Project IST-1999-11583

Malicious- and Accidental-Fault Tolerance for Internet Applications

Architectural Analysis of MAFTIA's Intrusion Tolerance Capabilities

Ian Welch, John Warne, Peter Ryan, Robert Stroud

University of Newcastle upon Tyne

MAFTIA deliverable D99

Public document

February 3rd 2003

Abstract

The purpose of this deliverable is to provide an analysis of MAFTIA's intrusion tolerance capabilities at an architectural level. We first summarise the various architectural concepts and mechanisms that MAFTIA has developed for constructing intrusion tolerant systems. We then present a realistic "use case" for the MAFTIA architecture, based on a simplified but realistic e-commerce application. Using a methodology based on fault trees, we provide a representative but by no means complete set of attack scenarios, which we then analyse in order to highlight the ways in which MAFTIA's architectural mechanisms support the construction of intrusion tolerant Internet applications. Finally, we conclude the document with a discussion of the overall MAFTIA approach to achieving intrusion tolerance, identifying the key architectural components, and highlighting areas for future research.

Table of Contents

List of Figures

Chapter 1 Introduction

The MAFTIA project has developed a range of mechanisms and protocols for building intrusion tolerant systems. The purpose of this deliverable is to demonstrate and evaluate MAFTIA's intrusion tolerance capabilities at an architectural level by demonstrating their effectiveness in a hypothetical yet practical setting. To this end, the analysis is based on a simplified but realistic e-commerce application, TRADEZONE, originally described as a possible use case for MAFTIA in [MAFTIA 2000], and further elaborated herein.

This deliverable presents the results of this evaluation, and is intended to complement MAFTIA deliverables D21 [Powell & Stroud 2003] and D22 [Adelsbach & Creese 2003], which together describe the concepts and principles underpinning the MAFTIA architecture, and present a variety of techniques for ensuring and evaluating MAFTIA's intrusion tolerance capabilities.

The evaluation work described in this document was performed in three major phases:

i. An **information-gathering phase**, summarising MAFTIA's collective design principles, its technical artefacts, and its system dependability assumptions, with specific reference to *open (Internet) distributed systems intrusion tolerance*;

ii. An **analysis phase**, identifying a representative set of MAFTIA's intrusion tolerance capabilities, the defences it provides against attackers, including *hypothesised attacks with fault-tree analyses*, presenting the likely actions of an attacker attempting to manipulate an intrusion tolerant MAFTIA-compliant TRADEZONE application;

iii. A **summary phase**, presenting a synopsis of the analysis - covering the effectiveness of MAFTIA's countermeasures against *example system intrusions*, together with an indication, where appropriate, of *future research directions* needed to improve or add to MAFTIA's intrusion tolerance capabilities.

The results of our evaluation are presented in the rest of this deliverable as follows. Chapter 2 contains a summary of the various architectural concepts and mechanisms that MAFTIA has developed for constructing intrusion tolerant systems. Chapter 3 presents a realistic "use case" for the MAFTIA architecture, based on a simplified but realistic e-commerce application. Using a methodology based on fault trees, we provide a representative but by no means complete set of attack scenarios, which we then analyse in order to highlight the ways in which MAFTIA's architectural mechanisms support the construction of intrusion tolerant Internet applications. Chapter 4 contains a discussion of the overall MAFTIA approach to achieving intrusion tolerance, identifying the key architectural components and design principles highlighted by the analysis in Chapter 3, and Chapter 5 concludes the document, identifying areas for future research.

Chapter 2 An overview of MAFTIA's approach to building intrusion-tolerant systems

Whilst most of the information presented in this section is scattered throughout the various MAFTIA deliverables, and the many other documents they reference, it is considered useful here to provide a reference summary of MAFTIA's principles and techniques for the purpose of assisting the reader throughout the analysis and discussion in the rest of this document.

2.1 Principles of intrusion tolerance

A **dependable** system [Avizienis *et al.* 2001] is defined as one that is able to deliver a service that can justifiably be trusted; attributes of dependability include **availability** (*readiness for correct service*), **reliability** (*continuity of correct service*), **confidentiality** (*prevention of unauthorised disclosure of information*), and **integrity** (*absence of improper system state alterations*). **Security** is the concurrent existence of a) availability for authorized users only, b) confidentiality, and c) integrity, with 'improper' taken as meaning 'unauthorized'.

MAFTIA uses fault tolerance techniques to build dependable systems that are *intrusion tolerant*, that is, able to continue providing a secure service, despite the presence of malicious faults, i.e. deliberate attacks on the security of the system. Such faults are perpetrated by attackers who make unauthorised attempts to access, modify, or destroy information in a system, and/or to render the system unreliable or unusable. Attacks are facilitated by vulnerabilities, which are faults in the requirements, specification, design, implementation, and/or configuration of a system. A successful attacker is said to be an intruder, and a successful attack results in an intrusion upon the system.

MAFTIA distinguishes between *attacks*, *vulnerabilities*, and *intrusions* as three types of interrelated *faults* [Powell & Stroud 2003]:

- **attack**: *a malicious interaction fault, through which an attacker aims to deliberately violate one or more security properties; an intrusion attempt.*
- **vulnerability**: *a fault created during development of the system, or during operation, that could be exploited to create an intrusion.*
- **intrusion**: *a malicious, externally-induced fault resulting from an attack that has been successful in exploiting a vulnerability.*

The related concepts of *intrusion tolerance* and *intrusion detection* are defined as follows:

- **intrusion detection**: *the set of practices and mechanisms used towards detecting errors that may lead to security failure, and diagnosing intrusions and attacks*.
- **intrusion tolerance**: *the means to provide a service implementing the system function despite intrusions*.

Attacks may be viewed either at the level of the human activity of the attacker, or at the level of the resulting technical activity observable within the considered computer system:

- **attack** (human): *a malicious human interaction fault whereby an attacker aims to deliberately violate one or more security properties*;
- **attack** (technical): *a malicious technical interaction fault aiming to exploit a vulnerability as a step to achieving the final aim of the attack.*

In general, an intrusion can result whenever an attacker is successful in exploiting a vulnerability with respect to any mechanism of a system. If that intrusion is not tolerated, then this can lead to a failure of the mechanism, which could in turn introduce a vulnerability in

other parts of the system that depend on the mechanism, allowing the original attack that caused the intrusion to propagate further into the system.

In its provision of intrusion-tolerance strategies and techniques, MAFTIA builds on the successful work of others, but provides additional strategies and techniques of its own.

One such earlier work was DELTA-4 – an EU-funded research project [Powell *et al.* 1988]. DELTA-4 was one of the first attempts to build a fully functional distributed, secure and robust replicated system. It developed a general architecture for dependable distributed systems. Part of the architecture [Deswarte *et al.* 1991] provided intrusion-tolerant services for data storage with secured access to authenticated, authorised users. Secrecy was afforded by encryption and data fragmentation, and availability by replicated data fragments. Secret encryption key sharing was supported, but no computations on shared secrets or robust protocols were implemented. DELTA-4 assumed a synchronous communications network.

It is instructive to briefly revisit the principles and techniques of fault tolerance here because they are directly relevant to MAFTIA and its philosophy for addressing the issues of intrusion tolerance.

MAFTIA, as in DELTA-4, applies the principles of *fault tolerance* to achieve its lines of defence against adversaries; namely, *error processing* [Avizienis 1967] and *fault treatment* [Anderson & Lee 1981]. *Error processing* (comprising *error detection*, *damage detection, damage confinement*, and *error recovery*) is aimed at removing errors from the computational state, if possible, before failure occurrence. *Fault treatment* is aimed at preventing previous faults from being exploited, and thereby inhibiting the success of similar attacks.

These principles when applied to systems comprising redundant hardware and software components can be supported by manifold techniques:

- ***self-checking computations*** *(capable of detecting anomalous events in expected program behaviour)* and ***agreement protocols with majority voting*** *(capable of achieving error recovery through fault masking);*
- ***threshold-cryptography*** *(capable of reducing the risk of compromising secret cryptographic keys by using split key management),* ***design diversity*** *(capable of providing functionally equivalent computations with dissimilar design and implementations to disallow an attacker from exploiting the same vulnerability in a replicated component), and* ***synchronous and asynchronous network communication protocols*** *(capable of reliable ordered atomic broadcast and secure causal atomic broadcast for general Byzantine agreement, with strong defences against attacker interference)*;

and finally,

- ***capability-based authorisation services*** *(capable of restricting access to all system service users according to the "least privilege" principle).*

These principles and techniques form the basis of the MAFTIA architecture and its approach to intrusion tolerance, as reviewed in Section 2.3.

2.2 Attacker profiles

MAFTIA classifies attackers according to their privileges with respect to the system*:*

Outsiders who have no given system security privileges

This is the most publicised type of attacker, but possibly not the most important, since he or she must successfully grapple with and bypass both the physical and logical security capabilities of a supposedly secure system - even though in current Internet applications the success of this type of attack is becoming all too frequent!

