ANALYSIS OF MULTICS

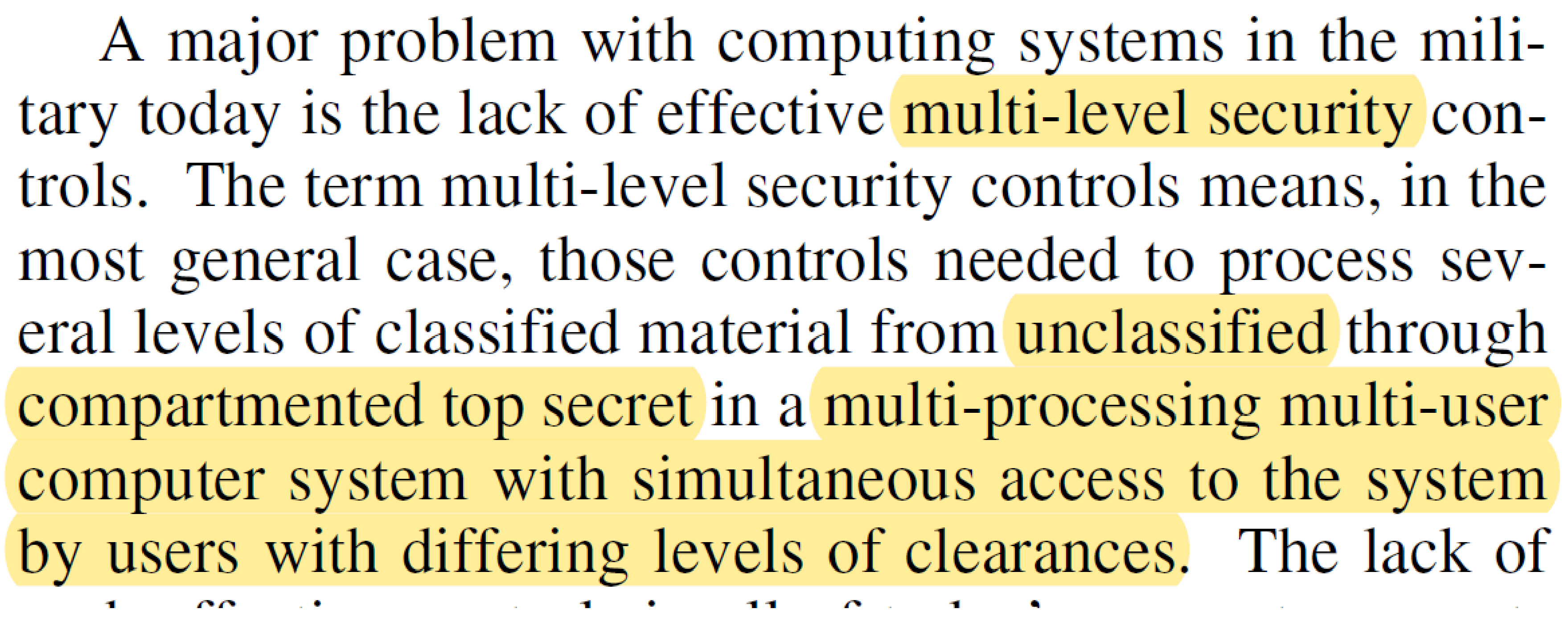
CS 361S

Spring 2020

Seth James Nielson

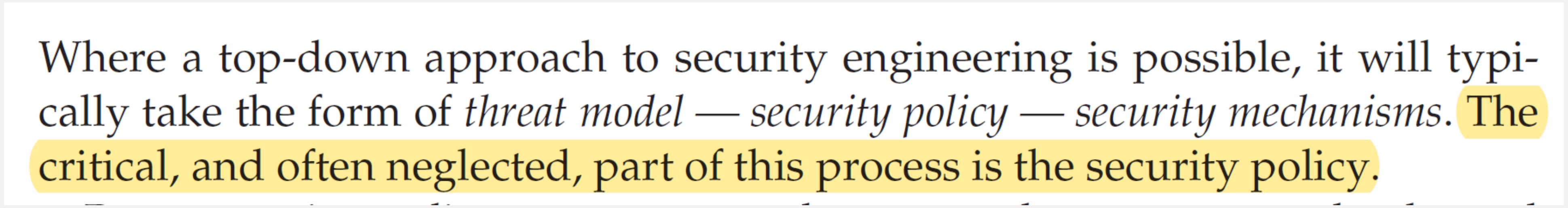
MULTICS SECURITY EVALUATION: VULNERABILITY ANALYSIS

- Originally Published in June 1974
- This was 10 years before Thompson's paper
- Computer security was already a concern in military circles

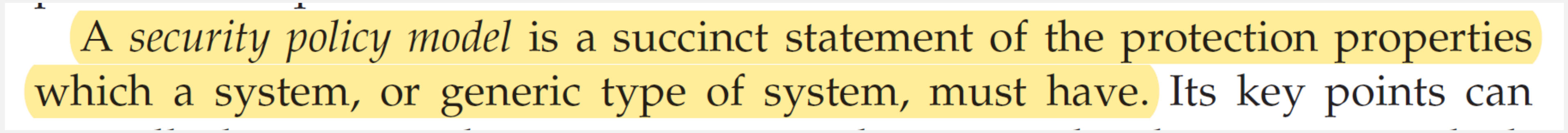

A major problem with computing systems in the military today is the lack of effective multi-level security controls. The term multi-level security controls means, in the most general case, those controls needed to process several levels of classified material from unclassified through compartmented top secret in a multi-processing multi-user computer system with simultaneous access to the system by users with differing levels of clearances. The lack of

GENESIS OF THE PAPER

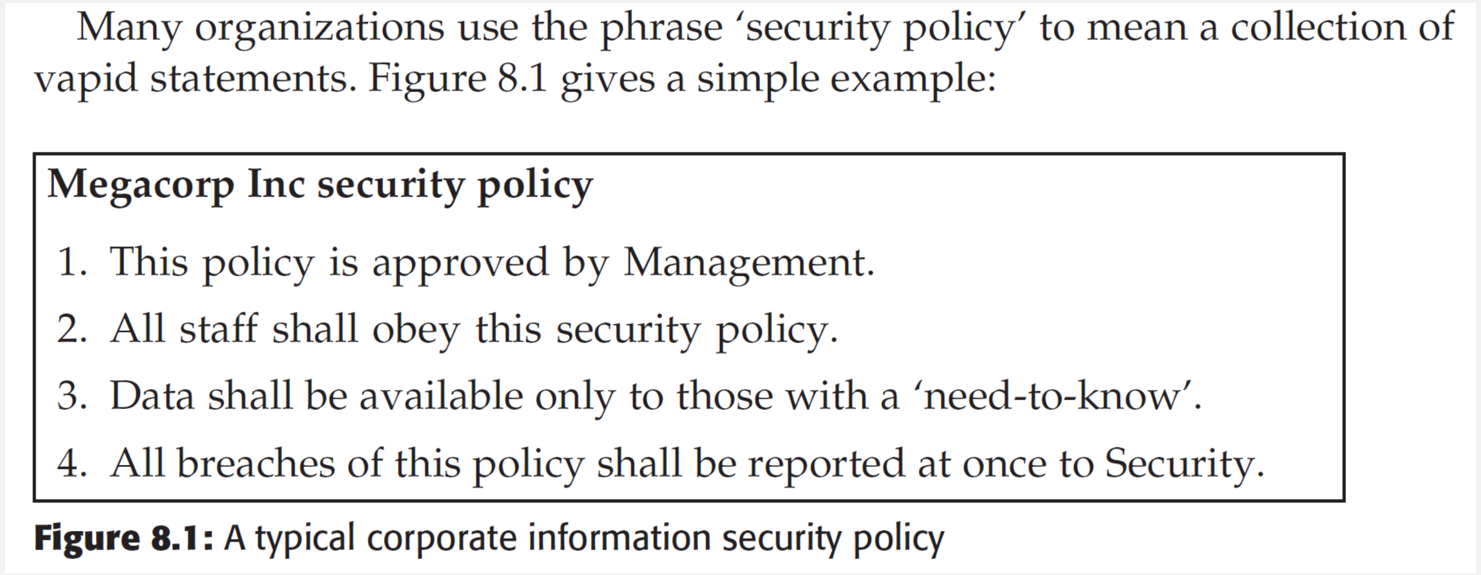
DETOUR TO ANDERSON: POLICY



Where a top-down approach to security engineering is possible, it will typically take the form of *threat model* — *security policy* — *security mechanisms*. The critical, and often neglected, part of this process is the security policy.



A *security policy model* is a succinct statement of the protection properties which a system, or generic type of system, must have. Its key points can



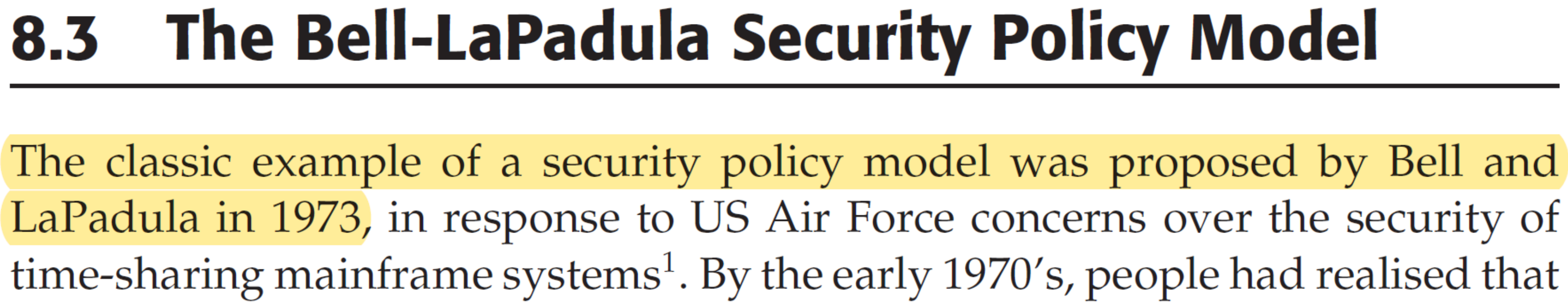
Many organizations use the phrase 'security policy' to mean a collection of vapid statements. Figure 8.1 gives a simple example:

Megacorp Inc security policy

1. This policy is approved by Management.
2. All staff shall obey this security policy.
3. Data shall be available only to those with a 'need-to-know'.
4. All breaches of this policy shall be reported at once to Security.

Figure 8.1: A typical corporate information security policy

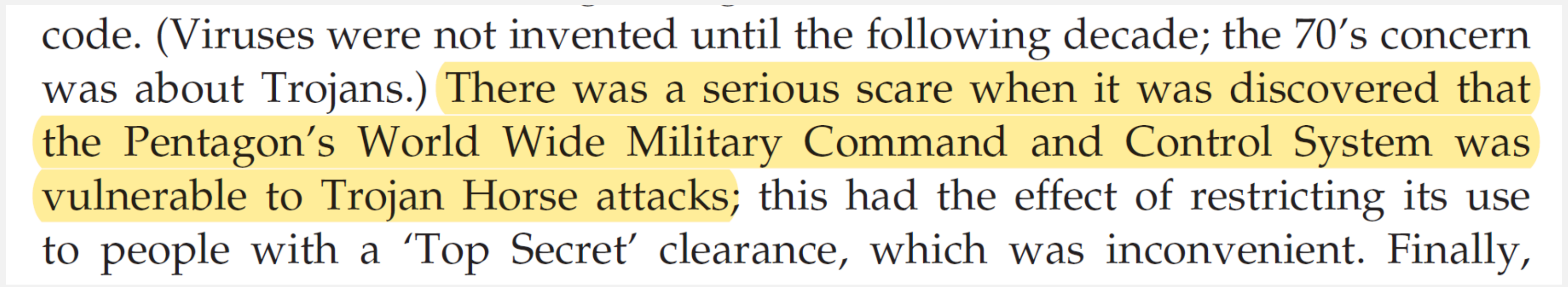
ANDERSON: BELL-LAPADULA

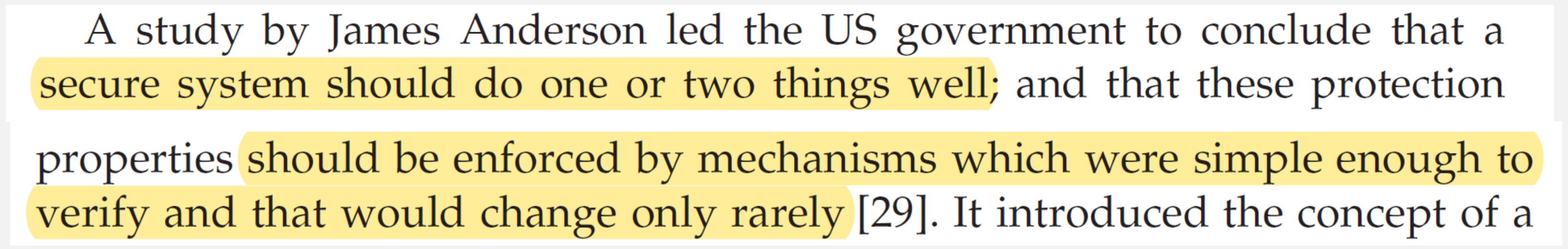


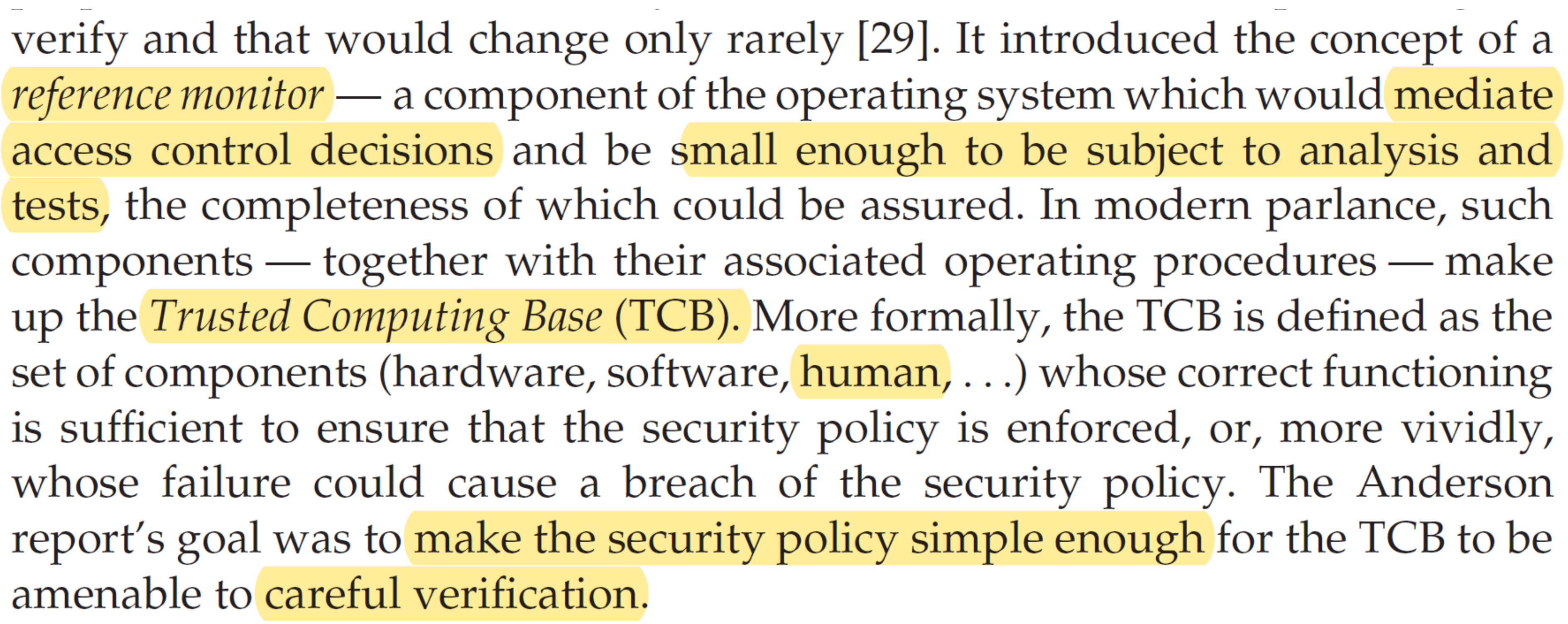
8.3 The Bell-LaPadula Security Policy Model

