determined using the results of the service and operational impact steps. A detailed description of each algorithm is provided in the following sections.

4.1 Network Fragmentation Algorithm

This algorithm tests whether the loss of a node or link will fragment the network and disrupt its ability to communicate. If the requested change does not cause fragmentation then the algorithm indicates that the system is intact. The algorithm has been given the nickname the “rock in pond algorithm” and is considered to be new work.

To demonstrate the algorithm the simple network shown in Figure 9 will be

\[ \text{Figure 9: Simple Network} \]

1 Network impacts are treated as a subset of service impacts in this study. If all missions are proceeding and all services are available then whether a network is fragmented becomes an undesirable but not operationally important condition.
used. It is assumed that the user needs to understand the impact of removing the node shown in red from the network in Figure 10(a). The algorithm begins by assigning each egress link from the red node with a different “wave-front” colour, Figure 10(b). The graph-walking algorithm sweeps the wave front across the network in a manner that is similar to ripples crossing a pond, Figure 10(c). As the graph-walking algorithm crosses the network, loops in the network cause the collision of different wave-front colours. When collisions occur, the colours are combined as shown in Figure 10(d). Once the network has been fully traversed the remaining colours serve as an indicator of network continuity. If the final result is a graph with a single colour then the requested change did not fragment the network. Referring to the network shown in Figure 9, the change fragmented the network into three colour-coded parts: purple, orange and brown. A fragmented network can be beneficial if it blocks the attacker’s ability to interact with the victimized system, but becomes a problem if essential client/server relationships are split. The algorithm described in the next section has been developed to establish whether a broken client/server relationship exists.

![Figure 9. Example network](image-url)
4.2 Service Transaction Hierarchy Algorithm

The disruptive effects of node loss and network fragmentation need to be related to the operation of each service. A service may include self-standing devices but more commonly involves groups of devices that need to exchange information to complete a transaction. When network fragmentation or a device outage occurs, the service’s ability to process transactions may be compromised.

The service hierarchy shown in Figure 11 is modeled as a simplified fault tree. The vertices represent computer devices that participate in processing service transactions and the interconnecting links represent a dependency relationship between devices. Each relationship is considered to be asymmetric in that the device on one end of the link is dependent on the device at the opposite end. It is assumed that the device seeking information is a client while the device providing the information is called the server. Diagrammatically each relationship is
shown with the server placed above the client giving the diagram its hierarchal form. When a client on the diagram has no active relationship with a server it is considered disabled. Thus, if the database at the very top of the diagram were to be disabled all of the underlying devices would be impacted. Alternately, if a redundant server were to break, the underlying clients would continue to run unfettered through the remaining servers.

![Client/server service diagram](image)

**Figure 11. Client/server service diagram**

One way to detect the loss of a dependency relationship is through a recursive database query. Recursive database queries are time consuming for multi-layered diagrams where one query is required to transverse each graph layer beneath a disabled server and a more direct method is preferred.

An alternative to the recursive approach involves using the products of prime numbers to recover all client relationships associated with a disabled device. The advantage of the product of primes approach is that the algorithm is able to
recover all client relationships in a single query. This new algorithm begins by assigning a prime number to each device in the service hierarchy. The smallest prime number is assigned to the top of the tree and then each subsequent prime number is larger until the largest primes are reached at the bottom. Non-redundant devices use a unique prime number while redundant ones share a common prime number. Once the primes have been assigned the Product of Primes (POP) may be calculated for each device. To do this the chains of primes leading back to the root devices are multiplied together as shown in Figure 12. For redundant devices (the pink vertices) the prime number assigned to each device must be raised to the power of the redundancy - in this case 2 x 32 as opposed to two distinct prime numbers if it was non redundant.

To recognize all dependent client relationships a division calculation with remainder check is performed. For example Figure 13 has the Tier 2b web server disabled. The POP for the web server is 10. Querying the database for all devices that have a POP which divides evenly by 10, yet is not equal to 10, results in the dependencies shown in red.
The problem becomes slightly more complex when redundant servers are present because the database query should only return client nodes when all
redundant servers are disabled. To deal with redundant servers an additional Boolean test is used in the database query. The test determines if a redundant server is present in the network. The test seeks vertices for which the POP is not evenly divisible by the server's prime number taken to the exponent of the total number of disabled servers plus 1. As shown in Figure 14 a single side of a redundant server has been disabled. The redundant server has a prime number of 3 and at present only one server is disabled. Since the total number of redundant disabled servers is 1 the divisor for the redundant Boolean test in the query is $3^{1+1}$. Since all the client vertices are cleanly divisible by $3^{1+1}$ this serves as an indicator that there is at least one redundant server left in the network. A second example is shown in Figure 15. Since two servers are disabled with a prime number of 3 then the divisor becomes $3^{2+1}$. Since the clients of the redundant servers are not evenly divisible by $3^{2+1}$ this means that all the redundant servers are disabled and an outage situation exists.
Figure 14. Querying a redundant relationship (one server down)

Count the number of disabled servers (1) and increase the total by 1. Apply this total to the exponent of the prime number divisor associated with the redundancy test ($3^{1+1}$).