This kind of attacker doesn't possess legitimate "login" capabilities to any external firewalls or internal hosts of the system. Associated attacks will try to bypass the capabilities of the firewall systems, the Internet and its application services, the operating system and its utilities, and the middleware and its encapsulated applications - each attack in turn to whatever extent the attacker is able.

Once the attacker has succeeded in intruding upon a system by exploiting a vulnerability that can be targeted from outside the system, he or she will attempt to exploit further vulnerabilities in order to continue the attack. For example, a buffer overflow attack on a Web service application might allow the attacker to gain access to the "root" privileges of a host, and thereby enable him or her to cause further intrusions in the middleware or application services. There might even be opportunities for the attacker to install *operating system kernel root-kits* that provide all or selected operating system privileges, whilst concealing the presence of the attacker within the system.

An attacker may also try to exploit existing trust relationships between users and hosts in the system, for example, by modifying DNS caches to allow host site spoofing.

Following a successful attack, the intruder might plant Trojan horses, logic bombs, or some other trapdoor mechanisms into the system, and thereby facilitate further opportunities for later attacks. A clever attacker will invariably attempt to erase any trace of his or her tracks to conceal their intrusions.

Insiders who have specific but limited system security privileges

This type of attacker attempts to extend his or her privileges by discovering vulnerabilities in the system's security defences, and thus gain access to unauthorised services.

Such attackers have legitimate "login" access to specific protected facilities of the system, and are, therefore, potentially much more powerful than intrusive outsiders. They may have legitimate access to insecure scripts and faulty administrative services that are not accessible to outsiders, and may therefore be able to exploit vulnerabilities in these internal mechanisms.

Malicious system security administrators who have extensive security privileges

This type of attacker abuses their extensive and legitimate rights by performing illegitimate actions that violate the system's security policy - *clearly, the most dangerous kind of intruder!*

In order to prevent this kind of attack, it is necessary to ensure that no single administrator has complete control over critical parts of system Thus, one way to safeguard against this kind of abuse is to ensure that a system's security duties and associated obligations can only be performed collectively by *multiple, independent, non-colluding, administrators*.

2.3 MAFTIA's capabilities for intrusion tolerance

Intrusion tolerance is concerned with ensuring that a system continues to provide security guarantees in spite of partially successful attacks. Five specific capabilities are identified: (i) *intrusion detection*; (ii) *group communication protocols based on generalised adversary structures*; (iii) *cryptographic techniques*; (iv) *data fragmentation and scattering*; and finally, (v) *access control*. Collectively, these capabilities provide greater strength against intrusions than each method alone.

Each of these intrusion tolerance capabilities is now described in more detail. Their application to a realistic application scenario is examined in the next chapter.

Intrusion detection - *aimed at detecting different kinds of system misuse and abuse.*

These methods are based on the hypothesis that system security violations can be detected by monitoring a system's activities in real-time, and by auditing a system's activities in post-mortem time. They are designed to detect anomalous system behaviour, hopefully soon enough to prevent attackers from realising significant system damage. Whilst not always successful in this regard, they also have a useful role to play in providing monitored information for later system behavioural analysis, needed to assist in the human determination of system intrusions and possible countermeasures.

MAFTIA defines a sensor-based model for the detection and prevention of system attacks. It does this by conjointly signalling interested system components about detected suspicious events, and thereby enabling them to take defensive actions wherever appropriate, and also by referring such events to System Security Officers (SSOs) for analysis and possible external remedial action. Such methods are intended to prevent attacks from being successful, and thereby preventing intrusions.

MAFTIA also defines a methodology for combining different Intrusion Detection System (IDS) services for enabling increased coverage and protection against attacks and intrusions, and, in this way, correlating the results of different IDS detection mechanisms to determine common system attacks and their potential intrusions (see Chapter 4 of [Dacier 2002]).

Group communication protocols – *aimed at combining classical fault tolerance techniques with design and implementation diversity.*

Using a group (or multiple groups) of functionally equivalent replicas is a classical method for achieving fault tolerant systems [Schneider 1990, Chérèque *et al.* 1992]. It is a method of redundancy that provides some protection against accidental (independent) replica failures as long as the correctly functioning membership of a group does not fall beneath the threshold necessary to ensure resilience to failure. This model is based on the assumption that faults occur independently of one another, affecting all replicas of a group with similar probability. For random and uncorrelated faults within a system, as well as those induced externally, but not maliciously, this assumption seems to be acceptable.

However, faults induced by the malicious acts of an attacker may not always match this assumption. This makes it problematical to use simple replication-based groups in adversarial environments. For example, if all replicas have a common vulnerability that permits an attacker to violate the integrity of the system as a whole, the effective working of the system can be easily compromised. The independence assumption applies here only to the extent that the effort required to break into each machine is the same. With sophisticated "hacking" techniques, this assumption becomes increasingly difficult to justify, especially in view of the daily-reported, large-scale, coordinated system attacks via the Internet.

MAFTIA has explored two different approaches to building intrusion-tolerant group communication protocols. The first approach [Cachin 2001] is to use a linear secret sharing scheme based on a *generalised adversary structure* that can model a more realistic set of fault assumptions. Replicas are classified according to one or more sets of attributes, and it is assumed that the characteristics of corrupting a replica vary according to these attributes. Suitable attributes include physical location, logical domain, system management personnel; type of operating system, protocol implementation, etc. Using an appropriately weighted linear secret sharing scheme, it is possible to construct protocols that can withstand the simultaneous corruption of all replicas in a given attribute class. For example, if the servers varied according to physical location and type of operating system, it would be possible to design a protocol that could tolerate the corruption of all the servers at a given site, and all the servers running a particular operating system.

The second approach [Neves & Veríssimo 2002] that MAFTIA has explored to constructing intrusion-tolerant group communication protocols is based on the use of a Trusted Timely Computing Base (TTCB). A TTCB is a trusted system component that can be used to provide timeliness and fail-silence properties in a hostile environment. Thus, the TTCB must be

implemented in a way that ensures its trustworthiness, perhaps by using a tamper proof hardware artefact with strict administrative control. Using a TTCB, it is possible to implement a reliable broadcast protocol that can tolerate up to *f* failures of *f+2* replicas.

Using intrusion-tolerant group communication protocols, it is possible to construct Trusted Third Party services (TTPs) such as *Certification Authorities*, *Fair Exchange*, *Notary*, *Authentication*, and *Authorisation* [Abghour *et al.* 2001, Cachin 2002].

Cryptographic techniques – *aimed at ensuring that data is securely transmitted over insecure physical communication channels, and that data is securely protected on storage devices – all of this, of course, underpinned by the safe management of cryptographic keys.*

These techniques play a crucial role in the provision of effective *identity and source of origin authorisation*, *safe cryptographic key management*, and *safeguards against misuse of data integrity and confidentiality of stored and communicated information*. They are, therefore, the basic defenses against intrusions that might damage information integrity and privacy.

MAFTIA's use of cryptography in this regard is essentially three-fold:

1) MAFTIA middleware provides reliable cryptographic communication protocols for single-cast message delivery; as well as reliable protocols for multi-cast, with causal and atomic properties for supporting single-valued and multi-valued Byzantine agreement.
2) Threshold cryptography is used to distribute knowledge of the secret keys used for digital signatures and decryption. Using a threshold scheme, a secret is shared between *n* parties in such a way that at least *t+1* of them must cooperate in order to decrypt messages or issue valid signatures.
3) The MAFTIA Authorisation Service provides cryptographic capabilities and vouchers for authorised access to system resources/objects, as described later.

Data fragmentation and scattering – *aimed at making data difficult to interpret as useful information.*

Fragmentation-scattering [Fray *et al.* 1986] is an intrusion tolerance method that can be likened to redundancy in classical fault-tolerance contexts, where redundancy is used to ensure that the occurrence of a fault in one replica copy will be of no consequence to the correct functioning and integrity of other replica copies. Fragmentation is the process of separating data in a way that renders each fragment of no interest to an attacker due to its lack of useful information. Scattering refers to the way in which each fragment is isolated from the others. When both of these techniques are successfully applied to the data held by a system, an attacker is potentially disabled from collecting any useful information. The number of intrusions that can be tolerated without revealing any significant information is dependent on the degree of fragmentation and scattering used. This degree is a system parameter that can be chosen with regard to acceptable trade-offs between security and performance.

Access control – *aimed at regulating access to resources/objects according to the principles of "least privilege" and "need to know".*

MAFTIA defines an intrusion-tolerant authorisation service that furnishes a scheme for granting permissions to each participant of a multiparty transaction, while distributing to each party only those permissions that are strictly needed to execute its own task [Abghour *et al.* 2001, Abghour *et al.* 2002]. This scheme is based on two levels of protection:

- A distributed authorisation server, in charge of granting or denying rights for operations involving one or more remote hosts. If such an operation is authorised, the authorisation server supplies all the necessary capabilities for the elementary operations needed to carry it out.

- A local *reference monitor* on each participating host, which is responsible for fine-grain authorisation, and is designed to enforce local access controls and restrict access to local resources by remote objects by intercepting remote method invocations and checking the capabilities that accompany each request. To ensure intrusion tolerance, critical parts of this security may be implemented using tamper-proof hardware on each participating host (e.g. a Java Card, which of course must be safely guarded by its authorised administrator).