The classic example of a security policy model was proposed by Bell and LaPadula in 1973, in response to US Air Force concerns over the security of time-sharing mainframe systems[1]. By the early 1970's, people had realised that

ANDERSON: SECURE SYSTEMS THEORY

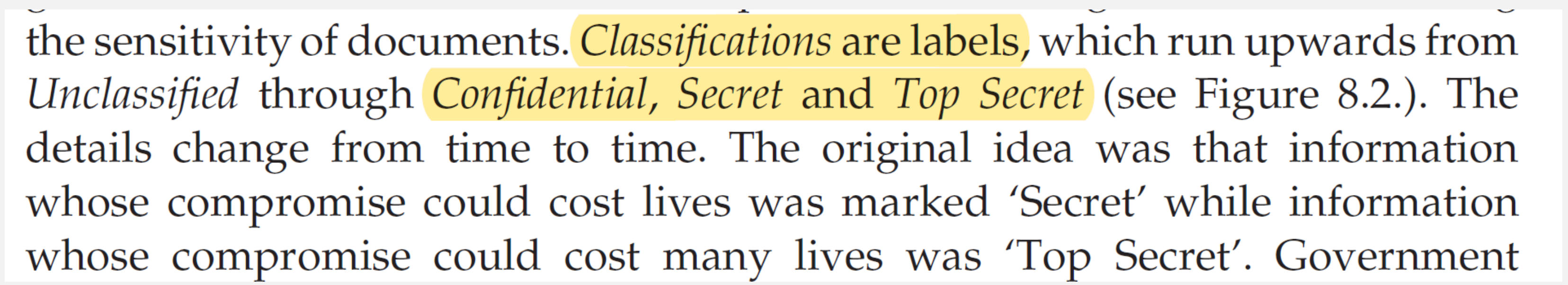

code. (Viruses were not invented until the following decade; the 70's concern was about Trojans.) There was a serious scare when it was discovered that the Pentagon's World Wide Military Command and Control System was vulnerable to Trojan Horse attacks; this had the effect of restricting its use to people with a 'Top Secret' clearance, which was inconvenient. Finally,

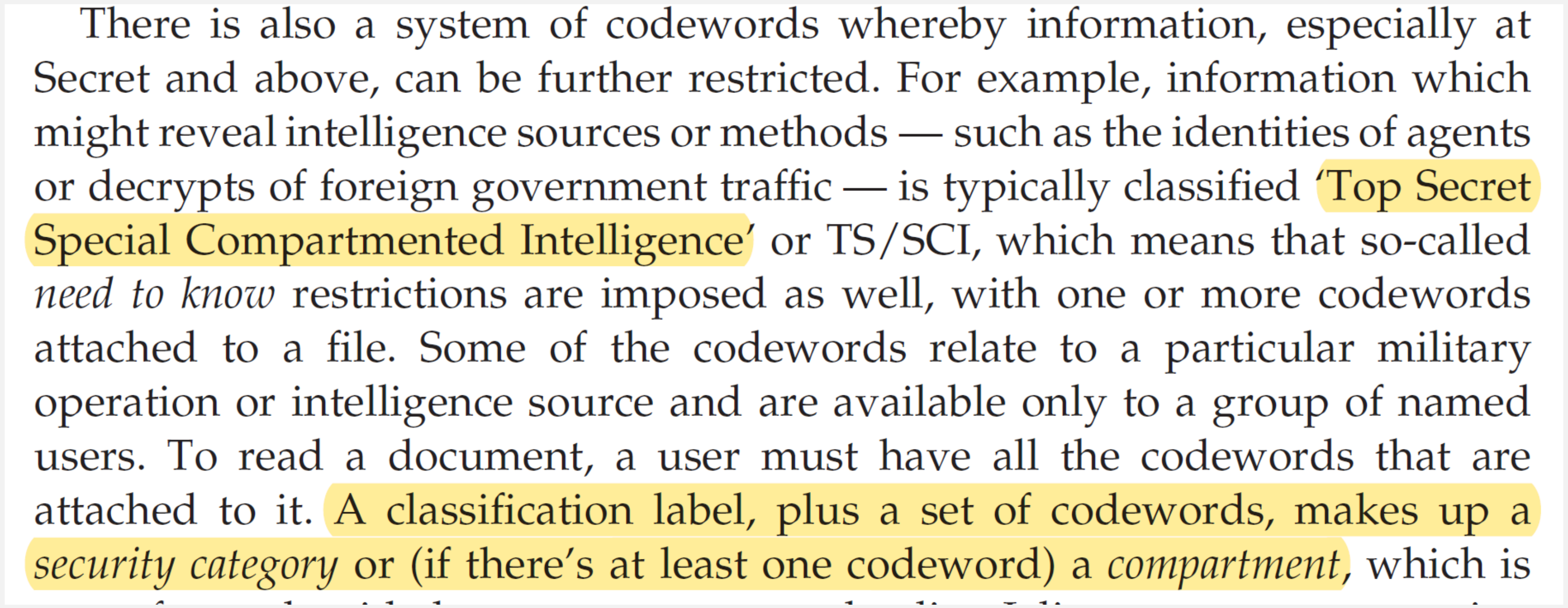

A study by James Anderson led the US government to conclude that a secure system should do one or two things well; and that these protection properties should be enforced by mechanisms which were simple enough to verify and that would change only rarely [29]. It introduced the concept of a


verify and that would change only rarely [29]. It introduced the concept of a *reference monitor* — a component of the operating system which would mediate access control decisions and be small enough to be subject to analysis and tests, the completeness of which could be assured. In modern parlance, such components — together with their associated operating procedures — make up the *Trusted Computing Base* (TCB). More formally, the TCB is defined as the set of components (hardware, software, human, . . .) whose correct functioning is sufficient to ensure that the security policy is enforced, or, more vividly, whose failure could cause a breach of the security policy. The Anderson report's goal was to make the security policy simple enough for the TCB to be amenable to careful verification.

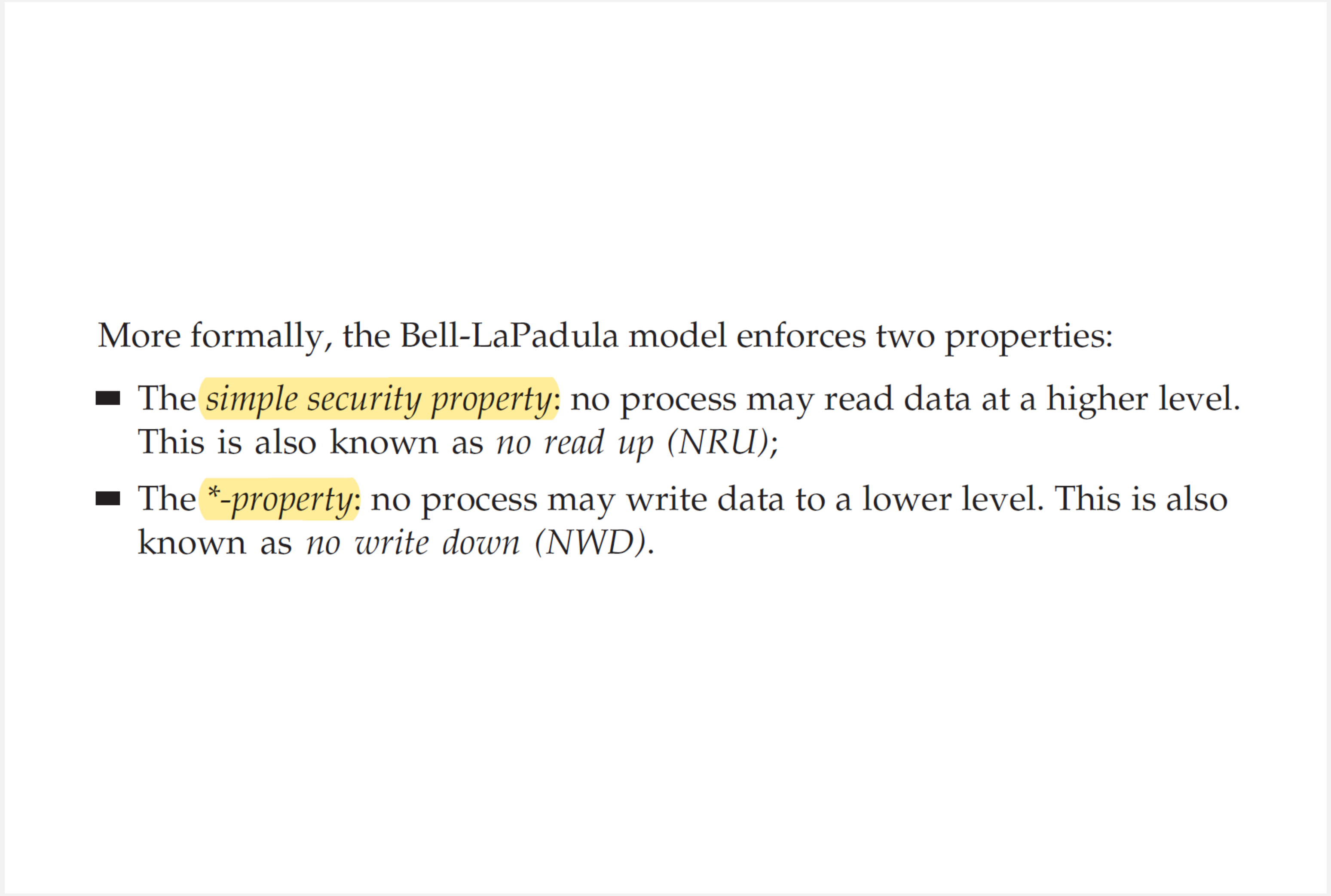
ANDERSON: *TRUSTED* COMPUTING BASE

ANDERSON: CLASSIFICATION


the sensitivity of documents. *Classifications* are labels, which run upwards from *Unclassified* through *Confidential*, *Secret* and *Top Secret* (see Figure 8.2.). The details change from time to time. The original idea was that information whose compromise could cost lives was marked 'Secret' while information whose compromise could cost many lives was 'Top Secret'. Government


There is also a system of codewords whereby information, especially at Secret and above, can be further restricted. For example, information which might reveal intelligence sources or methods — such as the identities of agents or decrypts of foreign government traffic — is typically classified 'Top Secret Special Compartmented Intelligence' or TS/SCI, which means that so-called *need to know* restrictions are imposed as well, with one or more codewords attached to a file. Some of the codewords relate to a particular military operation or intelligence source and are available only to a group of named users. To read a document, a user must have all the codewords that are attached to it. A classification label, plus a set of codewords, makes up a *security category* or (if there's at least one codeword) a *compartment*, which is

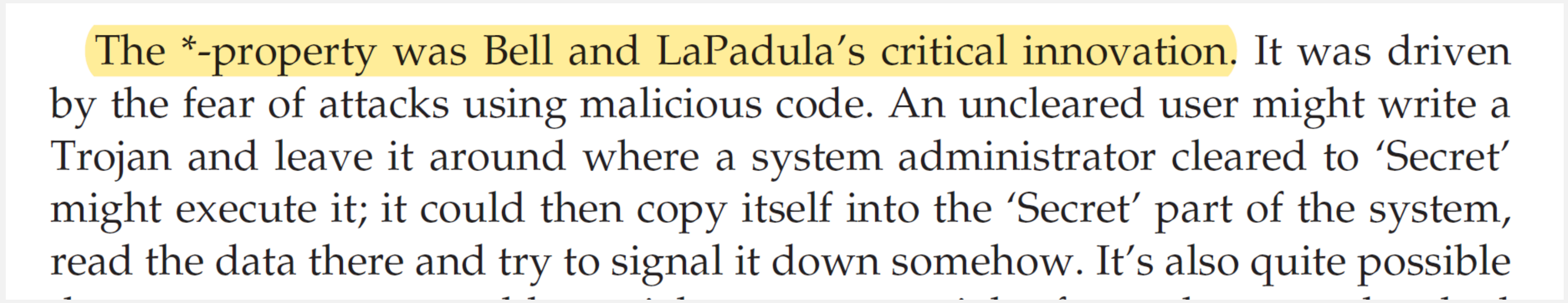
ANDERSON: BELL LAPADULA MODEL



More formally, the Bell-LaPadula model enforces two properties:

- The *simple security property*: no process may read data at a higher level. This is also known as *no read up (NRU)*;
- The **-property*: no process may write data to a lower level. This is also known as *no write down (NWD)*.

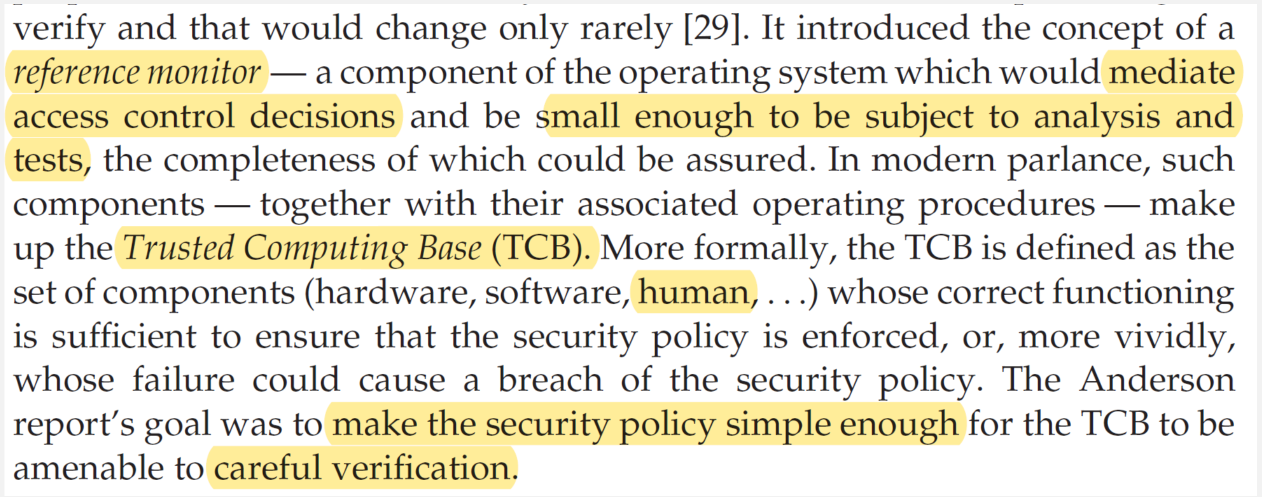
ANDERSON: *-PROPERTY



The *-property was Bell and LaPadula's critical innovation. It was driven by the fear of attacks using malicious code. An uncleared user might write a Trojan and leave it around where a system administrator cleared to 'Secret' might execute it; it could then copy itself into the 'Secret' part of the system, read the data there and try to signal it down somehow. It's also quite possible

