Malware Tolerance

Report 2

Michael Denzel <m.denzel@cs.bham.ac.uk>

March 16, 2015

Abstract

"Malware tolerance" is a term we invented to describe a technique which accepts that one or several devices are infected with malware, but, nevertheless, uses those malicious systems for security sensitive functions in a secure way.

In brief, this research aims to answer the question if we can use a malicious system securely.

To achieve this, Trusted Execution Environements as well as the three scenarios e-mail, Bitcoin, and secure storage will be examined.

Desired results are circumstances under which it is possible to securely use a platform, that is suspected to have malware, for a chosen scenario.

key-words: Malware Tolerance, Trusted Computing, Trusted Execution Environment (TEE), IT Security

CONTENTS

List of Tables

List of Abbreviations

CA	Certificate Authority
CVE	Common Vulnerabilities and Exposures
IDS	Intrusion Detection System
IPT	Identity Protection Technology
IT	Information Technology
LED	Light-Emitting Diode
MITM	Man-in-the-Middle
NFC	Near-Field Communication
NVD	National Vulnerability Database
OS	Operating System
PAVP	Protected Audio and Video Path
PC	Personal Computer
PGP	Pretty Good Privacy
Ph.D.	Doctor of Philosophy
SD	Secure Digital
SGX	Software Guard Extensions
S/MIME	Secure/Multipurpose Internet Mail Extensions
SMM	System Management Mode
TAN	Transaction Authentication Number
TCB	Trusted Computing Base
TEE	Trusted Execution Environment
TPM	Trusted Platform Module
TXT	Trusted eXecution Technology
USB	Universal Serial Bus
VM	Virtual Machine

1 Introduction

We propose the new approach **malware tolerance**. The idea behind it is to accept that certain systems are possibly infected, but find a way to use these computers anyway. This is important when we are not in control of the computer (a bank has no control over its customer's computers) or when we are not aware that the computer is even infected (e.g. Stuxnet [27]).

All in all, malware tolerance aims to **use (possibly) malicious computer(s) for security sensitive functions in a secure way**.

Previous research focused on defending systems, assuming that they are malware-free or the hardware is trustworthy. Both assumptions are not feasible in real-world scenarios. Considering online banking, from the bank's point of view the assumption that the client's computer is malware-free does not hold.

Supplementary, online banking was improved by a second channel - e.g. physical Transaction Authentication Number (TAN) lists, SMS, or banking tokens. This second channel enables bank and customer to skip possible malware on the customer's computer.

Electronic voting picked these existing techniques up and expanded them to tolerate attacks on multiple devices.

A problem, however, is the usability, since a banking token is not feasible for frequent tasks like e-mails.

In summary, only two setups exist to tolerate malware while most others assume malware-free systems.

1.1 Purpose and Objectives

The overall goal of this thesis is to examine the whole computer architecture (including phones etc.) in order to tolerate malware on systems. A first overview showed that hardware support is needed, because software is relying on the hardware beneath it.

We chose three scenarios, secure storage, e-mail, and Bitcoin, to examine regarding malware tolerance.

As the overall aim is optimistic for a Ph.D. thesis, we split it into different steps to improve the situation of secure computing. Secure computing is seen in general here and not only as Trusted Execution Environements (TEE's). Chapter 3.3 gives further details about the steps.

1.2 Evidence

The aims seem to be ambitious but there is evidence:

Analogy: Biased coin

A good example for the basic idea is a biased coin with unknown probability $p \neq 0.5$. The question is whether we can still make it a "fair" coin. This exercise appears unsolvable, but flipping the coin multiple times creates event sequences that have the same probability. In conclusion, we can wrap the biased coin to "remove" the bias.

This example showed that a protocols, or software, can "skip" manipulations. For an entire computer this becomes more challenging.

Evidence in Research

There are scenarios, as for example online banking, which are considered as already solved. A security token ensures that the computer cannot alter transactions. Thus, malware is tolerated to a certain degree on the computer.

Further scenarios include electronic tax return, as already implemented by the German government [14], and online voting, as shown by Du-Vote [44, 21] (we give details in Chapter 2 "Literature Review").