The authorisation service is composed of replicated and diverse servers (operated by independent non-colluding system personnel), so that any single fault or intrusion can be tolerated without degrading the service. Confidential authorisation data (*access control matrix*) is fragmented, replicated and scattered across the servers. In order to reconstruct the data, multiple servers must co-operate. This means that as long as only a minority of the replicas are compromised, there is no loss of confidentiality of private data.

The local reference monitor, supported by a safe distributed signature algorithm, controls access to local application resources. If it is compromised then the effect of the failure is localised. For its safety and integrity, its essential key management facilities should be contained in tamper-proof hardware (e.g. a Java smartcard under the strict controls of its legitimate human administration.)

Transaction error confinement – *aimed at providing error confinement by encapsulating multiple actions within a transaction that provides atomicity, consistency, isolation and durability.*

MAFTIA defines an intrusion-tolerant transaction service that is implemented using replicated transaction managers and resource managers that communicate using Byzantine agreement protocols, built on top of the TTCB. The transaction service can be used for error confinement at the application level.

Chapter 3 Analysis of a MAFTIA application

The purpose of this chapter is to show how MAFTIA's intrusion tolerance capabilities can be deployed to protect a realistic application (TRADEZONE) from malicious attacks, thus demonstrating how the various mechanisms provided by MAFTIA can be used to achieve intrusion tolerance. In order to analyse MAFTIA's intrusion tolerance capabilities, fault trees are used to describe a representative set of hypothetical attacks, showing how an attacker (Malice) might attempt to intrude upon a MAFTIA-compliant TRADEZONE application. The fault trees illustrate the series of MAFTIA mechanisms that an attacker must successively overcome in order to achieve their objective, and the difficulty of achieving each step in the process is discussed as part of the fault tree analysis.

It is important to note that the analysis has only been performed at a conceptual or architectural level, and is by no means complete. It would not be feasible within the constraints of the MAFTIA project to build a real application using MAFTIA mechanisms, and perform a complete security analysis. However, the analysis is sufficient for its primary purpose, which is to illustrate the potential of MAFTIA's intrusion tolerance capabilities.

3.1 TRADEZONE application scenario

The overall application domain of TRADEZONE has the following characteristics:

- Electronic purchasing of goods and services that can either (a) be represented in catalogue form (i.e., definable product types and attributes), or (b) can be defined by a request for quotation / tender workflow.
- A domain with three principal actors (see Figure 1):
  - Purchasers – individuals and corporate groups of individuals requisitioning, approving, placing orders, tracking fulfilment and making payments.
  - Suppliers – creating and managing product information, customer account classes and pricing, processing and completing orders, responding to tenders.
  - TRADEZONE – serving as a broker between the purchaser and supplier of goods and their banks, together with other 3rd party marketplace sites that facilitate all ancillary services associated with e-commerce activities.

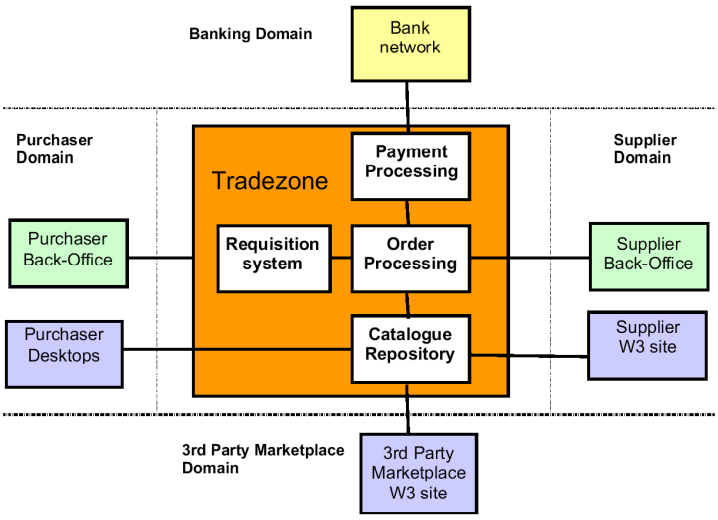

Figure 1: TRADEZONE application domains

Purchasers are able to source and order products through a custom browser window accessing their approved suppliers, and through a supplier's web site directly. There is a range of approval, management and reporting functions supporting the procurement process. Suppliers use a cataloguing system to create and maintain their product information, and define different account classes with specific pricing. They collect orders online and process through to fulfilment.

Market operators facilitate trading between partners in specific sectors (e.g., Internet portal businesses are typical customers). They attract suppliers and purchasers to the market, and can also import existing suppliers' catalogues where relevant. But the service as seen by an individual supplier or purchaser remains a one-to-one channel between themselves and each purchaser or supplier with whom they have a business relationship, i.e., it is not normally a public market where all suppliers are visible (although it can be).

The application scenario for this analysis examines the transaction protocols for placement of orders and supply of goods, within the context of a TRADEZONE system. The role of the TRADEZONE system is to provide a secure and trustworthy market place, and manage the sequence of steps involved in ordering, supplying, and paying for goods in a secure and trustworthy fashion, thus enabling suppliers to supply goods to purchasers, confident that they will receive payment in due course.

3.2 TRADEZONE security policy

As discussed in Chapter 3 of [Powell & Stroud 2003], a security policy can be thought of as comprising both goals and rules and we draw a distinction between these. Goals are intended to capture the high-level security requirements and as such any violation of the goals constitutes a security failure. In contrast, rules are typically lower level constraints on the system behaviour that are designed to ensure that the system is robust against (possibly malicious) faults. Violations of the rules will typically not correspond to security failures but correspond to erroneous states in which the system is more prone to failure.

3.2.1 Security goals

The high-level security goals of the TRADEZONE security policy are to provide a secure, and timely transaction service:

- Purchasers should be correctly charged for goods they receive.
- Suppliers should receive correct payment for goods supplied.
- Suppliers should dispatch goods in a timely fashion once an order has been accepted.
- Purchasers should confirm receipt of goods in a timely fashion.
- Banks should process payment requests and post confirmation signals in a timely fashion.
- The integrity and availability of supplier catalogues should be assured.
- All transactions should remain confidential.

3.2.2 Security rules

In order to achieve these security goals, the TRADEZONE application will be designed to enforce security rules such as the following:

- A registered purchaser should have read access to all the catalogues.
- A registered supplier should only have write access to their own catalogue.
- All communications should be authenticated and logged by TRADEZONE.

- All messages must be authenticated and encrypted and should follow the transaction protocol steps, outlined in section 3.3 below.

3.2.3 Security failures

A security failure will occur if one of the security goals of the TRADEZONE application is violated. For example:

- A supplier receives payment for which there is no corresponding delivery of goods.
- Goods are delivered but the supplier does not receive payment.

Such failures could be due to flaws in the formulation or implementation of the security rules, allowing someone to fake order and receipt messages for example. Alternatively, the failure of an authentication mechanism might allow fake messages to be introduced into the system. Finally, failures could be due to a fault in the architectural assumptions. For example, the implementation of trusted channels might be flawed, allowing fake payment messages to be inserted into the system, or the authorisation mechanisms might be bypassable.

In the analysis that follows, we will consider the various ways in which the MAFTIA mechanisms protect against such failures.

3.3 TRADEZONE transaction protocol

For the purposes of our analysis, we will ignore the registration and catalogue management issues and simply deal with the correct unfolding of the TRADEZONE transactional protocol between a purchaser and a supplier.

A registered purchaser (P) logs on, and once authenticated, is granted read access to the TRADEZONE (TZ) catalogue. She can now place an order for the goods. On receipt of an authorised order, TZ relays it to the appropriate supplier (S). S may now choose either to accept or reject the order. We will assume that S accepts the order and sends an "accept" message back to TZ, which TZ then logs and relays back to P. S then sends the goods. On receipt of the correct goods, P sends a receipt message to TZ, which then issues a payment order to P's bank (PB). P's bank arranges the transfer of funds to S's bank (SB) and sends P a confirmation message. SB also sends a message to S confirming that the payment has been received, thus completing the process.

Logically, the protocol involves the following steps:

1) P→TZ: purchaseGoods (P, S, GoodsId: x)
2) TZ → S: orderGoods (P, S, GoodsId: x)
3) S → TZ: acceptOrder (P, S, GoodsId: x) or reject
4) TZ → P: ack (P, S, GoodsId: x)
5) [S → P: deliver item x]
6) P → TZ: sendReceipt (P, S, GoodsId: x)
7) TZ → PB: makePayment (P, S, GoodsId:x)
8) [PB → SB: payment for x]
9) PB → P: confirmPayment (P, S, GoodsId:x)
10) SB → S: confirmPayment (P, S, GoodsId:x)

The TRADEZONE application would be responsible for ensuring that these steps were performed reliably and securely in the correct sequence. Thus, a MAFTIA version of the TRADEZONE application would make use of the services provided by MAFTIA

mechanisms such as the Transaction Manager and the Authorisation Server to provide this guarantee, but these additional interactions are not shown here. For example, each of the above steps would be allowed for the enacting parties if, and only if, the MAFTIA authorisation server had issued them with the necessary permissions. Similarly, interactions with the MAFTIA transaction manager would be used to ensure that these steps were performed reliably and atomically.