Query Where
Remainder(POP/18)=0 and Remainder(POP/3^{1+1})≈0 and POP<>18
Returns Nothing

Since all the devices beneath the disabled server are evenly divided by ($3^{1+1}$) this means that a redundant server is still available to handle transactions.

Count the number of disabled servers (2) and increase the total by 1. Apply this total to the exponent of the prime number divisor associated with the redundancy test ($3^{1+1}$).

Query Where
Remainder(POP/18)=0 and Remainder(POP/3^{1+1})≈0 and POP<>18
Returns 4 devices (shown in red)

Since all the devices beneath the disabled servers do not divide evenly by ($3^{1+1}$) this means that all the redundant servers are disabled.
The principles discussed above can be further extended to fragmented networks through additional query restrictions that are chained together with Boolean “or” operations. However, in a fragmented network each fragment perceives all others as being non-existent. Therefore each fragment must solve the hierarchy problem from its own perspective. This means multiple queries are required before the results may be combined together.

4.3 Operations Diagram / Operational Disruption Algorithm

A user defined operations diagram is shown in Figure 16. There are two types of vertices on the diagram: operational and service related. The service related vertices have client/server relationships that form an underlying service hierarchy. Most networks support multiple services and service relationships that overlap across devices. Since the same network node may support multiple services, a separate hierarchal graph is required for each service. By individually addressing each service, the impact of a hierarchal relationship can be more easily recognized.
Shown in Figure 17 are two services Y and Z in a hierarchal arrangement with the parent service at the top and the child service placed underneath. As shown for service Z, device B relies on device A to provide a lower level service. Subsequently, device B responds to device C in a higher-level service. In effect, the service layering has created a domino type failure: device A impacts device B which impacts device C in service Y. In brief, the existence of layered services requires a bottom up analysis approach. To facilitate the process the services are manually assigned to layers called service planes that are then processed in order from lowest to highest. The services within each layer are processed in a random order.
Figure 17. Chains of client/server dependencies

Figure 18 shows the operations diagram of Figure 16 redrawn with service planes introduced. The vertices on each row of the diagram share a service plane and a common level of abstraction. A complete description of each service plane is presented below:

- **Mission Service Plane**: The vertices in this plane describe the status of all devices that are associated with a particular military mission or business operation.

- **Capability Service Plane**: The vertices in this plane describe the major categories of software present in the network. The links between the mission and respective capability describe the generic software needs of a particular mission.

- **Application Service Plane**: Describes the software applications installed on a network. Multiple links between the application and capability layers occur because some applications are platform specific in a multi-platform environment; and because an integrated application may support multiple capabilities.

---

2 Please note these service planes are configurable and can be redefined or expanded to meet a user’s needs.
**Middleware Service Plane:** Provides specific features to the applications on the network. The linkages between the middleware and application service planes reflect the distributed features, web services or database connections needed by an application.

**Protocol Service Plane:** Describes the communication protocols used by the middleware or software applications. The protocols are physically observable by monitoring the network traffic communications.

Vertices on the various service planes may have relationships with multiple parents and children. For example as shown in Figure 18 the EMAIL vertex is associated with two missions. When a problem occurs in the email service the result could impact one or both missions. To properly associate impacts, each mission must have a list of devices associated with it\(^3\). This “membership list” is an inventory of devices that is referenced by the algorithm to assess operational

---

3 This list is non exclusive which means that one device may be part of multiple missions.
impacts⁴.

The algorithms previously discussed in Section 4.2 provide a complete list of devices impacted by a device or service outage. These outages will now be associated with specific operational impacts through the membership lists. It should be pointed out that the service transaction hierarchies and operational diagram could all be combined into a single mammoth relationship diagram. However doing so would make the user’s job more difficult. The use of a membership list with inheritance dramatically simplifies data entry requirements because low-level dependencies are separated from high-level deployment requirements. As an example, think of a situation where a mission planner knows which devices are to be deployed but not necessarily all underlying dependencies. Let us assume that three users with laptops are being deployed to the field. User A requires a laptop with email, user B with web viewing capabilities and finally user C with both applications. Each laptop has been preconfigured and the mission planner simply assigns the laptops to the mission. Figure 19 shows the resulting operations diagram. During the mission the email server becomes disabled. The service transaction algorithm recognizes that this outage will impact laptops A and C. The crippled devices are inherited into the mission using its membership list. An alert is generated for laptops A and C. This chain of events occurred without the mission planner even recognizing the mission is indirectly associated with the email

⁴ A parent must share a relationship with a child for a device outage to matter. A device may have an application installed; however, unless the mission needs the application its state is irrelevant.
server.