1.3 Brief Overview of Progress

To this point, the thesis was concerned with researching possible topics, deciding between them, exploring malware tolerance as the chosen topic, and narrowing it down to a special area to research.

For exploring the topic, literature was reviewed, an overview of malware, attacks, and defences was created, and vulnerability classifications were researched.

Furthermore, critical scenarios were classified for their security demands (see Table 2 and 3) and we searched for additional devices, which could help to solve the scenarios.

At the moment, the work is focusing on TEE's, integrated hardware offering protection mechanisms, as those could be the basis for malware tolerant systems. It turned out that those are not as secure as expected [57, 37, 52, 34] and we, thus, strive for further examining this.

Additionally, we aim to examine the "storage issue" - how to retrieve and work with secret data - under two assumptions (see Chapter 3.3).

The next milestone will be to detail goals in the thesis proposal and work specifically towards those.

2 Literature Review

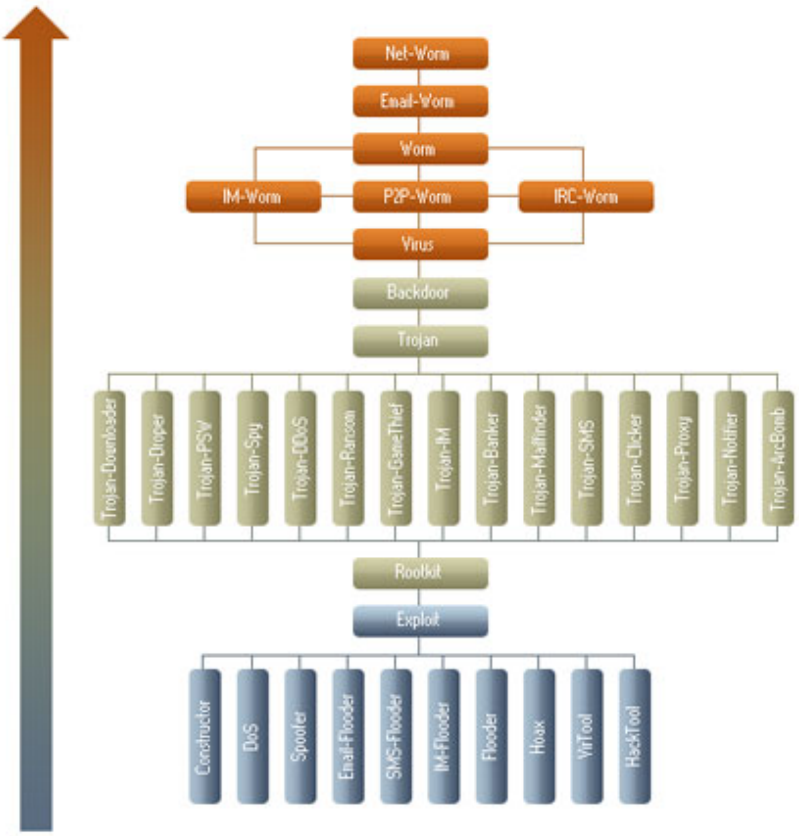
Since the term “malware tolerance” did not exist previously, there are only a few topics which are connected to malware tolerance.

This chapter will, first, give an overview of current attacks, then explain defence primitives, and conclude with examples of systems effectively tolerating malware.

2.1 Background: Attacks

To tolerate malware, it is important to know about **attacks** and **vulnerabilities**. A basic overview of malware is shown in Fig. 1. [23]

Fig. 1: Kaspersky: Types of Malware



Databases like National Vulnerability Database (NVD) and Common Vulnerabilities and Exposures (CVE), which collect and classify attacks and vulnerabilities, are important for research. A few projects [39, 40, 50] focused on retrieving infor-

mation of those to identify security problems of the current computer architecture.

2.2 Primitives

Primitives are basic techniques useful for defending malware. We clustered them into three sections: Software, integrated hardware, and standalone hardware.

2.2.1 Integrated Hardware

One technique for achieving malware tolerance is **Trusted Execution Environment (TEE)**. A TEE is a part of a platform that offers various security-properties to its customers, usually through a secured boot-process and authenticated software. TEE's are suitable as a basis for malware tolerant systems.