3.4 TRADEZONE in a MAFTIA setting

Figure 2 illustrates a possible implementation of the TRADEZONE application built using MAFTIA architectural principles. Although not shown, it is assumed that TRADEZONE and the banks communicate via secure channels.

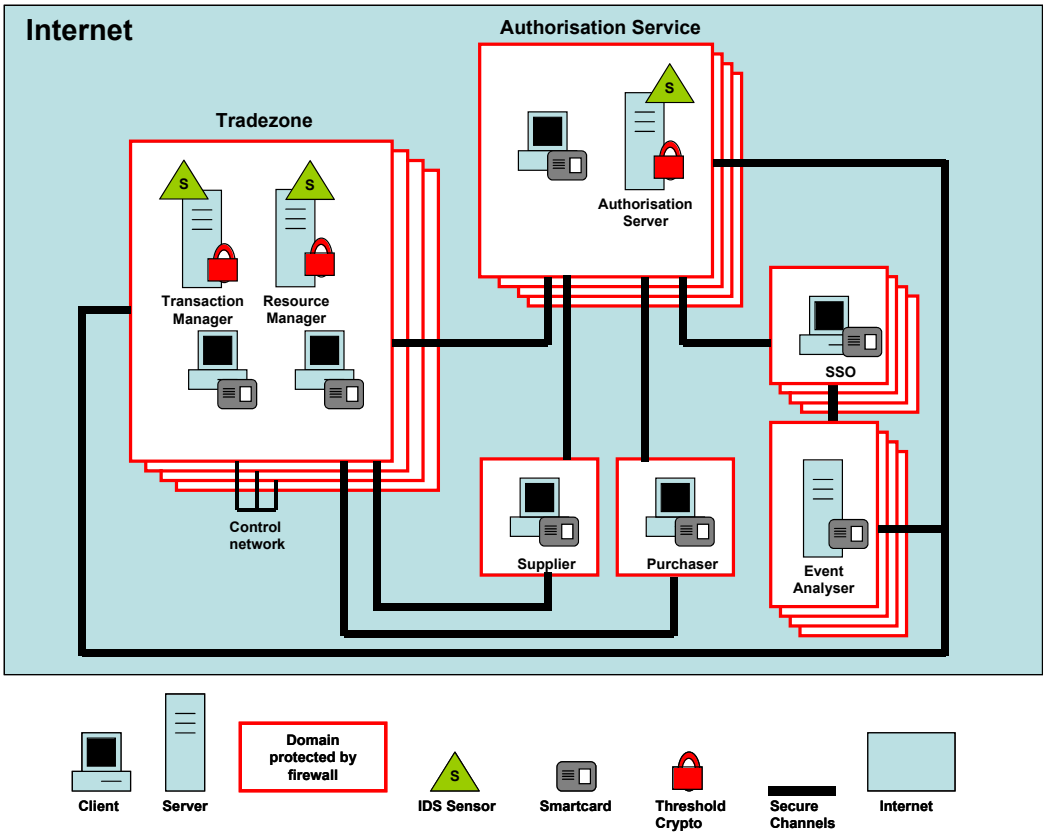

Figure 2: TRADEZONE in a MAFTIA setting

This setting represents both a logical and physical system configuration. The logical configuration identifies the interactions between components. The physical configuration shows the relationship between single site purchasers and suppliers (here only two are shown), the multi-site Authorization Service (AS), the multi-site Intrusion Detection System (IDS) and System Security Officers (SSOs), and the TRADEZONE service with its transaction and resource managers. The secure channels between the systems form a Virtual Private Network (VPN), implemented using a well-known standard such as IPSec. However, MAFTIA mechanisms are used to enhance the security capabilities of the services provided over the Internet. All multi-site services use replication and fault masking to achieve intrusion tolerance. The Authorization service uses probabilistic asynchronous Byzantine Agreement protocols, and depends on design diversity to achieve failure independence. The TRADEZONE services are replicated, using the TTCB to construct a trusted channel between the servers that is tamper-proof by design.

In order to be enabled for secure operation, each distributed site (host) must be associated with a MAFTIA issued smartcard containing the security keys needed for secure communication and authorisation.

3.5 Attack scenarios

3.5.1 Introduction

The following sections present a representative, although by no means exhaustive, set of informal attack scenarios that are intended to illustrate the difficulties that face an attacker (Malice) trying to effect successful intrusions upon the TRADEZONE application.

The scenarios apply the technique of fault forecasting to demonstrate the role of MAFTIA in the provision of a more secure system than a traditional IPSec approach. Note that a standard IPSec approach only provides confidentiality of communications, and thus IPSec forms a subset of MAFTIA's overall approach to achieving intrusion tolerance.

We use the notation of fault trees to describe the various attack scenarios. Fault trees are used in hazard analysis where each hazard is a situation in which there is actual or potential danger to people or to the environment [Storey 1996]. Fault tree analysis starts with an event directly related to an identified hazard and works backwards to determine possible causes. Each event in the tree may have child events that are combined together with logical operations such as AND and OR. Fault trees may also be linked to each other, where the top-level event of one tree may be a bottom-level event of another tree.

The security engineering community use a similar technique called *attack trees*. In an attack tree, the events are goals and the leaf nodes are ways of achieving the goals [Schneier 1996]. The tree can be annotated with measures of risk, and the overall risk of the root event occurring can be calculated by traversing the tree and combining risks according to functions associated with the logical operations. For example, assuming independent probabilities can be assigned to leaf nodes, the root node probability can be calculated by adding probabilities connected by OR operators, and multiplying together probabilities connected by AND operators. In our approach, however, we present a less rigorous intuitive measure of these risks, expressed in terms of value judgements about the effectiveness of MAFTIA's intrusion tolerant capabilities.

An overview of the notation used in our analysis is shown in the figure below.

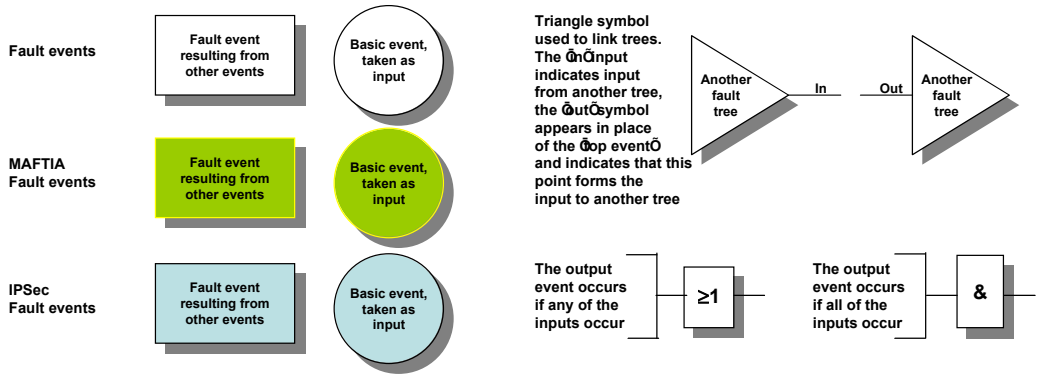

Figure 3: Graphical notation used for fault trees

Note that we distinguish MAFTIA fault events from IPSec fault events and others. To this end, we use different shadings. And in this regard, MAFTIA fault events correspond to failures of MAFTIA's intrusion tolerance capabilities.

As part of our analysis of each event, we discuss some of the assumptions that must hold for the correct functioning of MAFTIA-compliant system services.

3.5.2 Overview

The overview shown in Figure 4 depicts the top-level fault tree for our attack scenarios. We assume that the attacker, Malice, has successfully registered as a legitimate supplier with TRADEZONE. Malice is thus a corrupt insider. The root event in our fault tree occurs if Malice succeeds in illegally making money whilst avoiding detection. In order to make money illegally, Malice must either steal money from her competitors directly, or else gain an unfair advantage over them by manipulating the market within TRADEZONE in some way (for example, Malice could masquerade as a purchaser and reserve all her competitors' stock, so that she was the only supplier able to fulfil new orders).

Each of the leaf events shown in Figure 4 has its own attack/fault tree, which is described in more detail in a subsequent section.