In summary, a recursive algorithm is used to determine the operational impact of a change or outage. The algorithm starts at the bottom of the hierarchy and works its way to the top. It compares the disabled devices for each child service to the contents of the membership lists defined in each parent vertex. In effect, the recursive algorithm allows higher-level vertices the ability to inherit the outages detected in lower layers. Once the recursive algorithm has completed, the operations impacted by the change will be known.

4.4 Quantifying a Course of Action

In a large network there is a range of containment actions that can be taken. To properly quantify the impact of a course of action, a metric for comparison is
required. This study uses the following weighted formulas to evaluate a particular course of action as it relates to a service, mission or overall operations.

\[
\text{Service/Mission Impact} = \sum_{d \in D_s} V(d) V(s)
\]

\[
\text{Operational Impact} = \sum_{s \in S} \sum_{d \in D_s} V(d) V(s)
\]

Where

- \( V(d) \) is the value of device \( d \)
- \( V(d) \) is the value of service \( s \)
- \( D_s \) is the set of all devices that are members of service \( s \)
- \( S \) is the set of all services/missions.

The above equation requires the user to define a value for each mission critical device and service, with higher values indicating greater importance. For example each device could be assigned a value between 1 and 10. The magnitude of the result is a measure of the overall impact of the event.
5. Implementation

The course of action tool discussed in Section 4 was implemented in the Matlab programming environment. The tool is a smaller part of a larger research initiative involved in the advanced analysis and response to network based security events. The course of action tool itself fits inside a network visualization library as shown in Figure 20. To support the course of action tool, an external MySQL database is used as a repository for network topology, asset/service values, and service/operational hierarchies.

This environment allows a security analyst to read and analyze raw data collected from security sensors and network packet loggers. The security events
need to be validated by an analyst using the network traffic analysis library. Once validated, the security incidents can either be forwarded to the network presentation engine, packet routing engine and/or the course of action tool. The network presentation engine presents a logical network diagram of the computer network and the network update engine receives information about changes to the state of devices on the network diagram. The target of an attack can be highlighted on the network diagram based on the results from the network traffic analysis library. The routing engine is used to predict the path used by an attacker to reach the victimized system. The results are sent to the network presentation engine to present the results visually. The course of action tool uses its internal algorithms to predict the impact of the security incident on ongoing operations. The results can be output in either report or graphical form. The graphical form makes use of the network presentation engine for final display.

The Matlab environment is highly interactive. The user can draw upon a variety of resources seamlessly to recognize a security incident and to determine an appropriate course of action. The course of action tool assists in this process. A practical example describing how all these functional capabilities fit together is described in the next section.
6. **Hypothetical Demonstration**

The following demonstration illustrates how the course of action tool could be used during a response to a security incident. The multi-step process being followed is shown in Figure 21. The solution process covers the Observation, Orientation, and Decision steps of the OODA loop.

![Figure 21. Hypothetical demonstration solution process](image)

**Step 0: Security event detected:** The security event log records some unusual activity.

**Step 1: Security incident identification:** Using the security event as a point of reference a detailed traffic review is conducted and uncovers that the DNS server has been compromised. The results also show that the attacker is using...
the DNS port as a type of covert channel for unauthorized communication purpose.