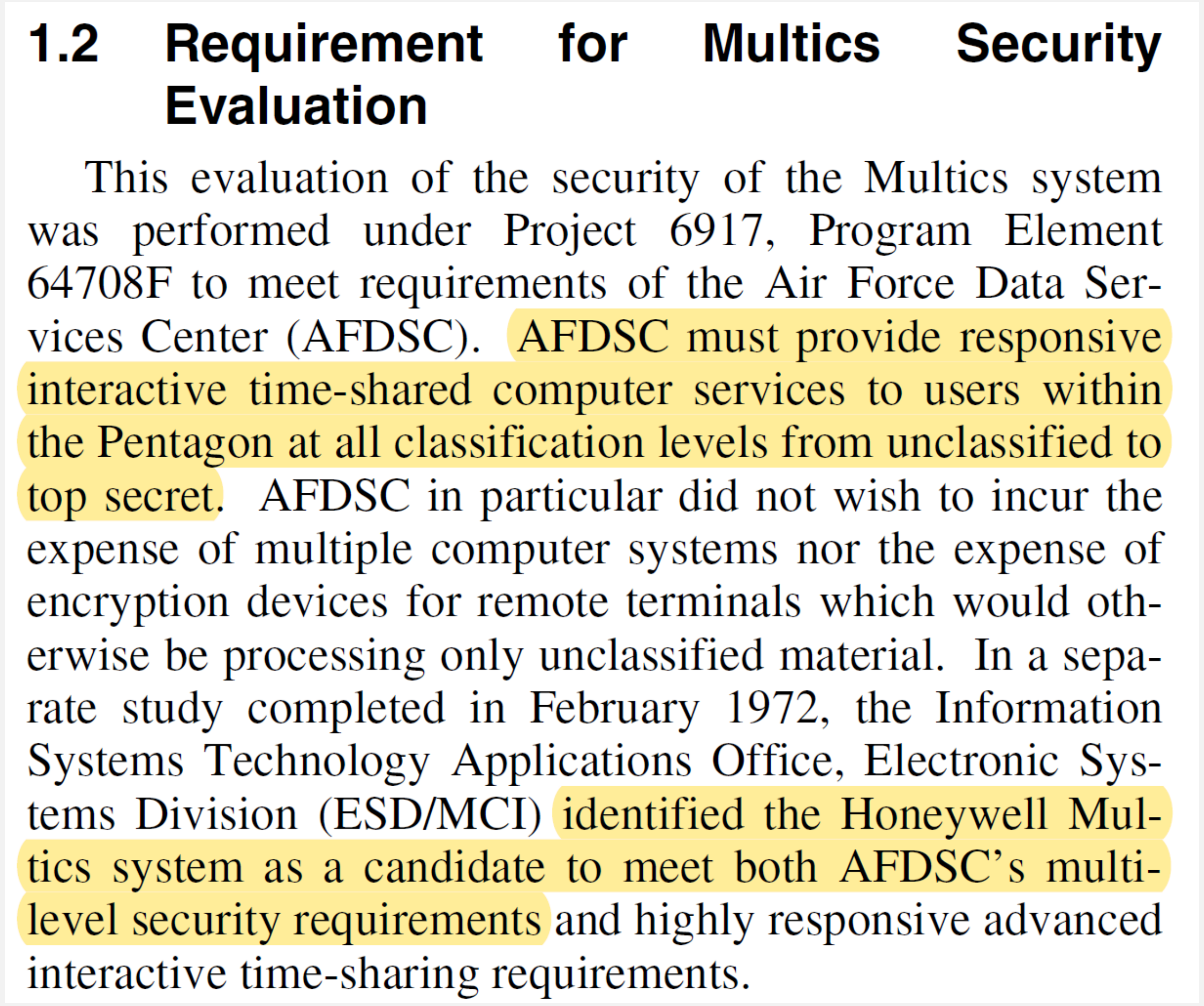
POLICY VS MECHANISM VS TRUST

- Bell LaPadula says NOTHING about implementation
- Implementation is the Mechanism for enforcement
- Remember ***TRUST:*** elements that, if broken, result in violated policy



verify and that would change only rarely [29]. It introduced the concept of a *reference monitor* — a component of the operating system which would mediate access control decisions and be small enough to be subject to analysis and tests, the completeness of which could be assured. In modern parlance, such components — together with their associated operating procedures — make up the *Trusted Computing Base* (TCB). More formally, the TCB is defined as the set of components (hardware, software, human, . . .) whose correct functioning is sufficient to ensure that the security policy is enforced, or, more vividly, whose failure could cause a breach of the security policy. The Anderson report's goal was to make the security policy simple enough for the TCB to be amenable to careful verification.

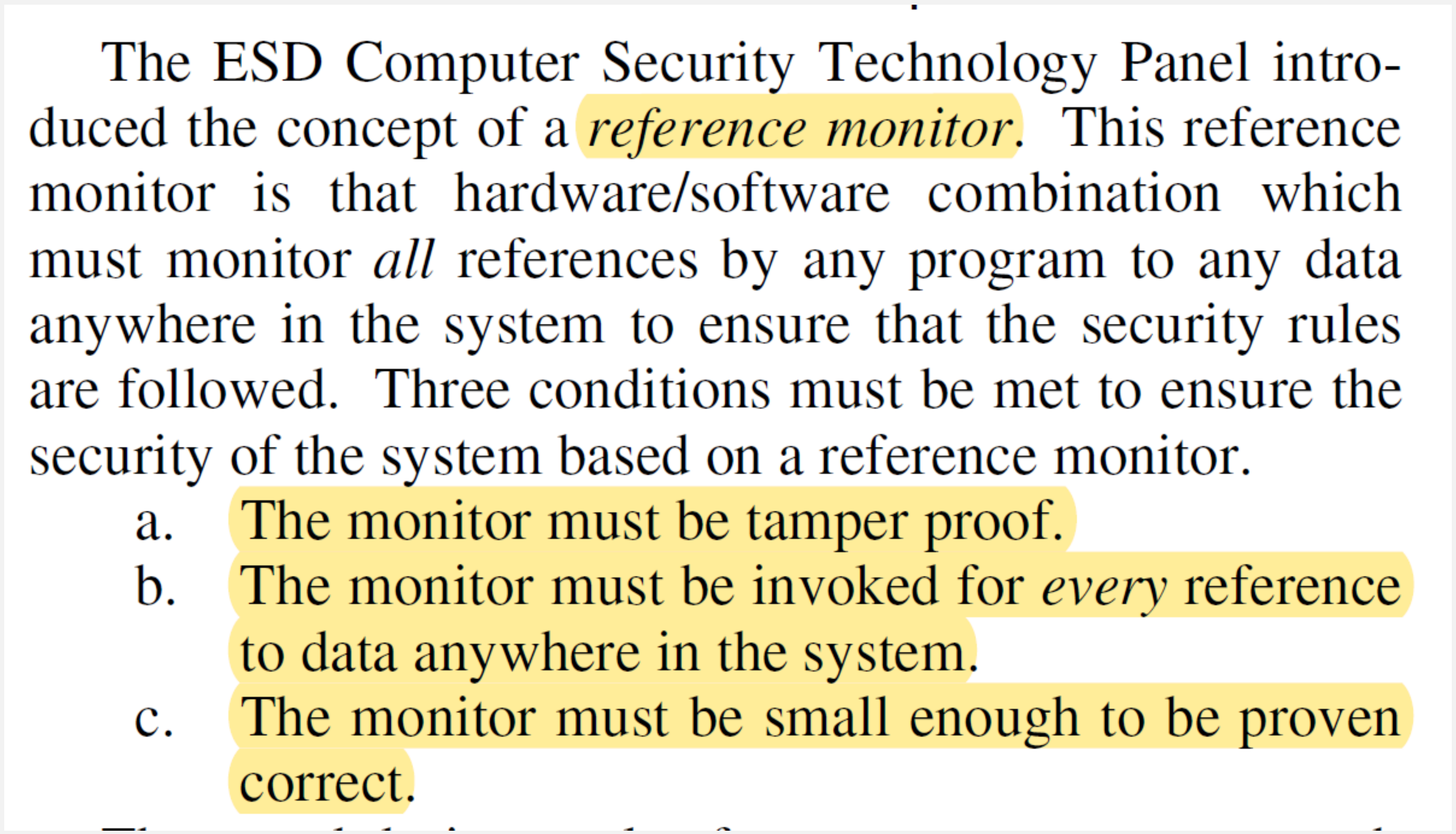
BACK TO MULTICS



1.2 Requirement for Multics Security Evaluation

This evaluation of the security of the Multics system was performed under Project 6917, Program Element 64708F to meet requirements of the Air Force Data Services Center (AFDSC). AFDSC must provide responsive interactive time-shared computer services to users within the Pentagon at all classification levels from unclassified to top secret. AFDSC in particular did not wish to incur the expense of multiple computer systems nor the expense of encryption devices for remote terminals which would otherwise be processing only unclassified material. In a separate study completed in February 1972, the Information Systems Technology Applications Office, Electronic Systems Division (ESD/MCI) identified the Honeywell Multics system as a candidate to meet both AFDSC's multilevel security requirements and highly responsive advanced interactive time-sharing requirements.

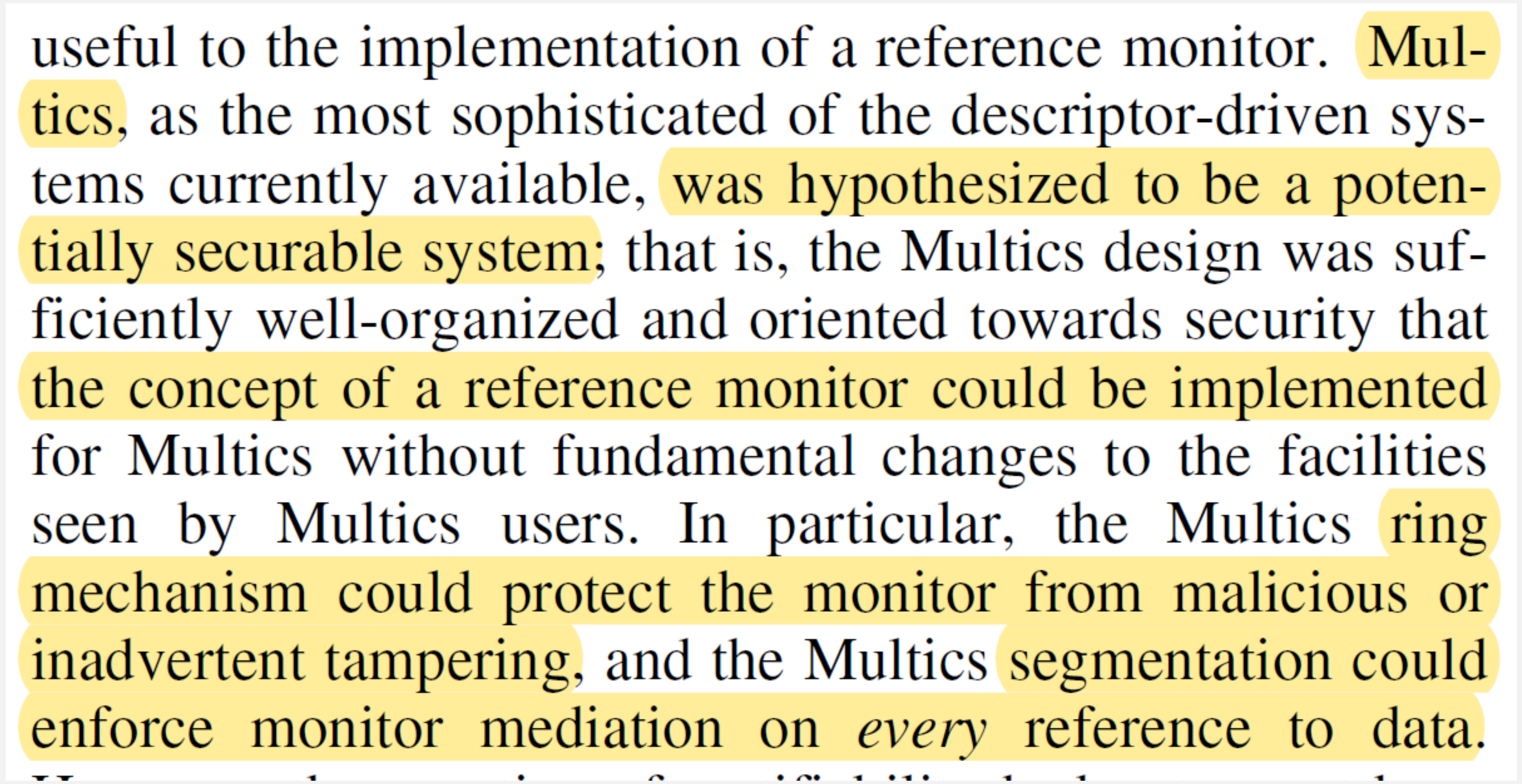
REFERENCE MONITORS AGAIN



The ESD Computer Security Technology Panel introduced the concept of a *reference monitor*. This reference monitor is that hardware/software combination which must monitor *all* references by any program to any data anywhere in the system to ensure that the security rules are followed. Three conditions must be met to ensure the security of the system based on a reference monitor.

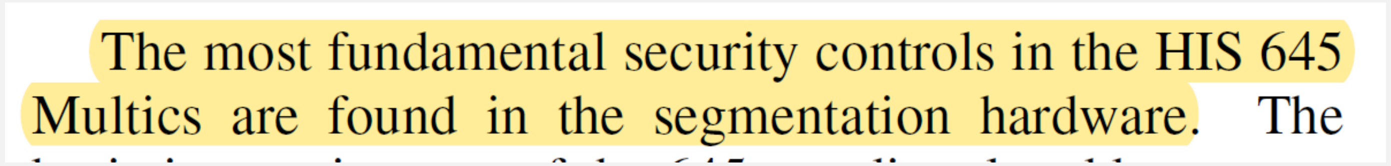
a. The monitor must be tamper proof.
b. The monitor must be invoked for *every* reference to data anywhere in the system.
c. The monitor must be small enough to be proven correct.

MULTICS REFERENCE MONITOR

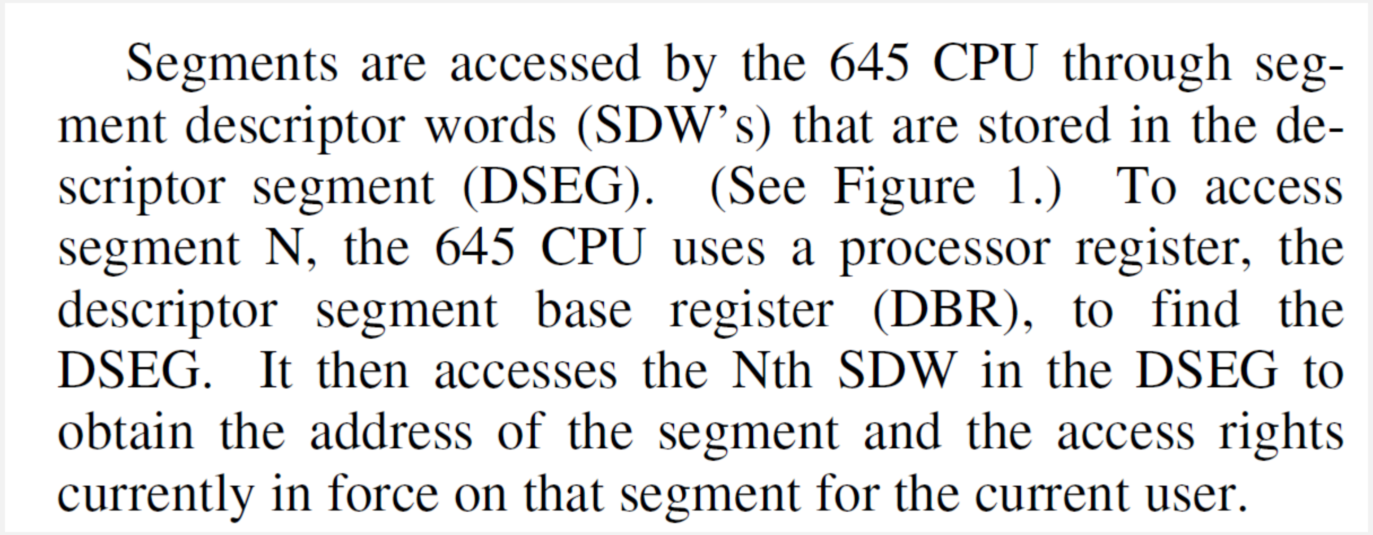


useful to the implementation of a reference monitor. Multics, as the most sophisticated of the descriptor-driven systems currently available, was hypothesized to be a potentially securable system; that is, the Multics design was sufficiently well-organized and oriented towards security that the concept of a reference monitor could be implemented for Multics without fundamental changes to the facilities seen by Multics users. In particular, the Multics ring mechanism could protect the monitor from malicious or inadvertent tampering, and the Multics segmentation could enforce monitor mediation on *every* reference to data.