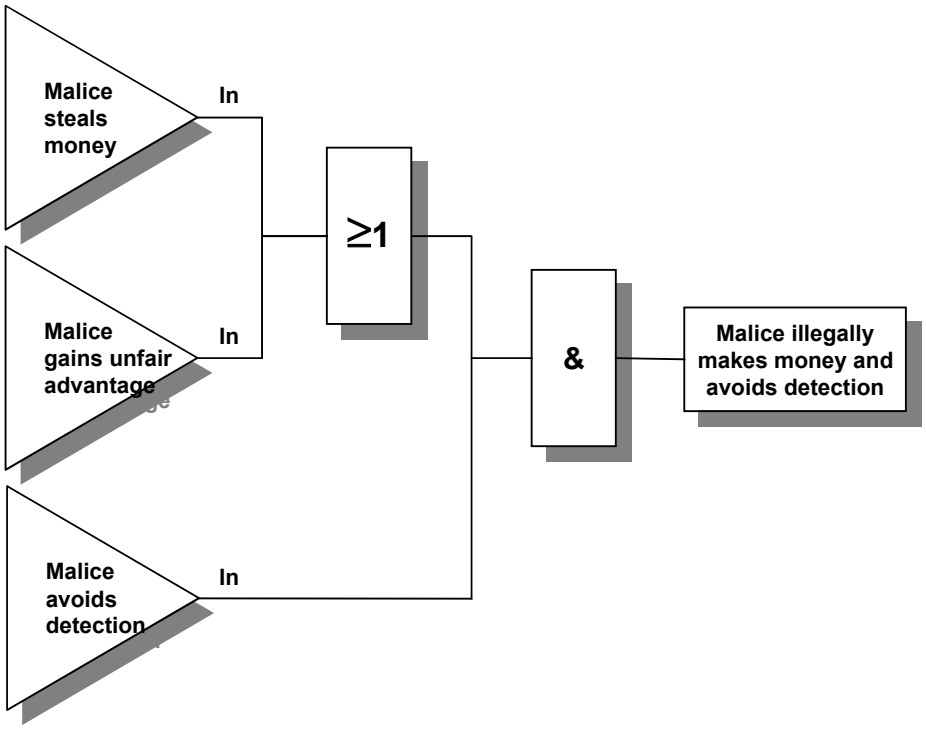

Figure 4: High level fault tree

3.5.3 Malice steals money

We consider three ways in which Malice could steal money from purchasers:

- *She receives money from a purchaser to which she is not entitled*

  The TRADEZONE application is supposed to ensure that the steps in the transaction protocol are followed in the correct sequence. Thus, TRADEZONE will only issue payment instructions for goods on completion of an order. To receive payment illegally, Malice must forge a **sendReceipt** request from a purchaser for goods that she has not delivered. The purchaser must either have legitimately ordered the goods from Malice, or else Malice must have forged the order as well. In order to forge a request, Malice must subvert the authorisation process.

- *She receives money meant for another supplier*

  To achieve this she must intercept and modify a legitimate **purchaseGoods** request from a purchaser, so that it names her as the recipient, issue this request to TRADEZONE, and then issue a **sendReceipt** request on behalf of the purchaser, causing payment to her.

  This is like the previous attack, but also requires Malice to break a secure channel between the legitimate supplier and TRADEZONE.

- *She subverts the TRADEZONE application entirely:*

  In which case, she has complete control over the system, and can issue illegal payments to herself. However, in order to subvert TRADEZONE, she must successfully attack the replicated servers within TRADEZONE itself, which are linked by a trusted network, assumed to be tamper-proof by design.

Note that we exclude the possibility that Malice can interact with the bank directly, because we assume that the channels between TRADEZONE and the banks are physically secure and not accessible to outsiders. (Note that Malice is an insider with respect to the TRADEZONE application domain, but an outsider with respect to the TRADEZONE administrative domain).

The attack scenarios identified above are outlined in Figure 5.

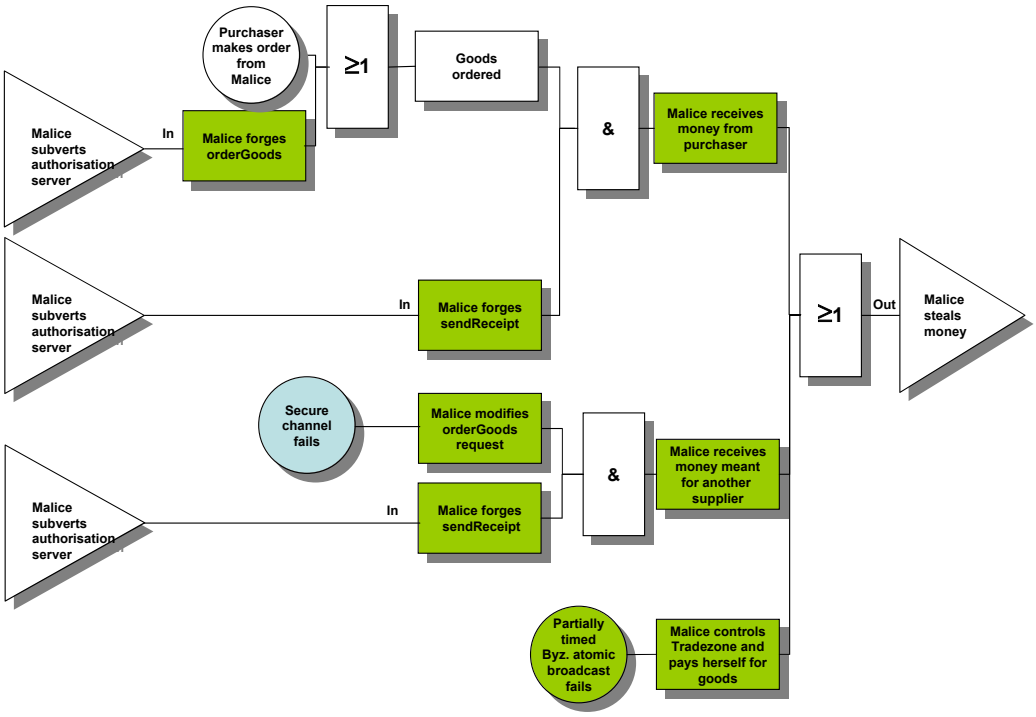

Figure 5: Malice steals money

In the first two cases, Malice must subvert the authorisation process and forge a **purchaseGoods** or **sendReceipt** operation with a correct but illegitimate proof of authorisation.

In the third case, she must defeat the TRADEZONE transaction manager service, which is implemented using TTCB-based Byzantine atomic broadcast, i.e. depends on the use of a trusted network built from tamper-proof components. This would require her to subvert a trusted hardware component on at least *f+1* out of *f+2* servers to which she does not have physical access.

3.5.4 Malice subverts authorisation process

If Malice is unable to overcome the TTCB-based protection of the TRADEZONE application, then she must subvert the MAFTIA authorisation service. This requires her to form acceptable operational requests that appear to come from target sites (in this case the purchaser's sites) by inserting or modifying high-level operation requests (with valid vouchers), and thereby having them honoured by the TRADEZONE reference monitor.

The cases considered are as follows.

- *Authorisation service compromised*
  - Malice modifies the authorisation service's access control matrix.
  - Malice forges illegal capabilities.
- *Authorised user impersonated*
  - Malice steals the purchaser's smartcard and obtains its PIN.
  - Malice forges a user's smartcard.

These cases are illustrated below, and discussed thereafter. Note that we do not consider the possibility that the authorisation mechanisms on the local host could be bypassed.[1] This is reasonable because the authorisation scheme does not rely on the trustworthiness of other hosts, and thus, even if a particular host is corrupted, it cannot persuade a non-corrupted host to execute unauthorised operations on its behalf. (See [Abghour *et al.* 2002] for more details.)

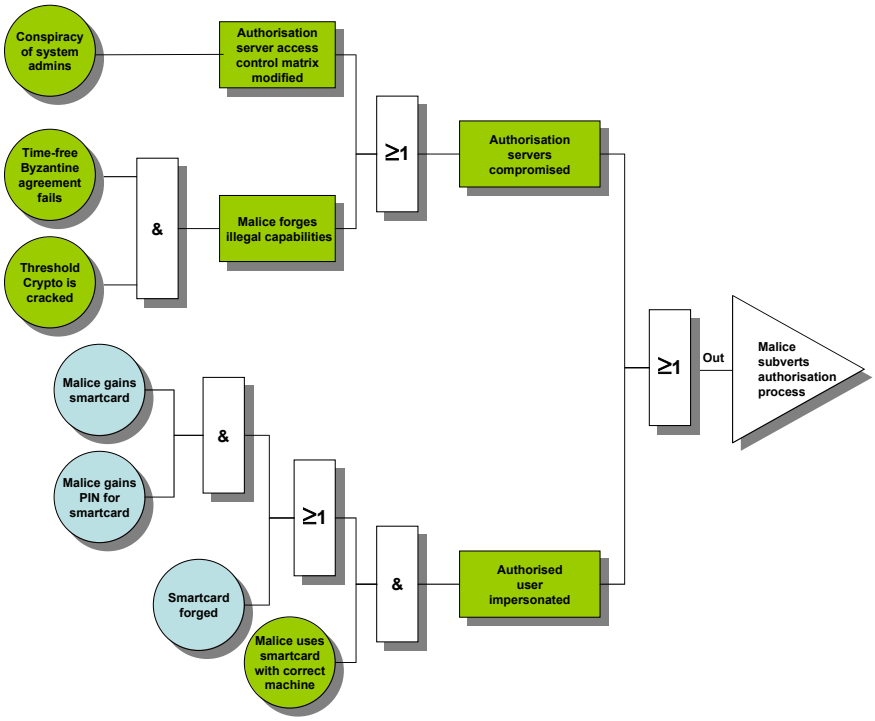

Figure 6: Malice subverts authorisation process

[1] Note that if the TRADEZONE server is considered to be a single host, an insider might be able to bypass the local reference monitor within TRADEZONE, and thus be able to run any unauthorized operation locally (e.g., fake purchase). However, this would be much more difficult if the TRADEZONE server was composed of dedicated hosts (e.g., catalogue, purchase request, invoice, delivery, etc.). In that case, even if one host is subverted, it would still not be possible (practically) for the attacker to realise a complete fake transaction.