**Step 2 - Assess value of a victim:** The effect of disconnecting the internal DNS is shown in Figure 22. The internal DNS server is shown in red while the nodes adversely impacted are shown in yellow. Based on the number of nodes impacted it is easy to see that unplugging the server would be highly disruptive and alternative courses of action should be considered.
Figure 22. Effect of disabling DNS server
Step 3 – Find the attacker's communication path: Since the attacker is using a covert channel to access the internal DNS, one way to contain the situation would be to block the attacker's access to the system. To determine the path of the attacker a route trace is performed. Figure 23 shows the results of the trace in red.
Step 4 – Validate that attack path is unique: In reviewing the attack path it becomes clear that one potential course of action would include blocking the attacker’s communication channel at the firewall. To be effective this plan must isolate the attacker from the victim. The connectivity and communication algorithm discussed in Section 4.1 can produce a visual representation of network fragmentation. Figure 24 shows two different isolation actions. Each diagram shows the disabled devices in red while surrounding devices are colored based on the network fragment of which they are a member. The network shown on the left has one firewall disabled. As can be seen the attacker and DNS server share the same network fragment and thus isolation is not guaranteed. In the network on the right both firewalls are disabled which results in two network fragments (shown in yellow and blue) on the diagram. This suggests that the attacker cannot circumvent the proposed action.
Step 5 – Testing course of action alternatives: Having found a valid set of isolation points, the next step involves submitting a potential action to the isolation assessment tool. The effect of blocking DNS communication through both firewalls is shown in Figure 25. By comparing Figure 22 and Figure 25 it becomes easy to see that blocking DNS service at the firewall would be far less disruptive than shutting down the internal DNS server.
In more complex scenarios several alternatives may need to be examined in detail. A more comprehensive user interface is available and illustrated in Figure 26. The tool's advanced interface is able to display several views of the network.
including:

**Logical view:** Presents the network and results in a form tuned by the user. Non essential nodes are minimized so that the user can concentrate on the critical system nodes rather than the whole network.

**Operations view:** Helps prioritize responses when multiple attacks are in play and multiple missions impacted.

**Transaction View:** Shows why a given service is reporting a problem.

**Impact Assessment Table:** Quantifies the impact associated with a given course of action so that alternatives can be numerically compared.

---

*Figure 26. Course of action tool advanced user interface*
Step 6 – Complete the course of action form: Once the best course of action has been established the course of action form shown in Figure 27 can be completed. At this point the attacker has been detected but the exact vulnerability is not known. At present no evidence of financial loss, criminal action, or fraud exists and as a consequence a full forensic investigation will not be done. However a system backup will be conducted. To isolate the system, the firewall rules will be changed to block the internal DNS server from accessing the Internet. This action will disable the attacker's ability to access the system. Unfortunately this action will also prevent internal users from resolving unknown external DNS entries. The system will be repaired and the internal DNS server will be reconfigured to forward future queries through the external DNS server to add an additional layer of defence.
### Computer Security Incident Course of Action Form

**Starting Premise**
- Internal DNS has been compromised and is using a covert channel to communicate with attacker.

**Forensic Analysis**
- Required: [ ] Yes [x] No
- Backup Vantage Point: [x] Local [ ] Remote

**Containment: Method Of Isolation**

<table>
<thead>
<tr>
<th>Recommended Isolation Action</th>
<th>Isolation Impact:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local</td>
<td>External DNS resolution will become unavailable.</td>
</tr>
<tr>
<td>Remote</td>
<td>Firewall</td>
</tr>
</tbody>
</table>

**Eradication**
- [ ] Reduce – permanently disable
- [x] Repair – remove source problem
- [ ] Restore – restore data from backup
- [ ] Rebuild – rebuild system from scratch
- [x] Replace – redesign implementation

**Recovery**
- Testing and Commissioning: [x] Yes [ ] No
- Assistance Required: [ ]

**Sign-off**
- Stakeholder: _______________
- CIRT: _______________

*Figure 27. Complete course of action form*
7. **Discussion and Conclusions**

This study has suggested that an additional course of action step be added to the incident handling process. This addition would require that an incident handler identify the effects of his action before disrupting ongoing commercial or military operations. To support the decision-making step a new tool has been developed to simulate the effects of disabling a device, link, or service.

The tool uses four separate algorithms to detect network fragmentation, recognize service outages, project operational impacts and quantify the results. The network fragmentation algorithm is good for handling communication interdependencies. It quickly determines which clients and servers are isolated because of network fragmentation. Subsequent processing stages leverage the fragmentation information to simplify dependency checks. Knowing whether a client and server share the same network fragment allows network interdependencies to be excluded from the service transaction hierarchies. The transaction hierarchy chains are very short since they only contain client and server nodes. This approach is well suited to mobile ad hoc networks where the network changes more frequently than the client/server dependencies and minimizes/simplifies table updates.

The service transaction algorithm uses a product of primes approach to detect service outages. The algorithm runs quickly and efficiently. One of the limitations of the algorithm is that for larger networks the algorithm will generate large integers that need to be stored in a database. Most databases have a maximum integer size that can be supported. For databases that support 32-bit
integers it is estimated that a 70,000-node network could be supported. To support even larger networks other databases that support 64 bit integers can be used [24].

The operational disruption algorithm used to detect operational outages worked well and allows mission planners to quickly deploy resources against a dynamically changing operational environment. Hidden dependencies do not need to be tracked or updated when using the prescribed approach.