SEGMENTATION



The most fundamental security controls in the HIS 645 Multics are found in the segmentation hardware. The



Segments are accessed by the 645 CPU through segment descriptor words (SDW's) that are stored in the descriptor segment (DSEG). (See Figure 1.) To access segment N, the 645 CPU uses a processor register, the descriptor segment base register (DBR), to find the DSEG. It then accesses the Nth SDW in the DSEG to obtain the address of the segment and the access rights currently in force on that segment for the current user.

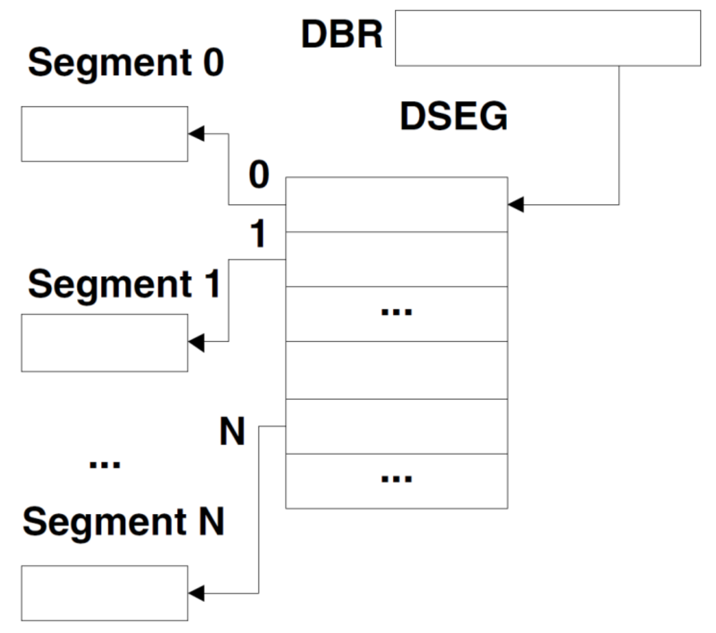
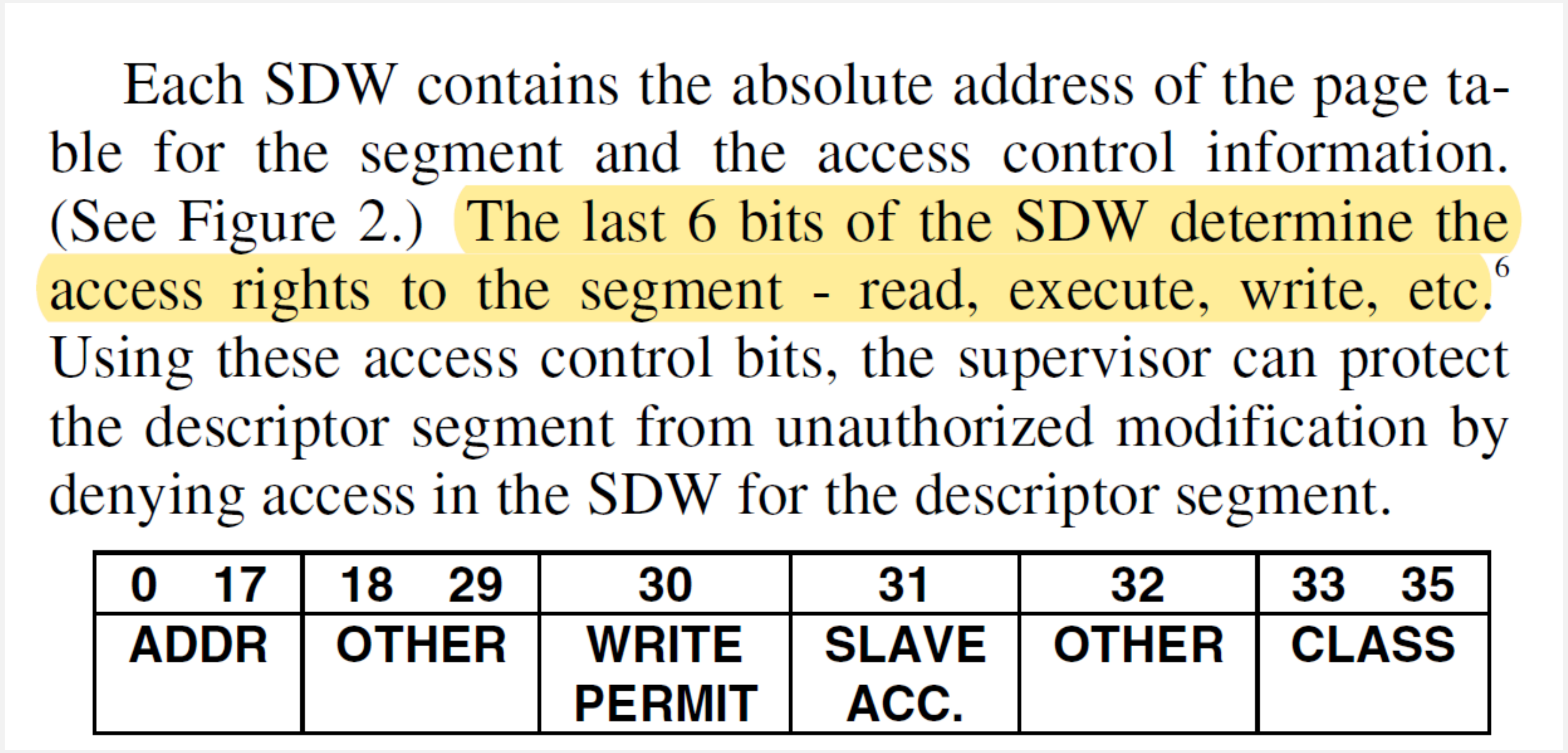


Figure 1. Segmentation Hardware

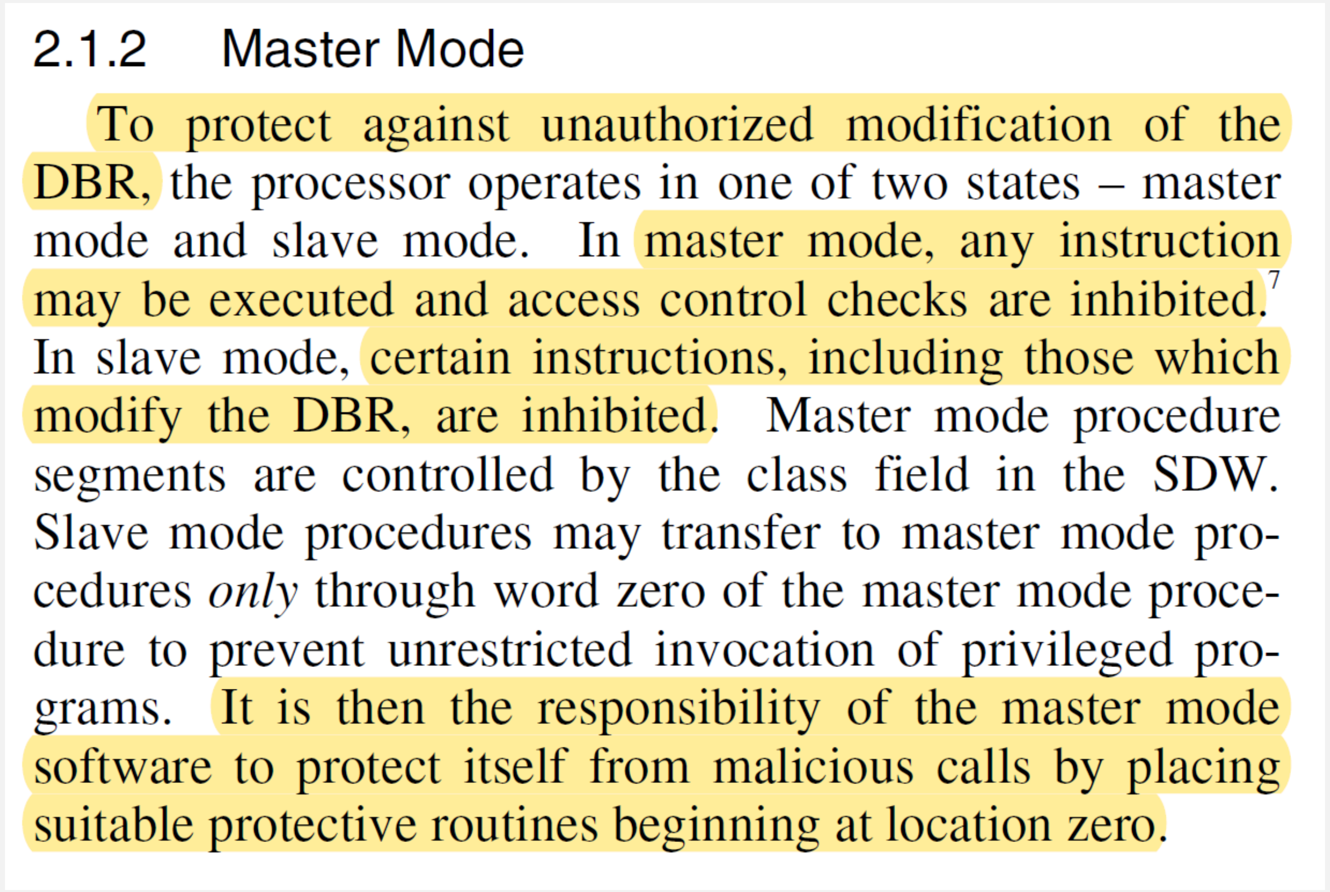
SEGMENT ACCESS CONTROL



Each SDW contains the absolute address of the page table for the segment and the access control information. (See Figure 2.) The last 6 bits of the SDW determine the access rights to the segment - read, execute, write, etc.[6] Using these access control bits, the supervisor can protect the descriptor segment from unauthorized modification by denying access in the SDW for the descriptor segment.

0 17	18 29	30	31	32	33 35
ADDR	OTHER	WRITE PERMIT	SLAVE ACC.	OTHER	CLASS

MASTER/SLAVE MODE



2.1.2 Master Mode

To protect against unauthorized modification of the DBR, the processor operates in one of two states – master mode and slave mode. In master mode, any instruction may be executed and access control checks are inhibited.[7] In slave mode, certain instructions, including those which modify the DBR, are inhibited. Master mode procedure segments are controlled by the class field in the SDW. Slave mode procedures may transfer to master mode procedures *only* through word zero of the master mode procedure to prevent unrestricted invocation of privileged programs. It is then the responsibility of the master mode software to protect itself from malicious calls by placing suitable protective routines beginning at location zero.

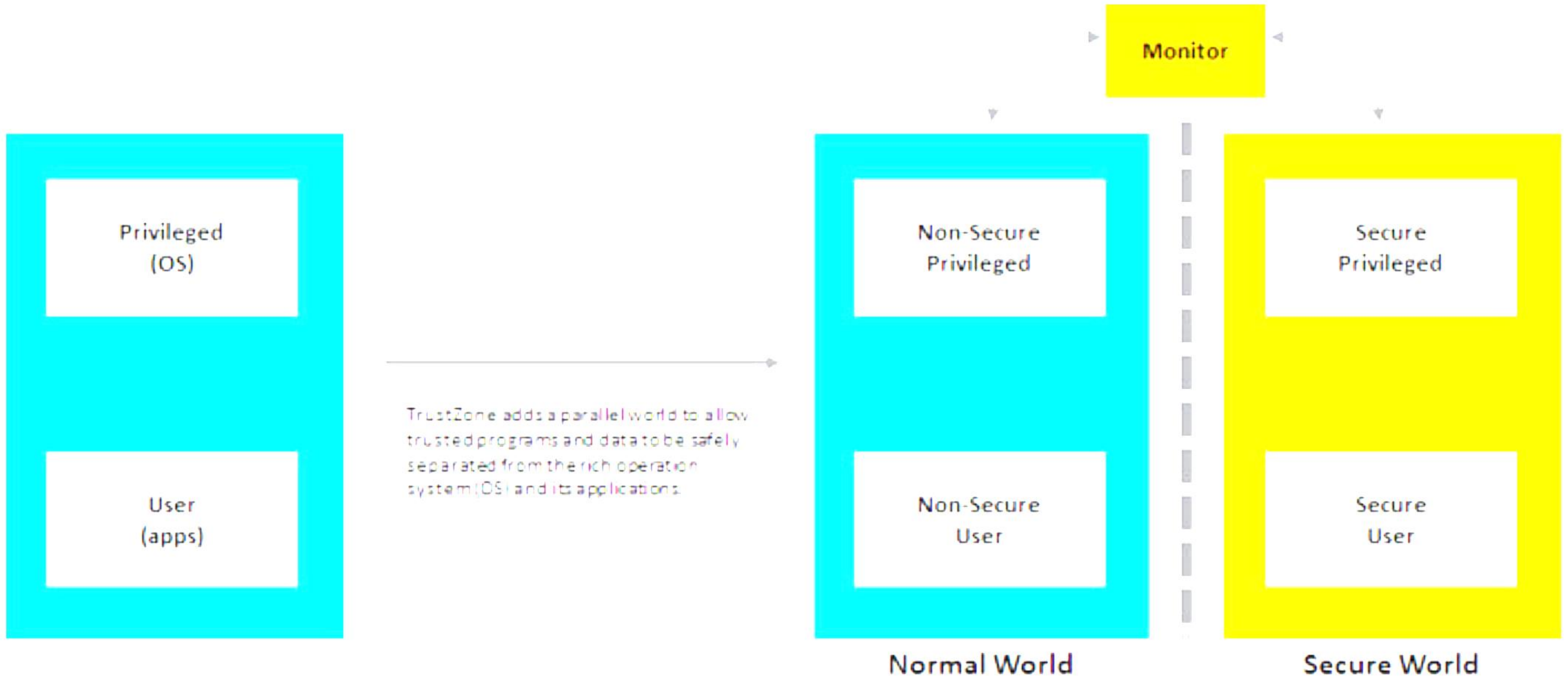
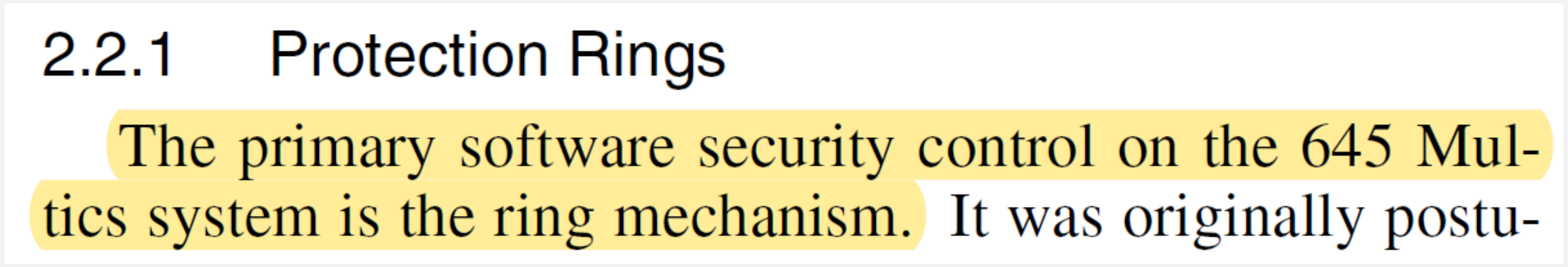

Monitor
Privileged
(OS)
User
(apps)
TrustZone adds a parallel world to allow trusted programs and data to be safely separated from the rich operation system (OS) and its applications.
Non-Secure
Privileged
Non-Secure
User
Secure
Privileged
Secure
User
Normal World
Secure World