Compromising the authorisation service so that Malice is able to obtain permissions to which she is not entitled can only be achieved if she either changes the access control matrix, or she subverts the asynchronous, time-free Byzantine agreement protocol used by the authorisation service, together with the threshold cryptographic protocol used to protect its secret key and thus its digital signatures.

Changing the access control matrix could be achieved if either there is a conspiracy of sufficient system administrators working in Malice's interest, or Malice can crack the threshold cryptography scheme used to scatter and encrypt the contents of the matrix. Administrator conspiracies are a social engineering problem that falls outside the technical jurisdiction of MAFTA, and so we do not consider this type of attack further.

Breaking the matrix scatter and encryption scheme requires Malice to control and understand the programmed semantics of $t+1$ hosting sites, where t is the intrusion resilience threshold above which the intrusion tolerance of the access matrix fails. Malice would also need to overcome the distributed diversity of the hosting sites. All of this taken together is regarded as a non-trivial problem with sufficient work factors to deter the attacker.

For Malice to obtain (forged) capabilities and vouchers, she needs to break the asynchronous Byzantine protocol of the authorisation service and crack the threshold cryptographic scheme for deriving the necessary secret key to sign capabilities and vouchers. The resilience of the authorisation service as a whole to attack depends on the underlying adversary structure, which assumes that servers are grouped together according to one or more sets of attribute values. The protocols can tolerate the failure of an entire set of servers within each attribute class (see [Cachin 2001] for more details). Again, host site diversity can also be brought into play to frustrate Malice.

As an alternative to attacking the authorisation server, Malice could instead attempt to impersonate another authorised user by stealing or forging that user's smartcard. The authorisation server can provide no guarantees about the behaviour of a compromised user, but it can prevent the compromised user from interfering with the activities of legitimate users.

The security of a smartcard and its PIN is the responsibility of the authorised cardholder. Loss of the card and its PIN cannot be protected by MAFTIA's technical defenses alone. The cardholder has two essential responsibilities in this respect: (i) to always keep the card in a safe place; and (ii) to never reveal its PIN. If, however, the card is stolen, then the MAFTIA-compliant application (in this case, TRADEZONE) should require the cardholder to report its loss to the security administration as soon as possible so that the certificate for the card (held on the card itself and known by the authorisation service) can be revoked. Additionally, the PIN that enables the card to be authenticated to a local reference monitor must never be allowed to be given incorrectly more than, say, three times without being revoked. Such policies accord with standard ATM banking practice. Such a security policy will of course require cooperation between the logic on the card, the reference monitor on a host, the security administration, and the authentication service.

The act of forging a smartcard is considerably more difficult than stealing an existing card. Although technical attacks on smart cards are known to exist (see for example, Chapter 14 of [Anderson 2001]), they are assumed to be beyond the resources of Malice. In any case, even if Malice were to be able to break into a smart card and steal the private keys it contained, she would still not be able to generate false permissions to access other MAFTIA hosts because this would require knowledge of the private key for the authentication server, which is not stored on the smart card [Abghour *et al.* 2002]. Rather than attempting to forge a smart card for an existing user, Malice would sensibly choose the easier option of stealing the card and deriving its PIN. However, in either case, before Malice was able to use the forged or stolen card, she would also have to defeat the certificate-based host authorisation mechanism used by the Java card, or gain physical access to her victim's machine, which is another obstacle for her to overcome.

3.5.5 Malice gains an unfair advantage

Instead of stealing money directly from her competitors, Malice could attempt to manipulate the TRADEZONE marketplace so as to ensure that her competitors cannot offer the same goods at a cheaper price. There are a number of possible ways that Malice (or an associate) might carry out such an attack, as illustrated below.

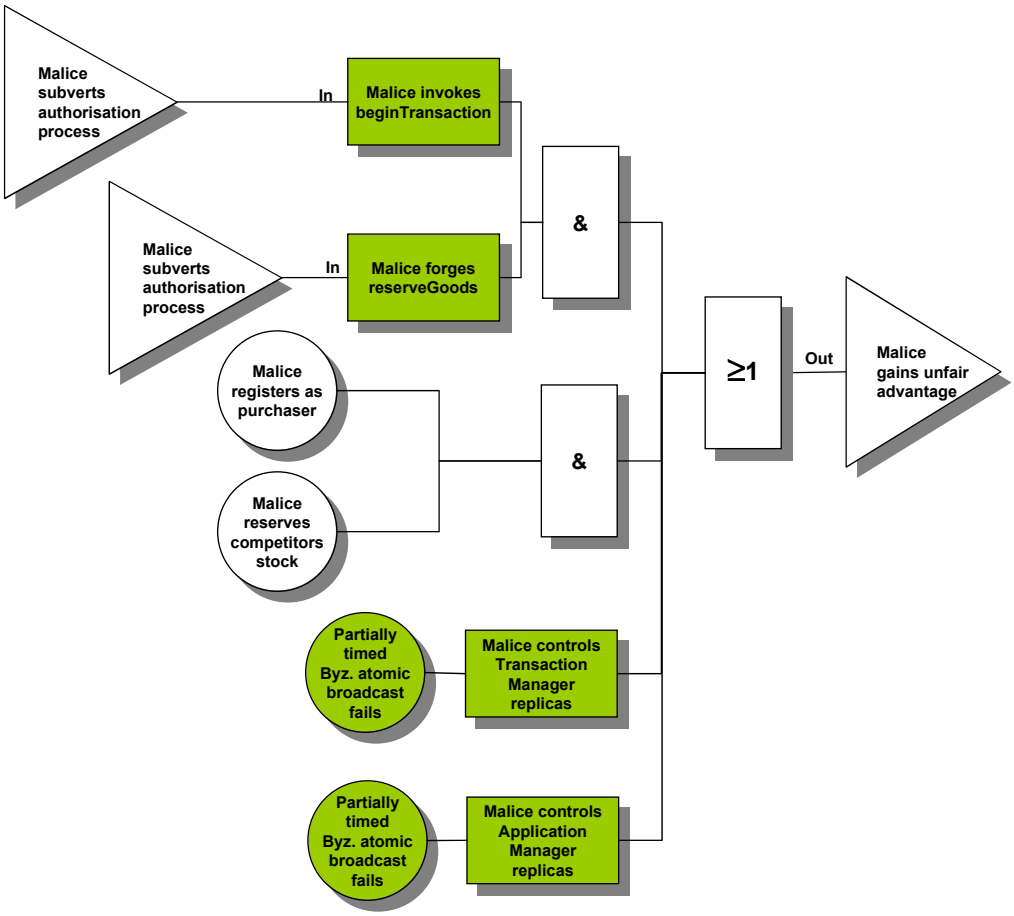

Figure 7: Malice gains unfair advantage

For example, Malice might attempt to subvert the TRADEZONE protocol by acting as a purchaser (although she is not registered as one) and both starting a transaction and, within its context, attempting to reserve all the available stock of her rivals. To do this she must defeat MAFTIA's authorisation service, as outlined earlier. In practice, a real TRADEZONE application should prevent a single purchaser from reserving all goods of a particular type offered by a supplier. However, Malice might attempt to work in concert with others in making this attack.

Alternatively, Malice might legitimately register herself as a purchaser with TRADEZONE, perhaps using a forged identity, and then quite legitimately make the same attack. Interestingly, there is nothing in the TRADEZONE security policy stated above in Section 3.2 to say that it is illegal for a supplier to also be a purchaser, and indeed, it might not be appropriate for TRADEZONE to impose such a restriction. However, without a clear separation of roles within TRADEZONE, this kind of abuse is possible, and cannot be prevented by solely technical means.

Malice might also try to bring sufficient transaction manager replicas under her control such that the resource managers lock the goods and thereby prevent them from being offered by

suppliers. As the transaction manager service is implemented using TTCB-based Byzantine atomic broadcast, the correct operation of the transaction service depends upon the integrity of the local TTCBs and their control network, which are designed to function correctly in a hostile environment and assumed to be tamper-proof by design. Similarly, she might attempt to bring sufficient resource mangers under her control and cause them to inform clients that the goods required are unavailable for purchase, but again this would require her to defeat the TTCB-based Byzantine atomic broadcast, and the same obstacles would apply

Other possibilities for this kind of attack, not considered here, would include attacks on the integrity of catalogues so as to manipulate the prices. This would involve either compromising the authorisation scheme or defeating the data fragmentation technique used by TRADEZONE to ensure the integrity of catalogues.