Vasudevan et al. [57] defined five **security properties** a mobile device needs to be trustworthy, but those properties are directly transferable to TEE's in general.

1. Isolated Execution

   A module's code and data has to stay intact (Integrity) and secret (Privacy). Usually, an Operating System (OS) provides these properties, but a compromised OS is not trustworthy. Therefore, the OS has to be verified or isolation has to be provided from a different source.

2. Secure Storage

   When the system is powered off, the secrecy and integrity of the data has to be guaranteed. This is usually implemented by encrypted memory.

3. Remote Attestation

   When receiving a message from a trusted machine, the remote must be able to verify the origin of the message. In this case, the sending system must attest to the entire (booted) Trusted Computing Base (TCB).

   Remark: This does not detect run-time exploits.

4. Secure Provisioning

   To synchronise data between multiple devices, data has to be send to a specific module on a certain machine. Secure provisioning usually authenticates a public key through remote attestation.

5. Trusted Path

   In order to communicate with users, the peripherals and the lines ("path") to them have to be authenticated and trusted. Additionally, secrecy could be ensured for enhanced security.

Murdoch [37] utilises this five point schema of Vasudevan et al. [57] to compare well-known TEE's (see Fig. 2).

Fig. 2: Comparison Trusted Execution Environments

	Isolated Execution	Secure Storage	Remote Attestation	Secure Provisioning	Trusted Path
TPM	Not really (too limited)	Yes (but very limited)	Yes	Yes	No (easily bypassed)
Flicker	Yes (but no drivers)	Yes (though TPM)	Yes (through TPM)	Yes (through TPM)	Limited
Trustonic/ KNOX	Yes (but restricted)	Yes	Yes	Yes	Somewhat
IPT	Yes (but restricted)	Yes	Yes	Yes	Somewhat
SGX	Yes	Yes	Yes	Yes	Probably

The first large-scale approach of hardware to support trusted execution was the **Trusted Platform Module (TPM) chip** which was used in a wide range of techniques. [9, 17, 24, 28, 30, 46, 53, 54]

E.g. **Flicker** [33] used the TPM to create a trusted zone with extensive defensive measures to protect against malware.

Following this development, Intel expanded their processors with a row of features, with the most important one for this research being Software Guard Extensions (SGX), a TEE like the TPM.

- **System Management Mode (SMM)** [13]
- **Identity Protection Technology (IPT)** [37]
- **Trusted eXecution Technology (TXT)** [33]
- **Protected Audio and Video Path (PAVP)** [22, 36]
- **Software Guard Extensions (SGX)** [1, 2, 35, 38, 41, 52]

ARM created a technology called **TrustZone** [4, 3] of which a joint project, named **Trustonic** [55], evolved between ARM, Giesecke & Devrient, and Trusted Logic Mobility.

Recently, Samsung released **Knox** [49, 47, 48] which builds on TrustZone and runs on Android smart-phones.

2.2.2 Software

In contrast to TEE's, **sandboxes** are an "untrusted environment" to lock malware in. A sandbox can only defend against attacks from the inside of the sandbox and

is, thus, limited. An attacker controlling hardware, operating system, or with direct access to the computer has an advantage and can usually not be controlled by a sandbox. [10, 31, 42, 62, 61, 29]

Virtual Machines (VMs) [17, 1] and **hypervisors** [5, 54, 58] are similar to sandboxes but instead of restricting accesses to the system, an entire new system is created virtually. The actual system is transparent from the virtual ones hiding the real system from programs running inside the virtual environment.

Another field, that is related to malware tolerance is **intrusion detection**, as it has a subsection "intrusion tolerance". While intrusion detection only deals with identifying and handling attacks against computers and network, intrusion tolerance tries to harden the Intrusion Detection System (IDS) itself against attacks. [26, 15] This is achieved by using different operating systems and architectures, as well as redundancy in executing tasks, so called "Byzantine Fault-Tolerance". [59, 12]

Last software primitive worth mentioning is **obfuscation**, a technique where the actual program code hides secrets. While general-purpose virtual black-box obfuscation is impossible according to Garg et al. [18], they proved the theoretical model of "indistinguishable obfuscation". It is defined as "obfuscations of any two distinct (equal-size) programs that implement identical functionalities [are] computationally indistinguishable from each other". [7, 6, 18, 25, 45]

2.2.3 Standalone Hardware (Devices)

Devices themselves are mostly untrusted as seen on the example of computers and phones.