Figure 3. TrustZone creates two parallel execution worlds

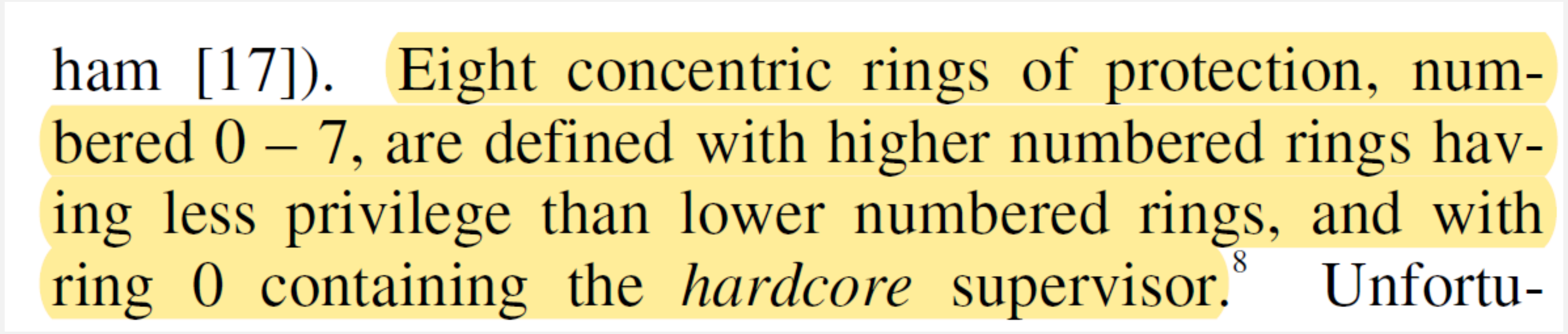
DETOUR: ARM TRUSTZONE

RINGS

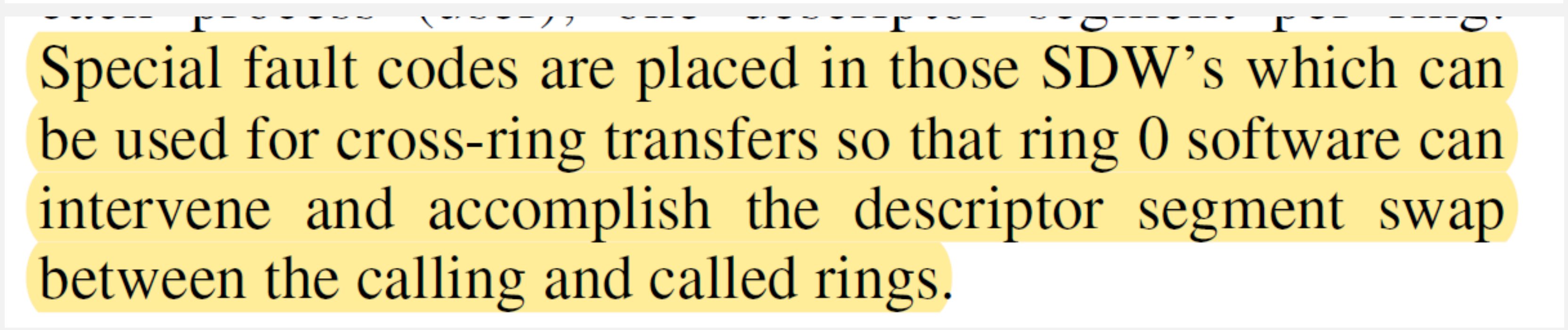


2.2.1 Protection Rings

The primary software security control on the 645 Multics system is the ring mechanism. It was originally postu-

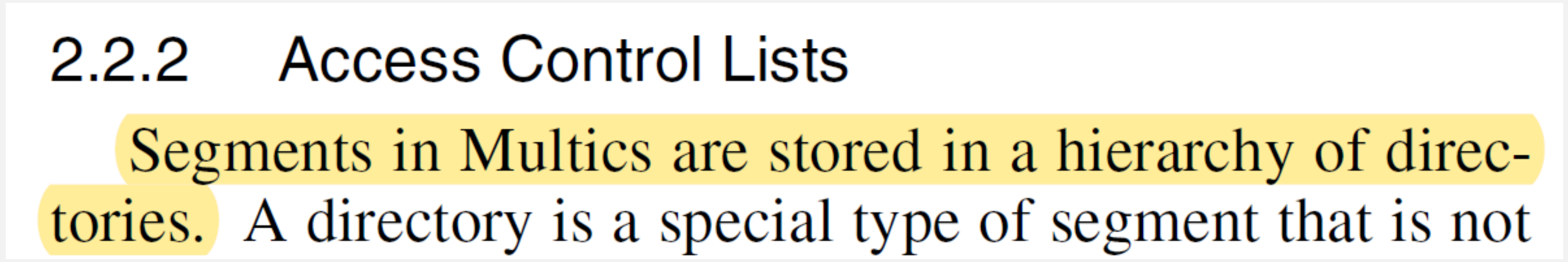


ham [17]). Eight concentric rings of protection, numbered 0 – 7, are defined with higher numbered rings having less privilege than lower numbered rings, and with ring 0 containing the *hardcore* supervisor.[8] Unfortu-



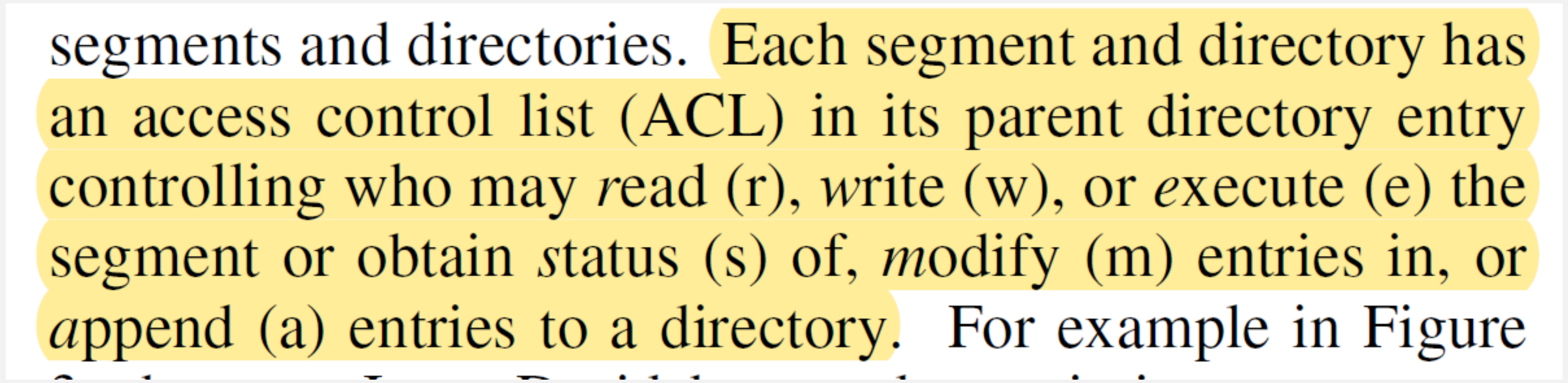
[illegible]
Special fault codes are placed in those SDW's which can be used for cross-ring transfers so that ring 0 software can intervene and accomplish the descriptor segment swap between the calling and called rings.

ACCESS CONTROL LISTS (ACLS)

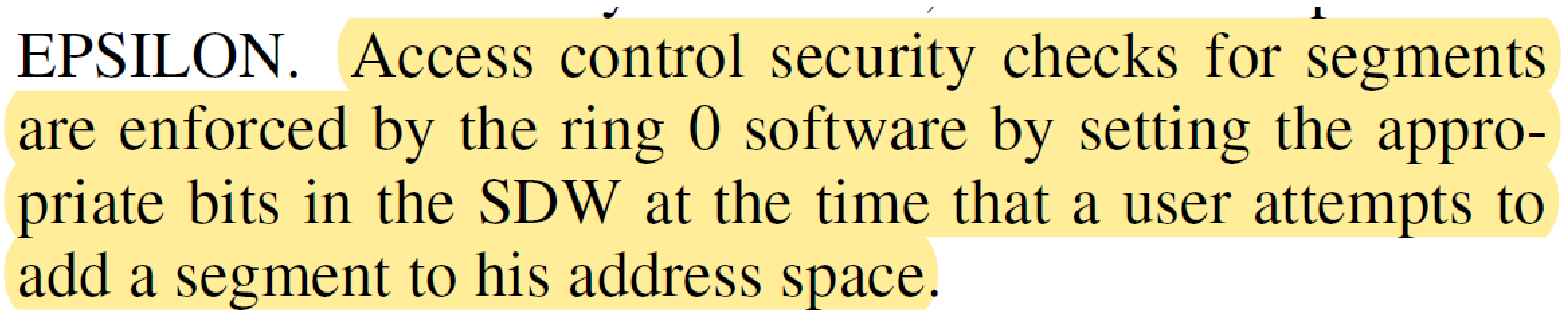


2.2.2 Access Control Lists

Segments in Multics are stored in a hierarchy of directories. A directory is a special type of segment that is not

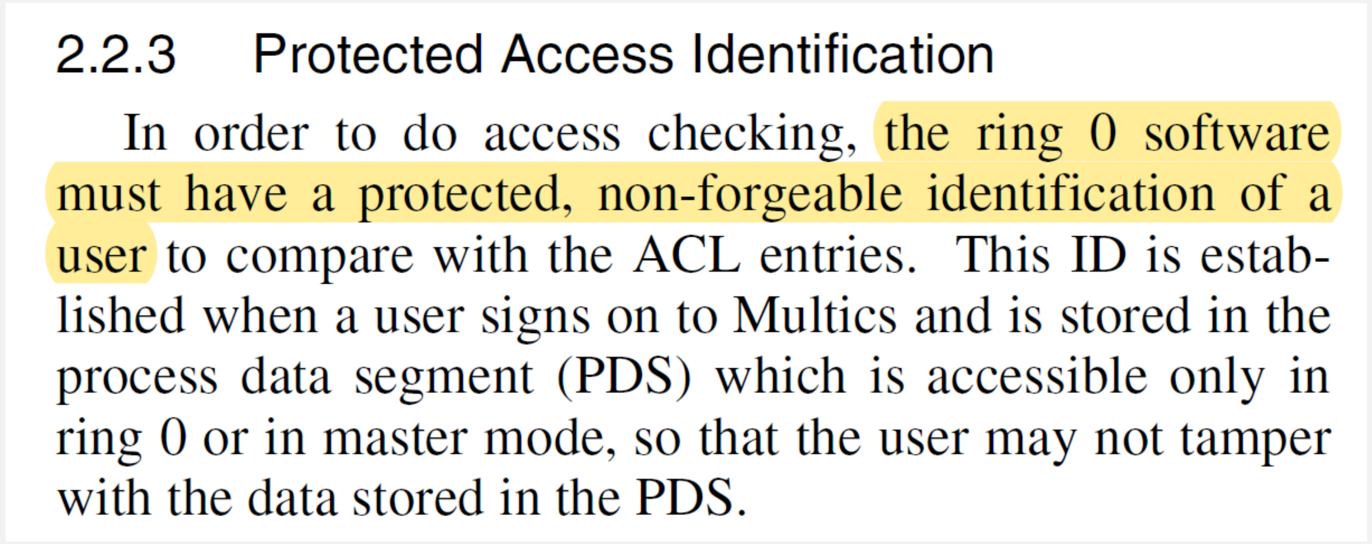


segments and directories. Each segment and directory has an access control list (ACL) in its parent directory entry controlling who may *r*ead (r), *w*rite (w), or *e*xecute (e) the segment or obtain *s*tatus (s) of, *m*odify (m) entries in, or *a*ppend (a) entries to a directory. For example in Figure


EPSILON. Access control security checks for segments are enforced by the ring 0 software by setting the appropriate bits in the SDW at the time that a user attempts to add a segment to his address space.

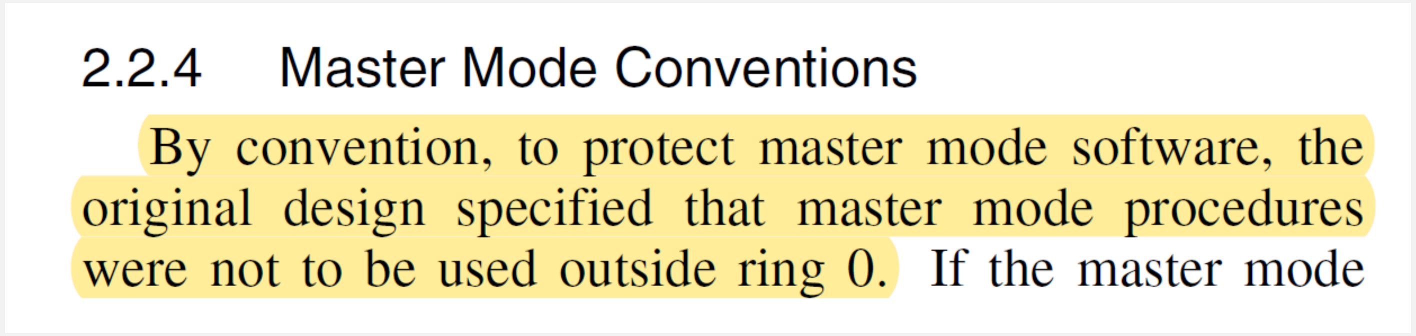
ACL ENFORCEMENT

USER IDENTIFICATION


2.2.3 Protected Access Identification

In order to do access checking, the ring 0 software must have a protected, non-forgeable identification of a user to compare with the ACL entries. This ID is established when a user signs on to Multics and is stored in the process data segment (PDS) which is accessible only in ring 0 or in master mode, so that the user may not tamper with the data stored in the PDS.

MASTER MODE PROTECTION



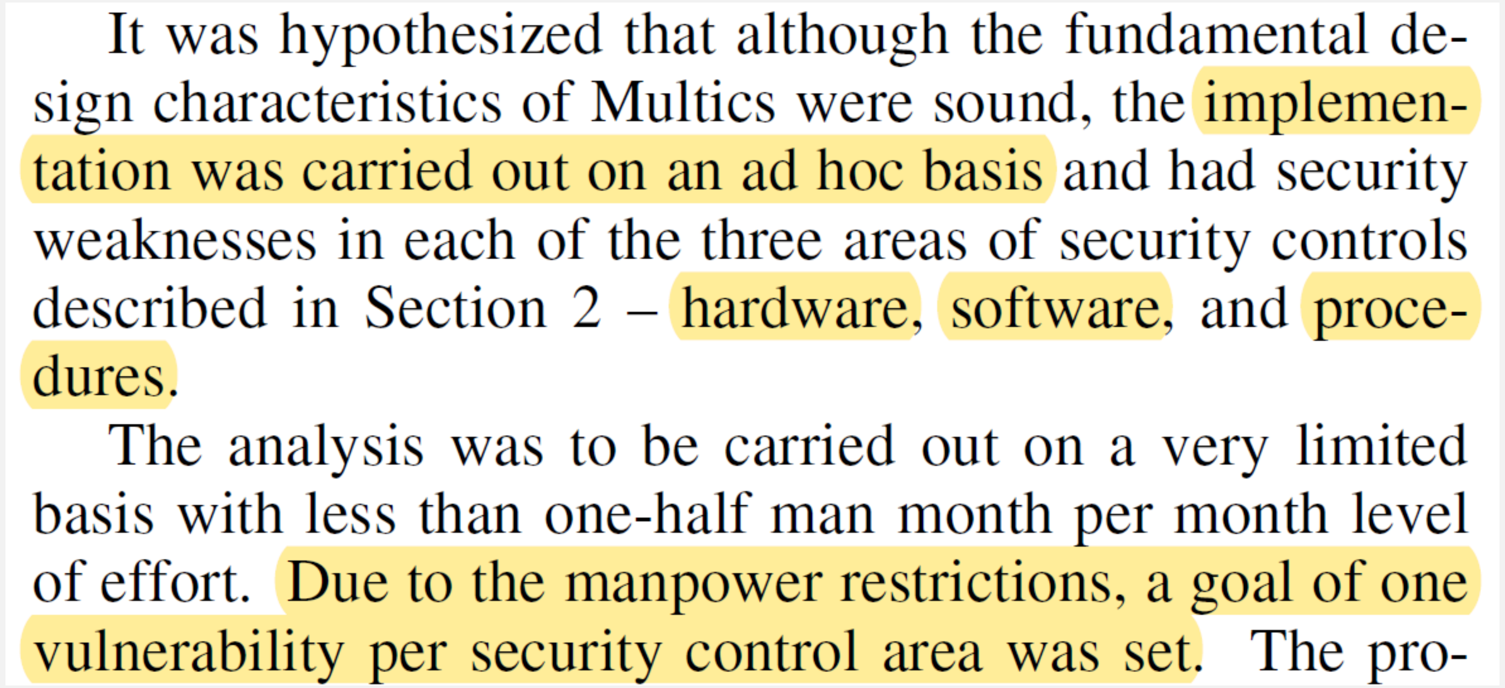
2.2.4 Master Mode Conventions

By convention, to protect master mode software, the original design specified that master mode procedures were not to be used outside ring 0. If the master mode