3.5.6 Malice evades detection

Malice would aim to avoid detection by the Intrusion Detection System (IDS) at least while she was carrying out her activities. This could be achieved either by corrupting the event analysers and thus preventing the IDS from correctly detecting her attack, or else by ensuring that events associated with her activities were not collected in the first place.

An example attack tree follows. Note that we only consider technical attacks on the IDS, and do not consider the possibility that Malice could successfully conceal her activities by masquerading as a legitimate user.

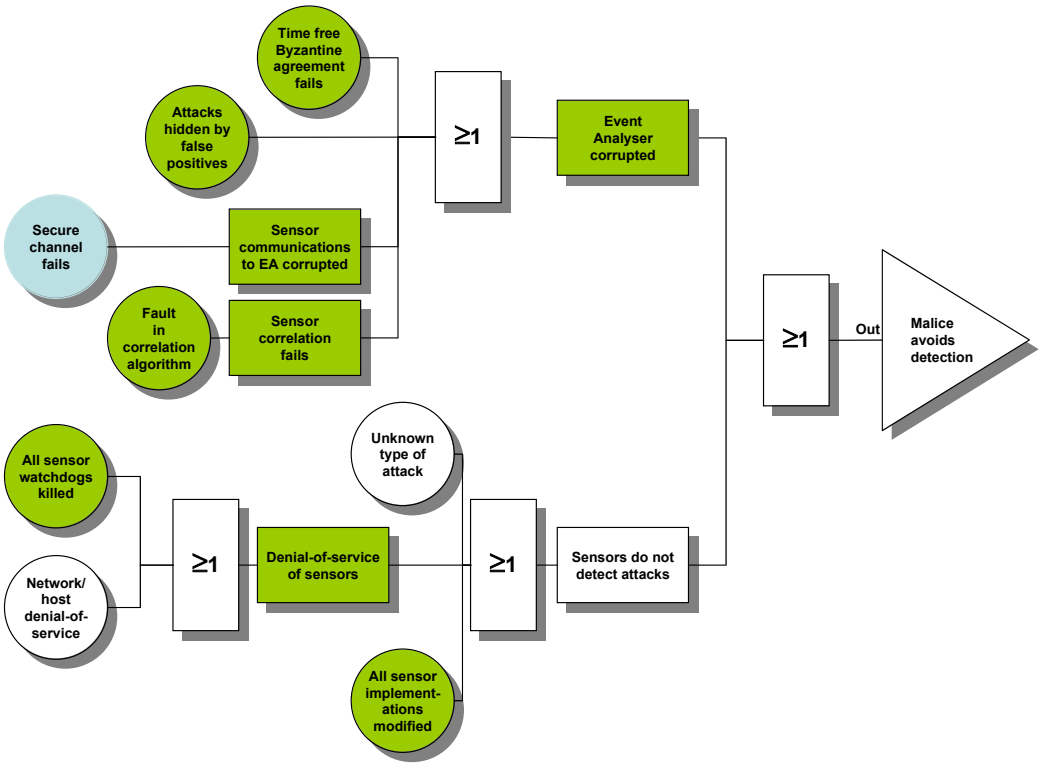

Figure 8: Malice evades detection

There are several ways in which Malice could prevent the correct correlation of events. She could either subvert the time free Byzantine agreement protocols used to ensure robustness against attack, hide events related to her attacks amongst false positives, corrupt communications with the sensors that provide the events for analysis, or allow events to be received but ensure that they are not correlated correctly by corrupting the correlation algorithm.

The steps for subverting agreement protocols were discussed earlier. Hiding attacks amongst false positives would require the data mining algorithms used by the IDS to eliminate false positives to be subverted so that events related to an attack were removed from the analysis. One approach might be to install faulty equipment that generates the same events as used by Malice for her attack, and then use the same machines to perform the attack. Subverting the secure channels between sensors and the event analyser requires subverting standard IPSec protection, which we do not consider further. Ensuring that events are not correlated correctly could result from simple design faults or some form of social engineering attack that we do not consider within the scope of the MAFTIA project.

Ensuring that sensors do not detect attacks could mean that Malice has chosen an attack that does not register with the sensors, there is a denial-of-service attack on the sensors, or all sensor implementations are modified. A denial-of-service attack can be achieved should Malice gain sufficient privilege on the host where the sensor is resident, and she kills both the sensor and the watchdog process that is intended to restart it after it is killed[2]. Alternatively Malice could mount a denial-of-service by flooding the network or physically disconnecting the sensors. However, denial of service attacks and sensor modification is made more difficult by sensor replication, distribution and host site diversity.

[2] In general, we would expect it to be more difficult for an outsider to kill the watchdog process because, unlike the sensor, the watchdog does not provide an interface that is accessible over the network.

Chapter 4 Discussion

As show by the analysis in the previous chapter, MAFTIA provides a variety of mechanisms that can be used in combination to construct intrusion-tolerant systems. An intrusion-tolerant system must be able to continue to deliver a secure service, despite the presence of intrusions, and thus, a "defence in depth" strategy is needed so as to avoid depending on any particular component of the system that could become a single point of failure. In this chapter, we highlight some of the key concepts and ideas underpinning MAFTIA's approach towards achieving intrusion tolerance, as illustrated by the scenarios discussed in the previous chapter.

4.1 Preventing errors from leading to security failures

MAFTIA Deliverable D21 [Powell & Stroud 2003] introduces the classical dependability concepts of fault, error, and failure, and shows how they may be applied to reasoning about prevention and tolerance mechanisms aimed at ensuring system security. An error is the manifestation of a fault on the system state, and a failure is the manifestation of an error on the service delivered to the system user. An error can have multiple causes. In particular, an error that could result in a security failure is not necessarily the result of a malicious fault – such an error could also result from an accidental fault. This is the reason that MAFTIA defines intrusion detection as the detection of errors that can result in security failure rather than the detection of errors that result from intrusions. However, regardless of the cause, the system is still liable to failure unless some step is taken to remove the error from the system.

MAFTIA's approach to error handling is to use error compensation techniques based on active replication and masking. Thus, Byzantine agreement protocols are at the heart of the intrusion tolerance mechanisms provided by MAFTIA. However, even though Byzantine agreement protocols are designed to deal with arbitrary faults in the value and time domain, they still assume failure independence. In the presence of malicious faults and deliberate attacks on systems, this is not a reasonable assumption, and MAFTIA must therefore take steps to address this problem by designing protocols that can tolerate more realistic failure assumptions, and by ensuring that compromising one replica does not make it any easier to compromise another. The standard way of achieving failure independence is to make some assumption about diversity, and thus, various forms of diversity are an important part of any intrusion tolerance strategy. As an additional intrusion tolerance strategy, MAFTIA also seeks to minimise the extent to which different parts of the system are trusted, so as to make it harder for the effects of a successful intrusion to propagate throughout the system. This is an example of error confinement.

It is worth discussing briefly two other possible methods of error handling, namely forward and backward error recovery. In principle, these techniques could also be used for intrusion tolerance, but in practice, this is not straightforward for at least two reasons. Firstly, both methods depend on a reliable error detection mechanism, but despite the progress made by MAFTIA in this area, building a reliable IDS is still an open research area. However, without the ability to detect errors correctly with a high probability, it is not possible to build a reliable fault tolerance mechanism based on an unreliable error detection mechanism. Secondly, any state-based recovery mechanism must deal with the problem of latency of error detection and the possibility that the intruder could have concealed their activities for a long time, or spread them throughout the system. Once an intrusion is discovered, it can be a very difficult and time-consuming process to construct (or re-construct) a system state that is free from the effects of the intrusion. Thus, with the current state of the art in intrusion detection, it would seem that error handling approaches based on error compensation are the only viable way of building intrusion tolerant systems at present.

4.2 Using active replication to achieve error compensation

MAFTIA has explored two different approaches to implementing error compensation mechanisms using active replication. Both depend on the use of Byzantine Agreement protocols that are capable of tolerating malicious faults and provide some degree of failure independence. However, the failure assumptions are different.

The first approach is based on the use of hybrid failure assumptions, which support the construction of systems from a mixture of trusted and untrusted components. However, components are only trusted to the extent of their trustworthiness. Thus, steps must be taken to increase the trustworthiness of each trusted component. MAFTIA has developed a set of partially timed Byzantine Agreement protocols based on the use of a trusted component called the TTCB. The TTCB is a distributed security kernel that uses a dedicated control network to provide strong guarantees about timeliness and failure. The TTCB component installed on each MAFTIA host is assumed to be tamperproof and fail-silent by design. One approach to achieving this is to use physical isolation techniques and special hardware. Thus, the TTCB is able to support the correct execution of Byzantine agreement protocols, even in the presence of malicious faults. Furthermore, because it is assumed that a TTCB cannot be compromised, it is possible for a TTCB group to support a reliable broadcast protocol with $f+2$ replicas, requiring an attacker to compromise $f+1$ TTCB servers in order to effect an intrusion. Thus, for the same number of replicas, a TTCB-based system can tolerate a larger number of intrusions than a more traditional approach (which normally require $3f+1$ replicas to tolerate f failures). However, it is still necessary to ensure failure independence using diversity techniques, otherwise, an attacker might be able to exploit a common vulnerability and intrude upon a set of replicas with reduced effort.