The course of action tool is relatively simple to set up and was designed in a way that is easy to maintain. In principle it is flexible enough to handle an unlimited number of services and service planes, and can present detailed or summarized results from a large network. The tool's output supports three graphical representations: logical, transactional, and operational. The tool can also present data in a table-based output format.

As currently existing, the tool is intended to prove a concept and can be used for rudimentary analysis. It is being developed to be part of a larger workbench for security event analysis and response.
8. References


Carnegie Mellon: Software Engineering Institute,
http://www.cert.org/archive/pdf/04tr015.pdf

By Step Computer Security,
http://cio.uiowa.edu/ITsecurity/resources/documents/i.pdf

http://www.sans.org

12. SANS (2006) Sample Incident Handling Forms,
http://www.sans.org/score/incidentforms/index.php


Internet Worm Containment, IEEE Security and Privacy, pages 24-32,

Propagation Sequences that Circumvent a Collaborative Worm Defense,
Proceedings of the 25th International Performance Computing and
Communications Conference April, pages 587-592,
http://www.csl.sri.com/papers/malware2006/

http://www.blackcoral.net/home.htm

Tool, In Proc. 29th Annual International Symposium on Fault-Tolerant
Computing (FTCS'99), pages 232-235


<table>
<thead>
<tr>
<th>Course Name</th>
<th>Location</th>
<th>Dates</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SANS Rocky Mountain 2019</td>
<td>Denver, COUS</td>
<td>Jul 15 - Jul 20, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS San Francisco Summer 2019</td>
<td>San Francisco, CAUS</td>
<td>Jul 22 - Jul 27, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Pen Test Hackfest Europe Summit &amp; Training 2019</td>
<td>Berlin, DE</td>
<td>Jul 22 - Jul 28, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>DFIR Summit &amp; Training 2019</td>
<td>Austin, TXUS</td>
<td>Jul 25 - Aug 1, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Riyadh July 2019</td>
<td>Riyadh, SA</td>
<td>Jul 28 - Aug 1, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Boston Summer 2019</td>
<td>Boston, MAUS</td>
<td>Jul 29 - Aug 3, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Melbourne 2019</td>
<td>Melbourne, AU</td>
<td>Aug 5 - Aug 10, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Crystal City 2019</td>
<td>Arlington, VAUS</td>
<td>Aug 5 - Aug 10, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>Security Awareness Summit &amp; Training 2019</td>
<td>San Diego, CAUS</td>
<td>Aug 5 - Aug 14, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS San Jose 2019</td>
<td>San Jose, CAUS</td>
<td>Aug 12 - Aug 17, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Prague August 2019</td>
<td>Prague, CZ</td>
<td>Aug 12 - Aug 17, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Minneapolis 2019</td>
<td>Minneapolis, MNUS</td>
<td>Aug 12 - Aug 17, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Amsterdam August 2019</td>
<td>Amsterdam, NL</td>
<td>Aug 19 - Aug 24, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Virginia Beach 2019</td>
<td>Virginia Beach, VAUS</td>
<td>Aug 19 - Aug 30, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Chicago 2019</td>
<td>Chicago, ILUS</td>
<td>Aug 19 - Aug 24, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Hyderabad 2019</td>
<td>Hyderabad, IN</td>
<td>Aug 26 - Aug 31, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Copenhagen August 2019</td>
<td>Copenhagen, DK</td>
<td>Aug 26 - Aug 31, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Brussels September 2019</td>
<td>Brussels, BE</td>
<td>Sep 2 - Sep 7, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Canberra Spring 2019</td>
<td>Canberra, AU</td>
<td>Sep 2 - Sep 21, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Munich September 2019</td>
<td>Munich, DE</td>
<td>Sep 2 - Sep 7, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Philippines 2019</td>
<td>Manila, PH</td>
<td>Sep 2 - Sep 7, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Network Security 2019</td>
<td>Las Vegas, NVUS</td>
<td>Sep 9 - Sep 16, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Oslo September 2019</td>
<td>Oslo, NO</td>
<td>Sep 9 - Sep 14, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Dubai September 2019</td>
<td>Dubai, AE</td>
<td>Sep 14 - Sep 19, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Raleigh 2019</td>
<td>Raleigh, NCUS</td>
<td>Sep 16 - Sep 21, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Rome September 2019</td>
<td>Rome, IT</td>
<td>Sep 16 - Sep 21, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Columbia 2019</td>
<td>OnlineMDUS</td>
<td>Jul 15 - Jul 20, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS OnDemand</td>
<td>Books &amp; MP3s OnlyUS</td>
<td>Anytime</td>
<td>Self Paced</td>
</tr>
</tbody>
</table>