PROCEDURAL SECURITY

- Encrypted Passwords
- Login/Logout Audit Trail
- Protected and auditable ring-0 software

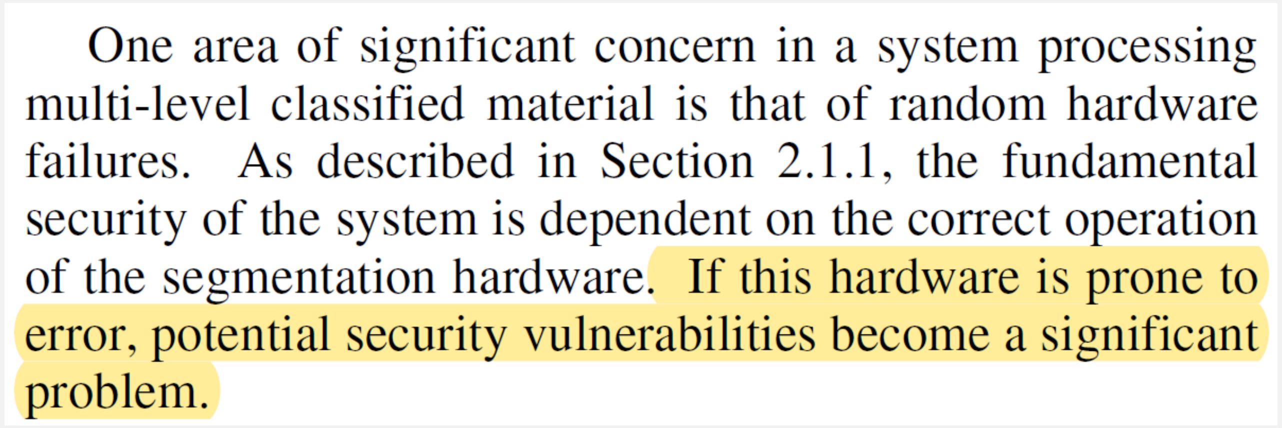
VULNERABILITY ANALYSIS PLAN

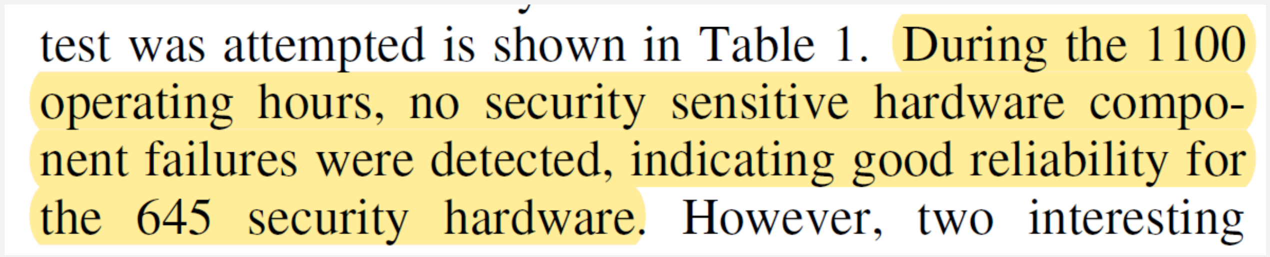


It was hypothesized that although the fundamental design characteristics of Multics were sound, the implementation was carried out on an ad hoc basis and had security weaknesses in each of the three areas of security controls described in Section 2 – hardware, software, and procedures.

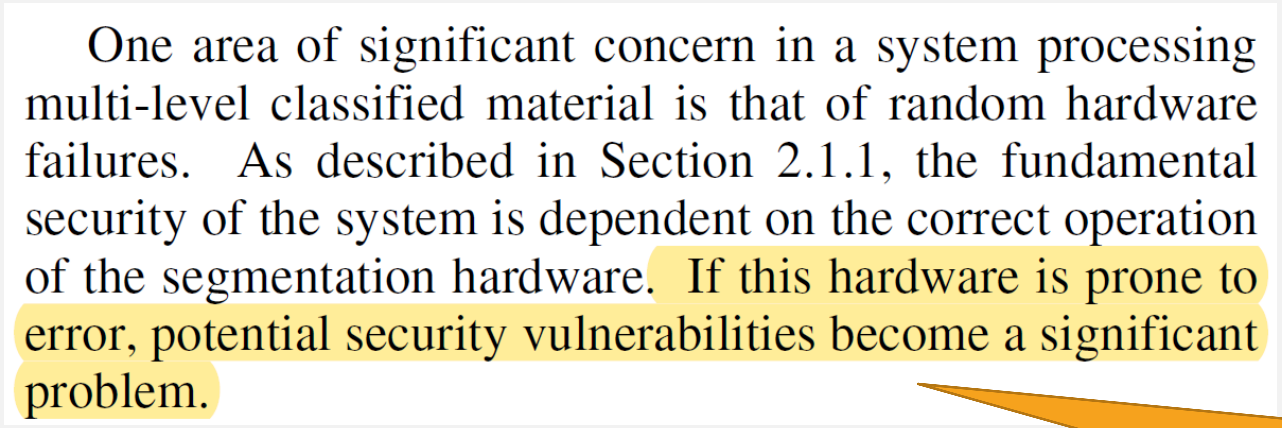
The analysis was to be carried out on a very limited basis with less than one-half man month per month level of effort. Due to the manpower restrictions, a goal of one vulnerability per security control area was set. The pro-

RANDOM HARDWARE FAILURE TEST

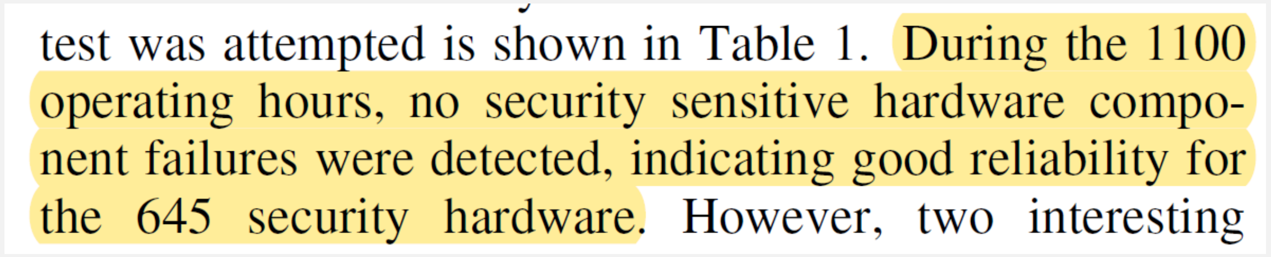

One area of significant concern in a system processing multi-level classified material is that of random hardware failures. As described in Section 2.1.1, the fundamental security of the system is dependent on the correct operation of the segmentation hardware. If this hardware is prone to error, potential security vulnerabilities become a significant problem.


test was attempted is shown in Table 1. During the 1100 operating hours, no security sensitive hardware component failures were detected, indicating good reliability for the 645 security hardware. However, two interesting

DETOUR:
RELIABILITY IS ***NOT*** SECURITY



One area of significant concern in a system processing multi-level classified material is that of random hardware failures. As described in Section 2.1.1, the fundamental security of the system is dependent on the correct operation of the segmentation hardware. If this hardware is prone to error, potential security vulnerabilities become a significant problem.



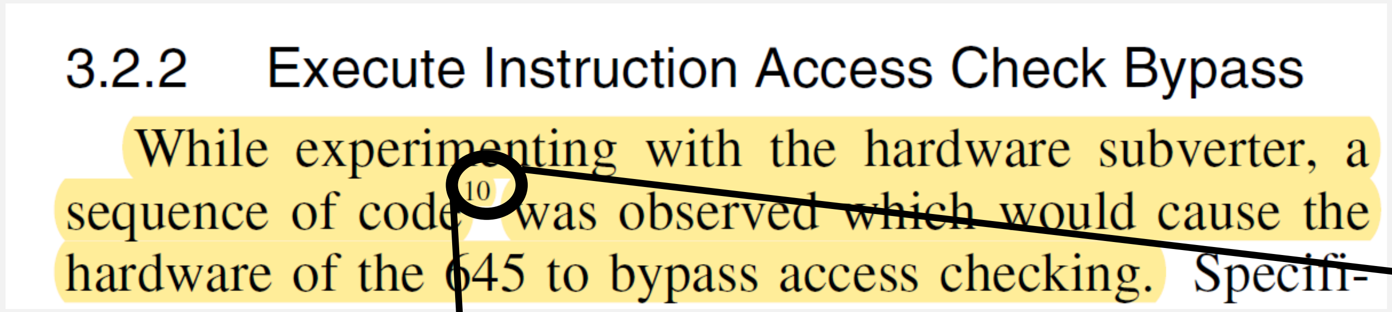
test was attempted is shown in Table 1. During the 1100 operating hours, no security sensitive hardware component failures were detected, indicating good reliability for the 645 security hardware. However, two interesting

This is NOT necessarily true. The *Reliability* of the hardware is not always a good indication of security/insecurity.

MORE ABOUT RELIABILITY

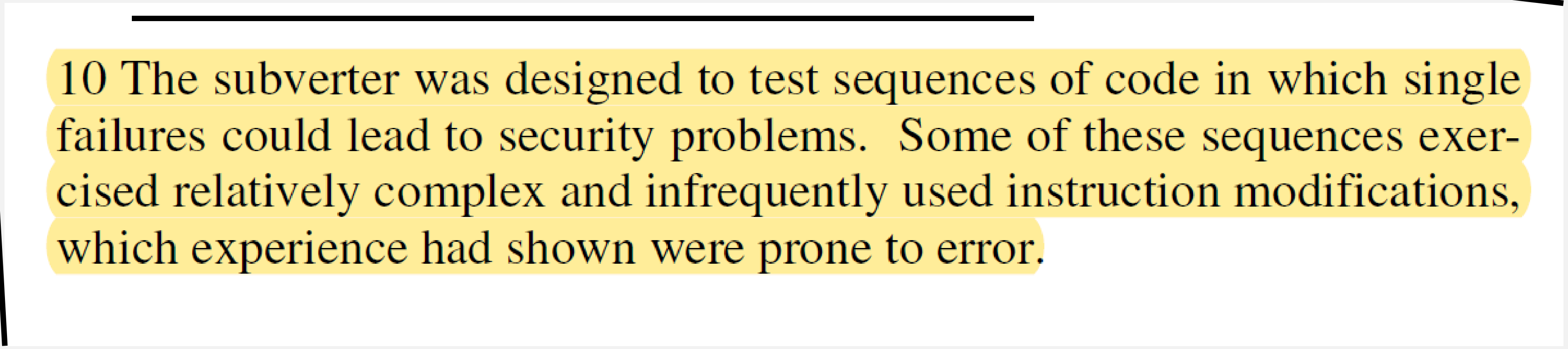
- In classical software engineering, ***reliability*** is key!
- In security engineering, sometimes reliability is incorrectly used as a metric
- ***How frequently the system fails is not typically what matters***
- Rather, small bugs that ***ALMOST NEVER HAPPEN*** are the problem!

VULNERABILITY FOUND DURING RELIABILITY TESTING

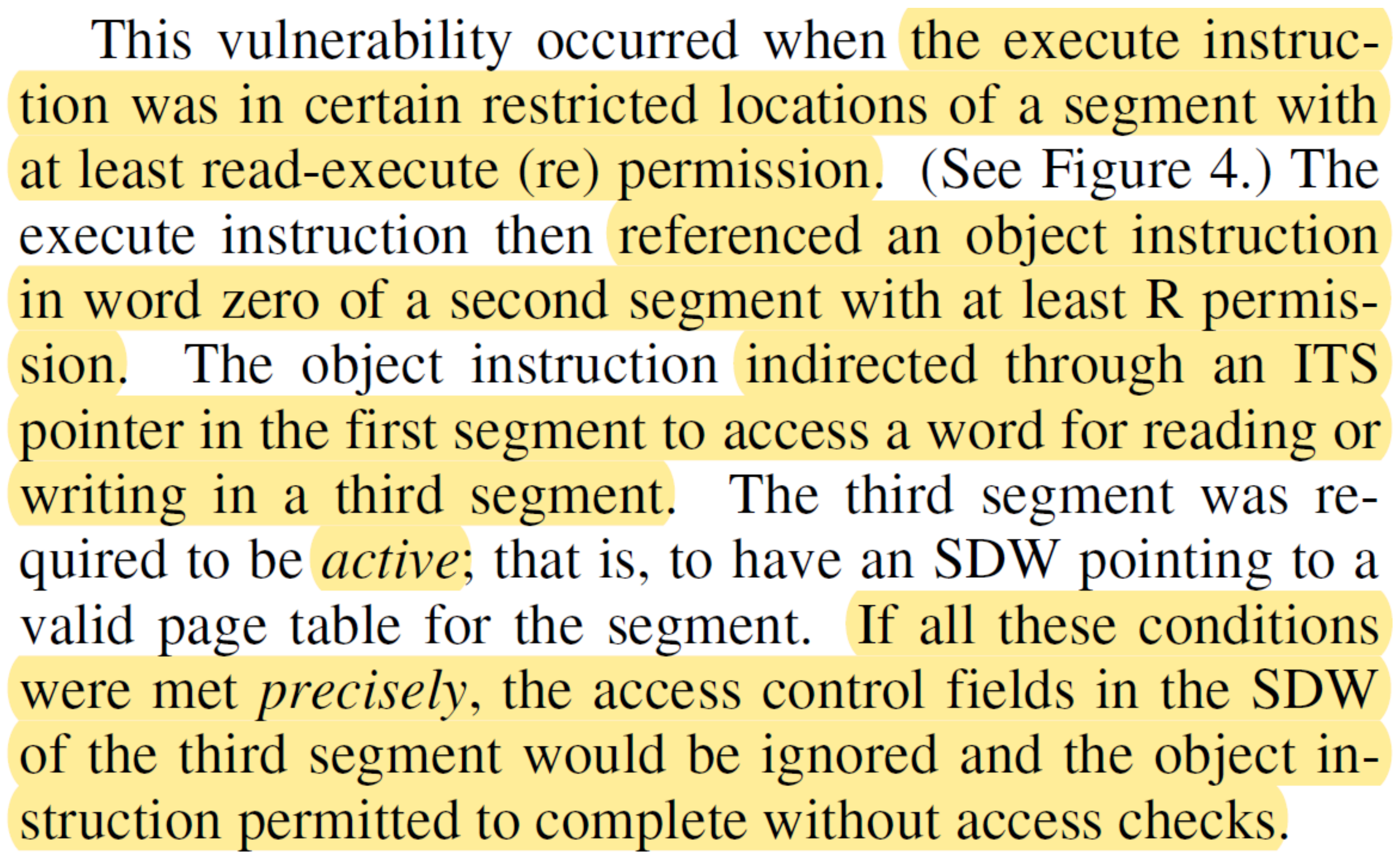


3.2.2 Execute Instruction Access Check Bypass

While experimenting with the hardware subverter, a sequence of code[10] was observed which would cause the hardware of the 645 to bypass access checking. Specifi-



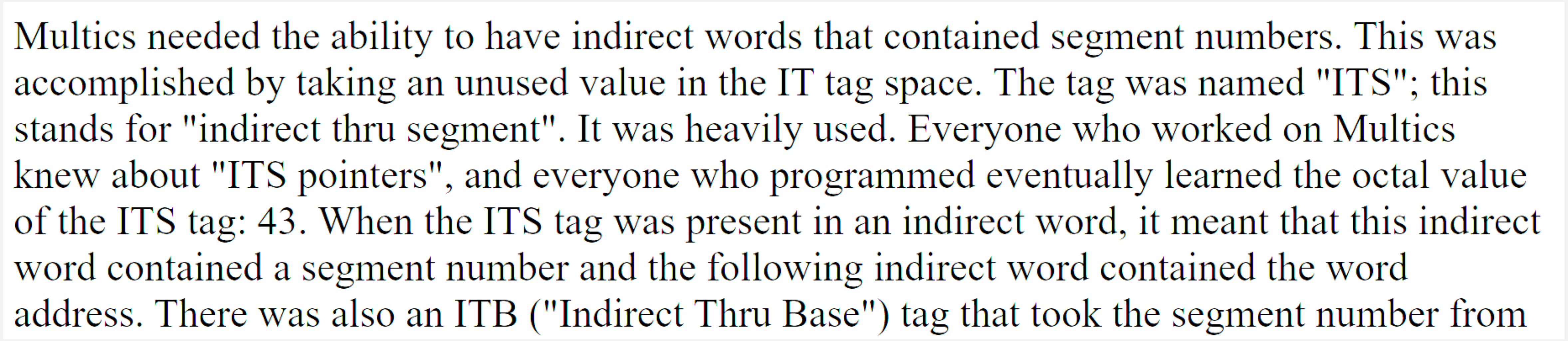
10 The subverter was designed to test sequences of code in which single failures could lead to security problems. Some of these sequences exercised relatively complex and infrequently used instruction modifications, which experience had shown were prone to error.