The TTCB approach towards constructing intrusion-tolerant Byzantine Agreement protocols is based on the recursive application of fault prevention and fault tolerance techniques. Fault prevention techniques are used to build a trustworthy TTCB, which can then be used to support the execution of fault-tolerant protocols. In contrast, the other approach that MAFTIA has developed for implementing Byzantine Agreement protocols does not make any assumptions about the trustworthiness of individual components or hosts. There are no fail silence assumptions or timeliness guarantees. Instead, cryptographic algorithms are used to implement an efficient, probabilistic, Byzantine Agreement protocol in an asynchronous setting. These protocols assume a generalised adversary structure, in which replicas are classified according to one or more independent sets of attributes. Assuming that there are at least four different values for each attribute, it is possible to design protocols that can tolerate the simultaneous failure of a complete set of replicas in each attribute class. Again, it is necessary to make assumptions about failure independence, but the protocols are designed to tolerate a limited number of common failures with a more realistic set of failure assumptions. In order to successfully attack these protocols, an attacker must either break the diversity assumption and compromise a sufficient number of hosts, or else break the underlying cryptographic algorithms on which the protocols are based. See [Cachin 2001] for a precise formulation of the failure model.

4.3 Approaches towards achieving error confinement

An important part of realising a “defence in depth” strategy is to accept that components of the system can be compromised, and minimise the extent to which they are trusted by other components of the system. Thus, error confinement strategies are an important part of achieving intrusion tolerance.

There are many examples of such strategies within MAFTIA. For example, Threshold Cryptography schemes are used extensively for secret sharing. A threshold scheme shares a secret between n parties in such a way that at least $t+1$ shares are required in order to use the secret, for example, to generate a digital signature or to decrypt a message. Thus, no party is

trusted with the entire secret or is able to reconstruct the secret, and consequently, the system is able to tolerate a limited number of intrusions, up to the threshold parameter *t*.

Similarly, the MAFTIA Authorisation Scheme is designed for Internet applications, in which hosts are mutually suspicious of each other. Although Java Cards are used to implement a local security kernel on each host, and it is assumed that it would be very difficult for an attacker to access or forge the cryptographic keys stored on the Java Card, even if the attacker were to succeed in compromising the Java Card in this way, it would still not be possible for the intrusion to propagate beyond the local host. This is because a faulty host cannot generate false permissions for other MAFTIA hosts without knowing the private key of the Authorisation Server, which is protected by a Threshold Signature scheme.

As a final example, there are various ways in which the alarm signals generated by the sensors deployed by an Intrusion Detection System (IDS) could be compromised or manipulated by an attacker, and an intrusion tolerant IDS needs to protect against these kinds of attack. [Dacier 2002] discusses a number of strategies for building an intrusion tolerant IDS, some of which could be viewed as error confinement mechanisms that seek to reduce the amount of trust that an IDS puts in any particular sensor. For example, event correlation techniques or the use of diverse IDS implementations that respond to a given attack in different ways can be regarded intrusion tolerance strategies that exploit diversity in order to build a more trustworthy distributed IDS out of IDSs that are individually untrustworthy.

Chapter 5 Conclusions and future research

This report has presented a summary of MAFTIA's approach towards building intrusion tolerant systems, and demonstrated its effectiveness by constructing an architectural model of a small but realistic use case based on an e-commerce application. Fault trees have been used to analyse a representative but by no means exhaustive set of attacks on this model, and thus highlight the obstacles that an attacker must overcome in order to defeat MAFTIA's intrusion tolerance mechanisms and intrude upon the application.

The analysis was designed to illustrate and highlight a number of key design principles for building intrusion tolerant systems. These include: the use of a "defence in depth" strategy, the notion of trusting components to the extent of their trustworthiness but no more, measures to increase trust such as the use of Byzantine agreement protocols, and measures to decrease the reliance on trust such as the use of error confinement mechanisms to prevent intrusions from propagating throughout a system. Detailed discussions of all of these models and techniques can be found in the relevant MAFTIA deliverables.

Because of the difficulties of building reliable error detection mechanisms in a hostile environment and the problems posed for state based error recovery methods by long latency of error detection and malicious dormant faults, MAFTIA has concentrated on achieving intrusion tolerance using error compensation mechanisms based on masking. However, the success of these measures depends on failure independence assumptions, which are typically based upon claims about diversity. Thus, three important areas for future research are: developing techniques for ensuring and measuring diversity in the presence of arbitrary malicious faults, improving the quality of error detection mechanisms so as to make state-based error recovery techniques feasible as a means of intrusion tolerance, and finally, finding solutions to the problems of long latency of error detection and malicious dormant faults.

References

[Abghour *et al.* 2001] N. Abghour, Y. Deswarte, V. Nicomette and D. Powell, *Specification of Authorisation Services*, MAFTIA Project, Deliverable D27, January 2001.

[Abghour *et al.* 2002] N. Abghour, Y. Deswarte, V. Nicomette and D. Powell, *Design of the Local Reference Monitor*, MAFTIA Project, Deliverable D6, April 2002.

[Adelsbach & Creese 2003] A. Adelsbach and S. Creese (Eds.), *Final Report on Verification and Assessment,* MAFTIA Project, Deliverable D22, 2003.

[Anderson 2001] R. J. Anderson, *Security Engineering: a Guide to Building Dependable Distributed Systems,* Wiley Computer Publishing, 2001.

[Anderson & Lee 1981] T. A. Anderson and P. A. Lee, *Fault Tolerance — Principles and Practice,* Prentice-Hall, 1981.

[Avizienis 1967] A. Avizienis, "Design of Fault-Tolerant Computers", in *Fall Joint Computer Conference,* AFIPS Conf. Proc., 31, pp.733-43, Washington D.C.: Thompson Books, 1967.

[Avizienis *et al.* 2001] A. Avizienis, J.-C. Laprie and B. Randell, *Fundamental Concepts of Dependability*, Research Report N°01145 (Revision 1: December 2002), LAAS-CNRS, August 2001 (UCLA CSD Report no. 010028; Newcastle University Report no. CS-TR-739).

[Cachin 2001] C. Cachin (Ed.), *Specification of Dependable Trusted Third Parties,* MAFTIA Project, Deliverable 26, 2001.

[Cachin 2002] C. Cachin (Ed.), *Full Design of Dependable Third Party Services,* MAFTIA Project, Deliverable D5, 2002.

[Chérèque *et al.* 1992] M. Chérèque, D. Powell, P. Reynier, J.-L. Richier and J. Voiron, "Active Replication in Delta-4", in *22nd IEEE Int. Symp. on Fault-Tolerant Computing Systems (FTCS-22),* (Boston, MA, USA), pp.28-37, IEEE CS Press, 1992.

[Dacier 2002] M. Dacier (Ed.), *Design of an Intrusion-Tolerant Intrusion Detection System,* MAFTIA Project, Deliverable D10, 2002.

[Deswarte *et al.* 1991] Y. Deswarte, L. Blain and J.-C. Fabre, "Intrusion Tolerance in Distributed Systems", in *Symp. on Research in Security and Privacy,* (Oakland, CA, USA), pp.110-21, IEEE CS Press, 1991.

[Fray *et al.* 1986] J.-M. Fray, Y. Deswarte and D. Powell, "Intrusion-Tolerance using Fine-Grain Fragmentation-Scattering", in *Symp. on Security and Privacy,* (Oakland, CA, USA), pp.194-201, IEEE CS Press, 1986.

[MAFTIA 2000] MAFTIA, *Reference Model and Use Cases*, Deliverable D1, MAFTIA Project, 2000.

[Neves & Veríssimo 2002] N. F. Neves and P. Veríssimo (Eds.), *Complete Specification of APIs and Protocols for the MAFTIA Middleware,* MAFTIA Project, Deliverable D9, 2002.

[Powell *et al.* 1988] D. Powell, G. Bonn, D. Seaton, P. Veríssimo and F. Waeselynck, "The Delta-4 Approach to Dependability in Open Distributed Computing Systems", in *18th IEEE Int. Symp. on Fault-Tolerant Computing Systems (FTCS-18),* (Tokyo, Japan), pp.246-51, IEEE CS Press, 1988.

[Powell & Stroud 2003] D. Powell and R. J. Stroud (Eds.), *Conceptual Model and Architecture of MAFTIA,* MAFTIA Project, Deliverable D21, 2003.

[Schneider 1990] F. B. Schneider, “Implementing fault-tolerant services using the state machine approach: a tutorial”, *ACM Computing Surveys*, 22 (4), pp.299-319, 1990.

[Schneier 1996] B. Schneier, *Applied Cryptography,* Wiley and Sons, Inc., 1996.

[Storey 1996] N. Storey, *Safety-Critical Computer Systems,* Addison-Wesley Longman, Essex, England, 1996.