Trusted or partly trusted devices are usually embedded devices like **security tokens** from online banking, **smart cards**, special **Universal Serial Bus (USB) sticks** [51, 19], and **Secure Digital (SD) cards**. [20, 56]

Furthermore, the Near-Field Communication (NFC) technology offers new possibilities like NFC cards. [56]

2.3 Full System Approaches

Likely the most well-known working system, among the general population, is **online banking**. Even though it is not completely secure, latest techniques involving banking tokens, commonly with display, are trusted. These banking tokens enable users to tolerate malware on their Personal Computers (PCs).

But there are more systems to tolerate malware in some forms:

MP-Auth [32] hides long-term passwords from untrusted machines by employing mobile phones. Thus, the computer itself can be tunnelled and has no access to the data. This technique is not very secure, since Phones or PDAs connected to PCs

are vulnerable. [60] Nevertheless, it is a first step.

Boyens and Günther [8] focused on using homomorphic encryption to **hide data from servers**, while a client can still make simple requests. A disadvantage is, however, that complex requests are not possible.

Prior to Intel SGX, a similar technique utilising a special **compiler** was introduced. [1] The compiler creates separate modules with strong access control rules which are enforced by a small trusted hypervisor via monitoring of the program counter.

Recently, the German government introduced an **electronic tax return** procedure involving asymmetric cryptography with pre-shared keys. [14] Security is offered in three different levels: In software (weak), on a USB stick, and on a smart card. Users can produce their tax return form locally on their computer and send it securely encrypted to the tax office.

Ultimately, Du-Vote [44, 21] shows that an untrusted system can securely process **electronic voting**. It uses a banking token to split the protocol between a server with a bulletin board, the client computer, the voter, and the hardware token. This way, manipulations are identifiable even when an attacker controls multiple devices.

3 Model and Methodology

In this section, we introduce our assumptions, develop research questions and formulate steps to work on those.

3.1 Attacker Model

It is not realistic to assume the attacker to be all mighty, since there is no attacker controlling everything. Nevertheless, defensive measures have to adapt to the attacker and our attacker model has to reflect this.

For this purpose, we combined the already existing models “honest-but-curious” and “malicious-but-cautious” [43] to a layered approach with four levels (Table 1). The level “malicious-if-anonymous” was introduced by us.

All parties can be classified into one of these levels.

Table 1: Attacker Model

Level	Name	Parties
1	malicious-if-anonymous	unknown attackers, governments (do not need to be cautious)
2	malicious-but-cautious	known attackers (e.g. companies), governments (if they have to)
3	honest-but-curious	open source projects (or otherwise securely controlled)
4	honest	users (regarding own data)

Anonymous attackers can be assumed to be malicious. As long as attackers stay anonymous there is no reason to withdraw from attacks (malicious-if-anonymous). If anonymous attackers become known, they will be more cautious (malicious-but-cautious) and, thus, switch the level in our model.

Considering e.g. a service provider, if a business is known to be malicious, no one will use it. That means, companies will only attack their users in a cautious fashion (malicious-but-cautious).

On the opposite, their production firms are more anonymous and are possibly more malicious. But, nevertheless, companies have quality controls and if a backdoor is found in their chips, they will trace it back to the source.

This solves the issue with classifying private attackers (malicious-if-anonymous) and malicious companies (malicious-but-cautious). Governments, however, do not fit into this model. They have to be assumed to be malicious-if-anonymous in our model, even though they are not anonymous. The reason is, that they do not need to be cautious, as recent leaks turned out.[1] When controlled by law, this can turn into malicious-but-cautious.

Another party to classify are open-source projects. Since their products are completely open for everyone, active attacks are not feasible. They are classified as honest-but-curious.