This vulnerability occurred when the execute instruction was in certain restricted locations of a segment with at least read-execute (re) permission. (See Figure 4.) The execute instruction then referenced an object instruction in word zero of a second segment with at least R permission. The object instruction indirected through an ITS pointer in the first segment to access a word for reading or writing in a third segment. The third segment was required to be *active*; that is, to have an SDW pointing to a valid page table for the segment. If all these conditions were met *precisely*, the access control fields in the SDW of the third segment would be ignored and the object instruction permitted to complete without access checks.

SEQUENCE OF CODE

DETOUR: WHAT IS AN ITS POINTER?



Multics needed the ability to have indirect words that contained segment numbers. This was accomplished by taking an unused value in the IT tag space. The tag was named "ITS"; this stands for "indirect thru segment". It was heavily used. Everyone who worked on Multics knew about "ITS pointers", and everyone who programmed eventually learned the octal value of the ITS tag: 43. When the ITS tag was present in an indirect word, it meant that this indirect word contained a segment number and the following indirect word contained the word address. There was also an ITB ("Indirect Thru Base") tag that took the segment number from

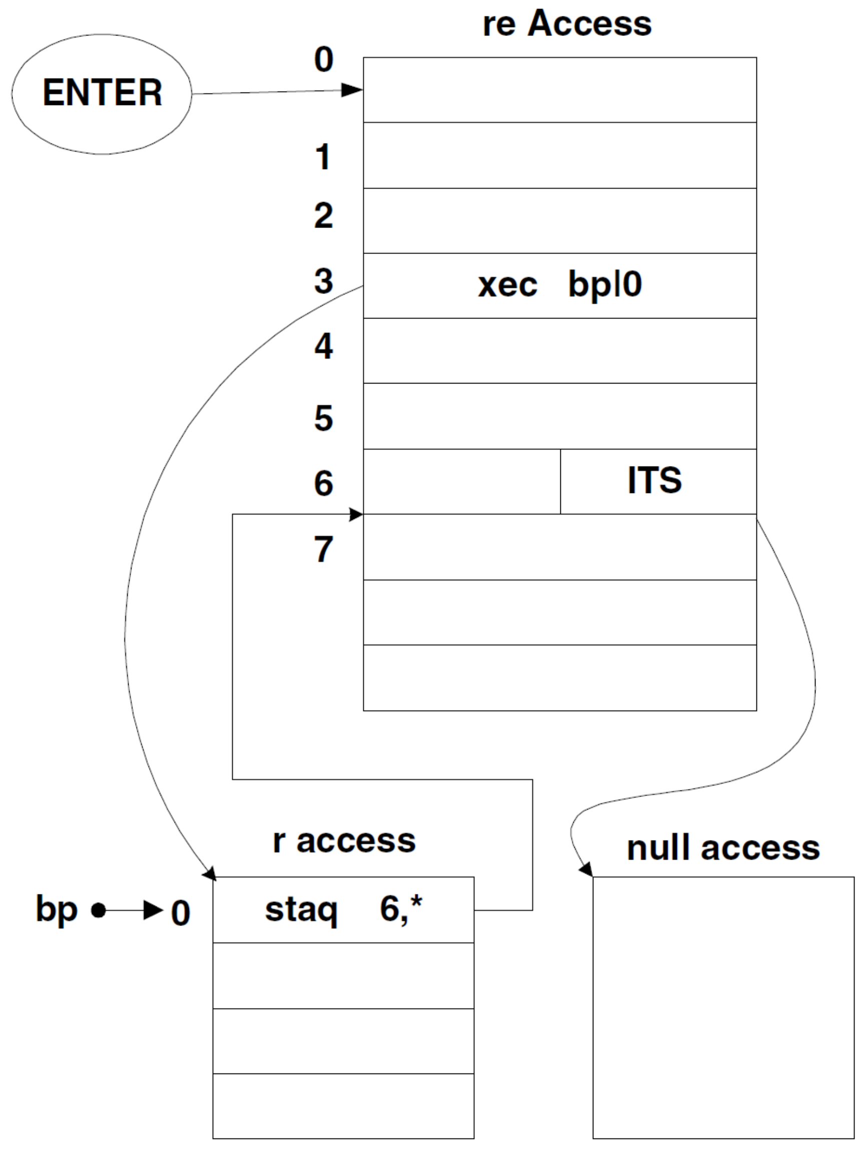


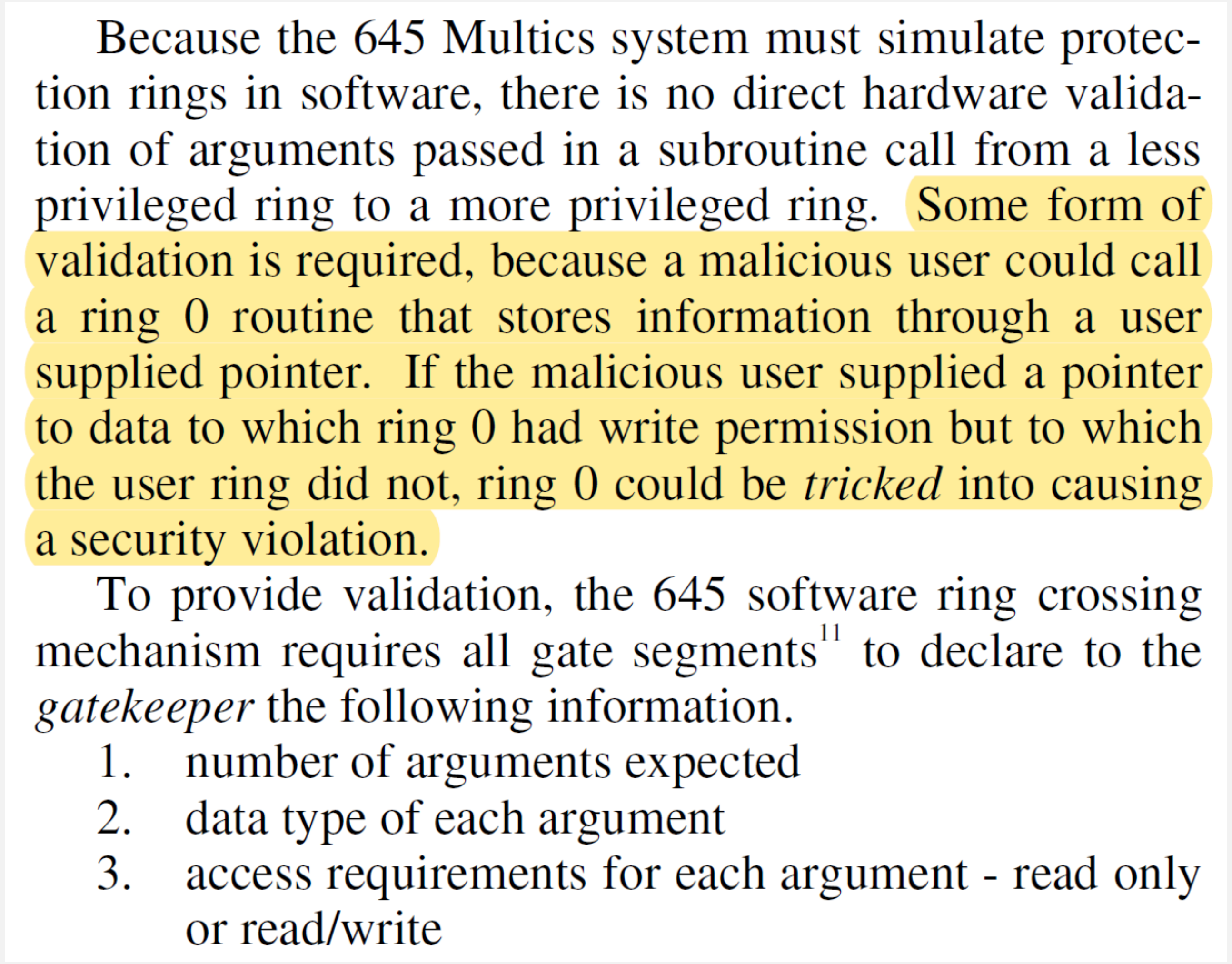
Figure 4. Execute Instruction Bypass

This hardware bug represents a violation of one of the most fundamental rules of the Multics design - the checking of *every* reference to a segment by the hardware. This bug was not caused by fundamental design problems. Rather, it was caused by carelessness by the hardware engineering personnel.

DETOUR: FUZZING

- The reliability tester was, unintentionally, a fuzzer
- That is, it ran a bunch of unusual code sequences automatically
- Although it was looking for ***reliability*** failures, it found a vulnerability

RING-0 TRICKED

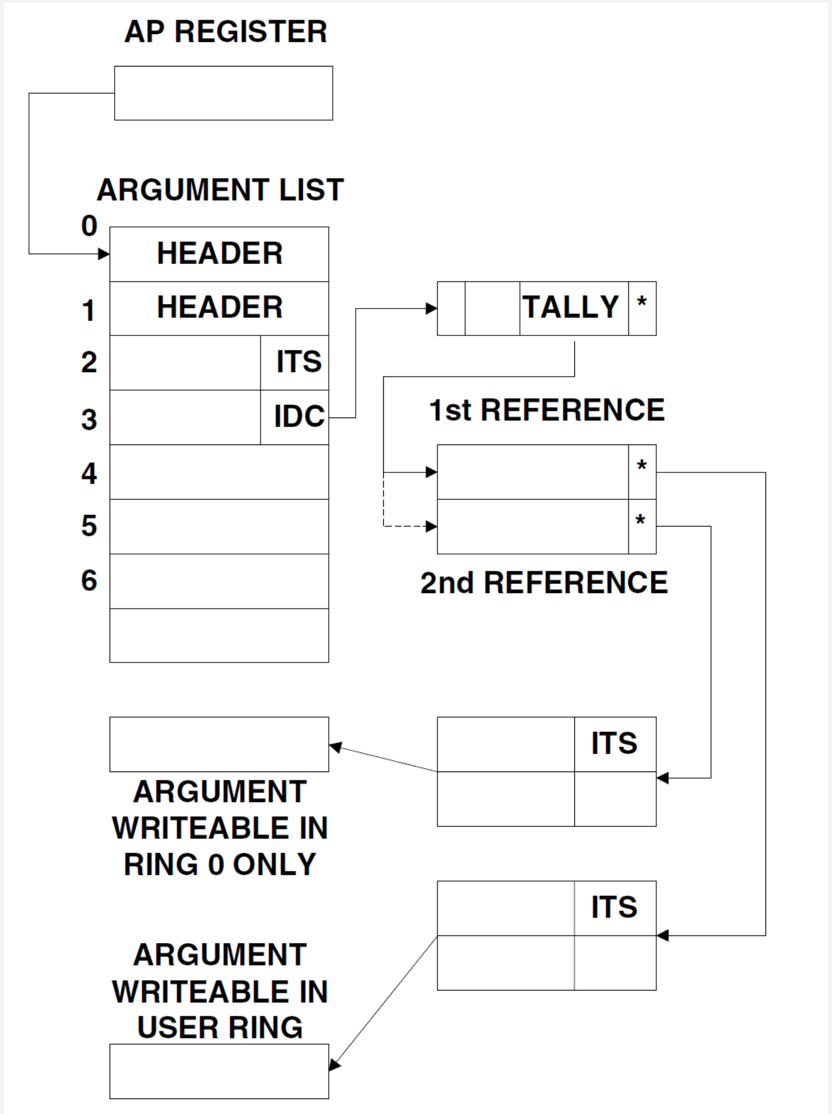


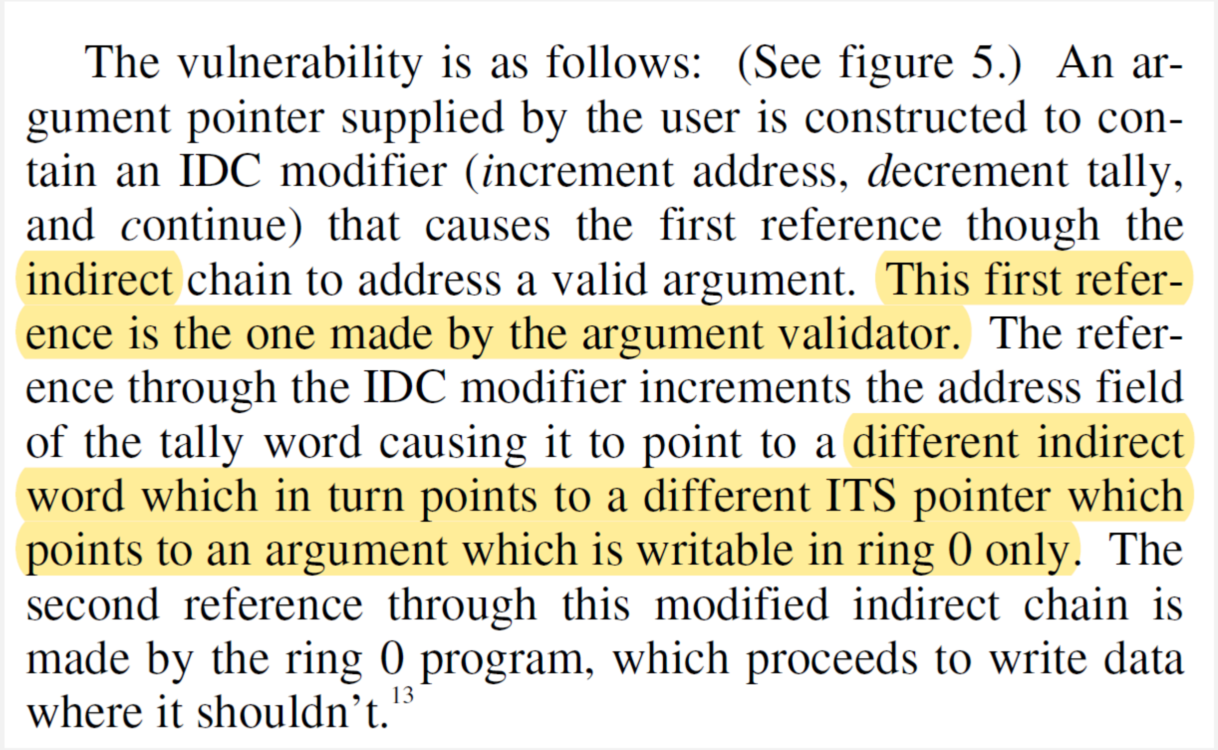
Because the 645 Multics system must simulate protection rings in software, there is no direct hardware validation of arguments passed in a subroutine call from a less privileged ring to a more privileged ring. Some form of validation is required, because a malicious user could call a ring 0 routine that stores information through a user supplied pointer. If the malicious user supplied a pointer to data to which ring 0 had write permission but to which the user ring did not, ring 0 could be *tricked* into causing a security violation.

To provide validation, the 645 software ring crossing mechanism requires all gate segments[11] to declare to the *gatekeeper* the following information.

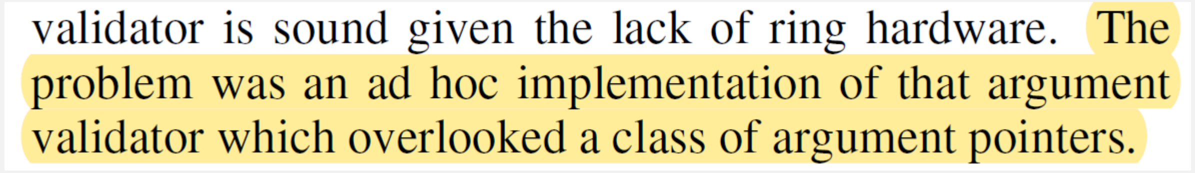
1. number of arguments expected
2. data type of each argument
3. access requirements for each argument - read only or read/write

VIOLATING RING-0





The vulnerability is as follows: (See figure 5.) An argument pointer supplied by the user is constructed to contain an IDC modifier (*i*ncrement address, *d*ecrement tally, and *c*ontinue) that causes the first reference though the indirect chain to address a valid argument. This first reference is the one made by the argument validator. The reference through the IDC modifier increments the address field of the tally word causing it to point to a different indirect word which in turn points to a different ITS pointer which points to an argument which is writable in ring 0 only. The second reference through this modified indirect chain is made by the ring 0 program, which proceeds to write data where it shouldn't.[13]



validator is sound given the lack of ring hardware. The problem was an ad hoc implementation of that argument validator which overlooked a class of argument pointers.

SOMETHING SEEMS FAMILIAR…

- Isn’t this the second vulnerability via ITS?

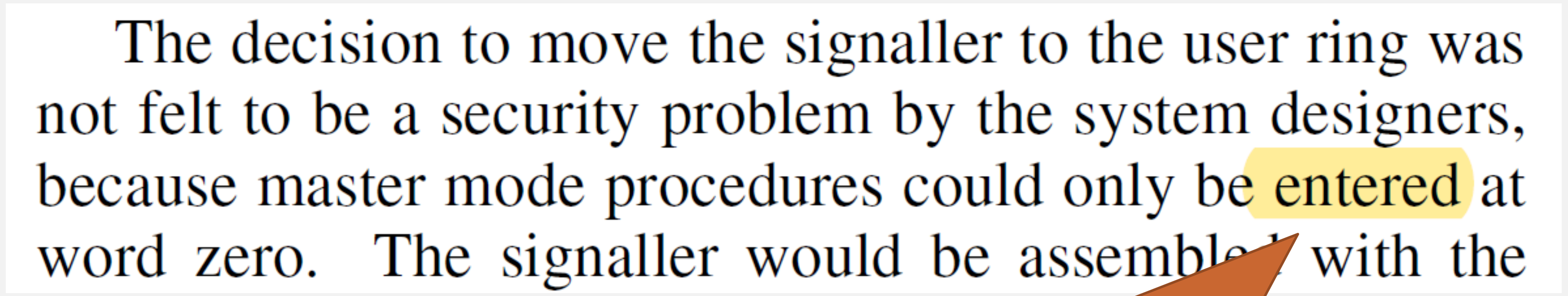
BREAKING INTO MASTER MODE

As stated previously, master mode should be ring-0 only

However, fault handling, which is master mode, was expensive

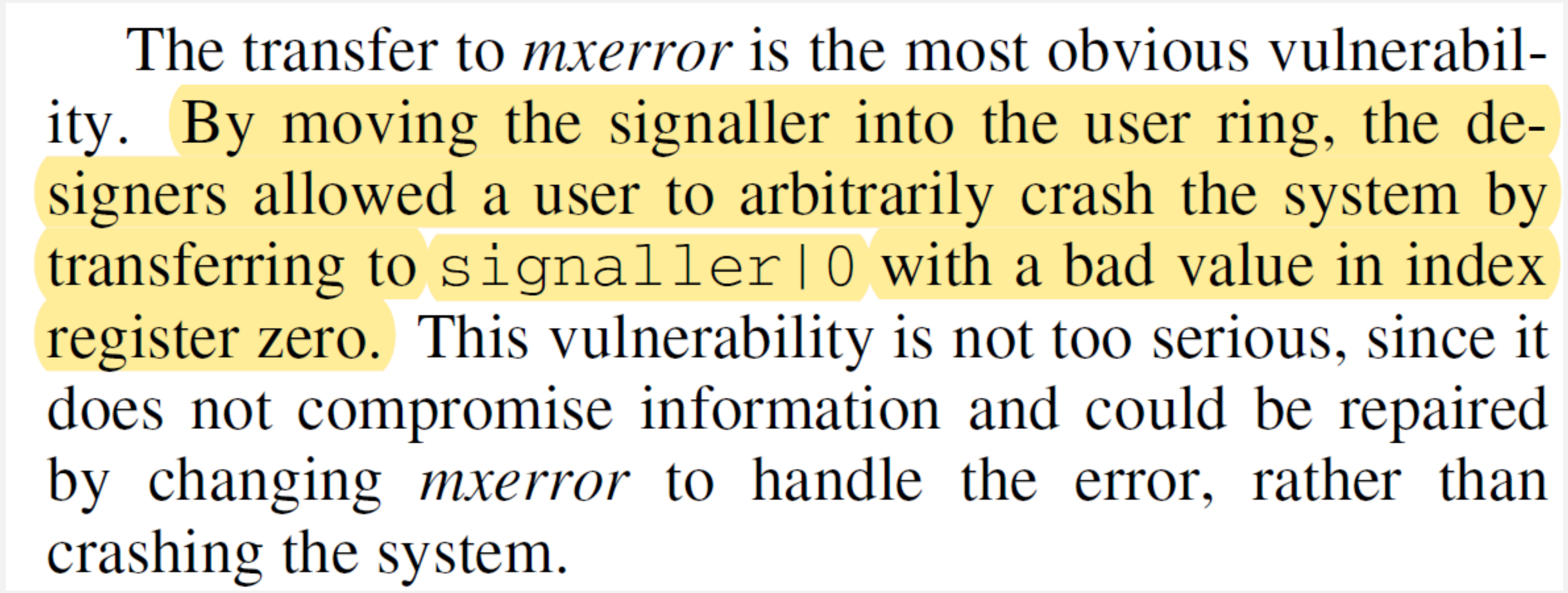
A *signaler*, which was master mode, was moved to the user ring

SECURITY RATIONALE

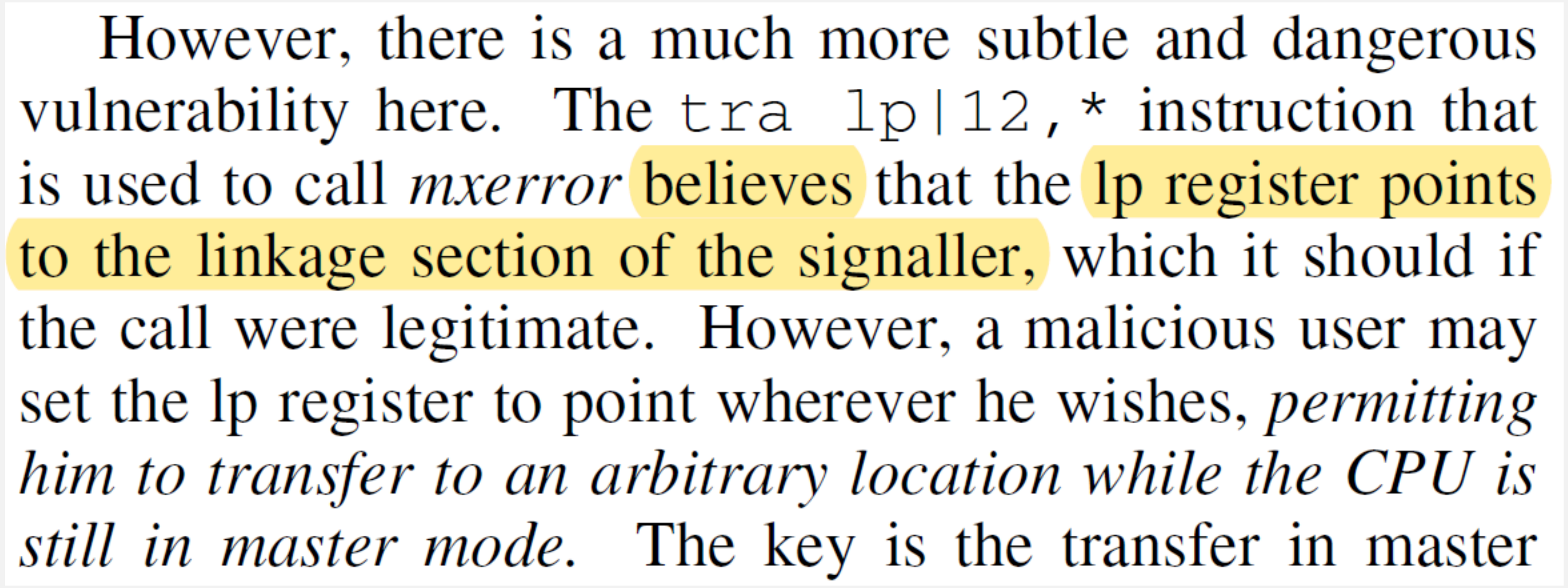

The decision to move the signaller to the user ring was not felt to be a security problem by the system designers, because master mode procedures could only be entered at word zero. The signaller would be assembled with the

In other words, so long as it <u>started</u> in ring-0, they were sure it was fine.

VULNERABILITY 1: CRASHING

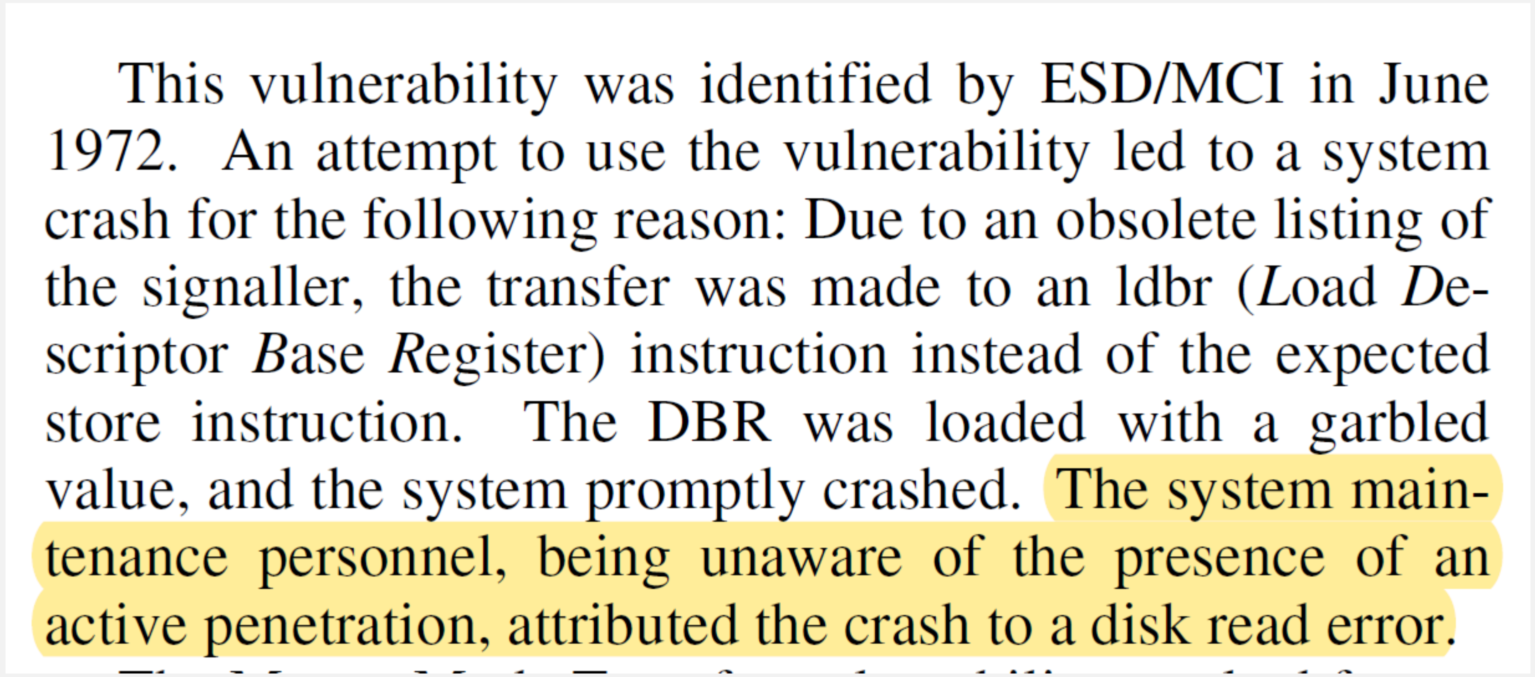

The transfer to *mxerror* is the most obvious vulnerability. By moving the signaller into the user ring, the designers allowed a user to arbitrarily crash the system by transferring to `signaller|0` with a bad value in index register zero. This vulnerability is not too serious, since it does not compromise information and could be repaired by changing *mxerror* to handle the error, rather than crashing the system.

VULNERABILITY 2: ARBITRARY CODE



However, there is a much more subtle and dangerous vulnerability here. The `tra lp|12,*` instruction that is used to call *mxerror* believes that the lp register points to the linkage section of the signaller, which it should if the call were legitimate. However, a malicious user may set the lp register to point wherever he wishes, *permitting him to transfer to an arbitrary location while the CPU is still in master mode*. The key is the transfer in master

ASSUMPTIONS STILL


This vulnerability was identified by ESD/MCI in June 1972. An attempt to use the vulnerability led to a system crash for the following reason: Due to an obsolete listing of the signaller, the transfer was made to an ldbr (*L*oad *D*escriptor *B*ase *R*egister) instruction instead of the expected store instruction. The DBR was loaded with a garbled value, and the system promptly crashed. The system maintenance personnel, being unaware of the presence of an active penetration, attributed the crash to a disk read error.

UNLOCKED STACK BASE

- Very briefly discuss here
- Change in security design assumptions
- Never fully updated the implementation to the new design

DUMP AND PATCH

Multics can re-write all of memory while it is running

(BTW, if that sounds bad, all modern O/Ss can too, including device drivers)

These utilities are supposed to be ring-0 only

Using the previously described vulnerabilities, were run in user ring

OTHER VULNERABILITIES

- Using Dump and Patch, it is possible to become "anyone" (forged ID)
- Access to the password file through forged ID
- Modifying Audit trails, through forged ID
- Insertion of Trap Doors, using forged ID

ONE ATTACK BUILDS TO ANOTHER

- Unlocked Stack Base plus Insufficient Argument Validation…
- Leads to Dump and Patch…
- Leads to Forged ID…
- Leads to password file access, trojans, etc

SUMMARY

We did not have time to review every attack in deep detail

We focused on certain attacks if they were difficult to understand

Or if they had important pedagogical value

Or if they are related to contemporary security concerns