Lastly, the user is assumed honest regarding his own data and services. It is in one's own interest to keep personal data secure. A user attacking other data of the system, falls under the first two models, malicious-if-anonymous or malicious-but-cautious.

In the end, the model is more representative than "everyone is completely untrusted".

[1] `http://www.theguardian.com/world/2014/jul/18/-sp-edward-snowden-nsa-whistleblower-interview-transcript` (accessed: 2014-08-25)

3.2 Scenarios Involving Malware Tolerance

We identified ten scenarios in "daily" life that are security related and classified them using the following criteria:

- Confidentiality/Privacy
  - Authentication
  - Data control of the participants
- Integrity
- Availability/System Stability
- Verifiability of errors/manipulations
- Anonymity of the participants
- Usability
  - Frequency
- Security level (How critical is the scenario?)
  - Reversability of the process (undo, redo)
  - Value (How useful is it if secured?)

Some of these points might be subjective but we kept it as objective as possible. E.g. the point "security level" was linked to facts, like for example money. Online shopping is more critical than social networking since money is involved and one has the option to not share information on social media.

3.2.1 Definitions

The next paragraphs will explain a few uncertain classification-points.

System Properties

The **integrity** of a system describes whether the system is corrupted or not. A classical attack on integrity is altering the message body of a network packet in a Man-in-the-Middle (MITM) attack. The integrity of the message is corrupted. Its integrity could e.g. be secured by a signature.

When the parts of the system fit together, the system is in a "consistent" state (**consistency**).

Correctness means a system is working correctly. In contrast to consistency as a state, correctness describes a behaviour.

Lastly, **persistence** is connected to storing data and how long information lasts. E.g. deleting an entry in a database is invalidating its persistence.

We defined these properties here, because they are used in different terms. For the classification we focused on integrity as it is the most important one for security. For scenarios involving storage, like cloud storage, persistence is also essential.

Confidentiality and Data Control

While **confidentiality** deals with keeping data private and avoiding unintended leakage, **data control** expresses who is in control of the data.

For example, in social networking initially users are in charge of their data because they can decide if they share it with the network. Confidentiality is not given as social networks are usually not encrypted. After sharing, the social network provider is in control of the data.

Data control is a contested criterion. While users want to be in charge over their own data, service providers want to scan data to prevent illegal actions[2], as well as using the information for targeted advertisements.

3.2.2 Explanations of Three Examples

Before dealing with the other examples, we will start with three important ones. These examples are secure storage, secure e-mail, and Bitcoin.

Secure Storage

Secure storage is very important since it is connected to multiple scenarios. Most common are cloud services which usually exclusively provide storage space. This includes local clouds like OwnCloud. Secure clouds, also called zero-knowledge,[3] already exist and protect data online but the problem is the own device. Once the password is stolen, full access to the cloud is granted.

Furthermore, data-loss through forgotten passwords arises as new obstacle. An additional device could help recovering the password, by for example giving a password hint or giving a one-time-approach to decrypt the data.

E-mail

E-mail, or communication in general, is used by everyone and private messages are important to contact family, friends, companies, colleges, banks, doctors, etc.

PGP and S/MIME already offer encryption for e-mails but they are not widely used and rely on trusted third parties, namely web-of-trust and Certificate Authorities (CA's), to exchange keys.

In addition, locally installed malware has full access to these messages. Consequently, they have to be stored securely.

[2] see e.g. `http://fortune.com/2013/03/05/much-like-google-microsoft-also-scans-your-email` (accessed 2014-08-25)

[3] `http://zeroknowledgeprivacy.org/` (accessed 2014-09-08)

Bitcoin

The main problem of money in general is that it is lost or stolen. This also holds for online currencies as Bitcoin. For computers, there are two opposed solutions to these issues: Encryption and backups. While backups offer protection from loss, they represent an additional point for attackers. In conclusion, Bitcoin would also benefit from secure storage.

3.2.3 Full Classification

After introducing the three most important scenarios for this research, we will present the remaining results of our classification. They are summarised in Table 2 and 3.

While we consider online banking, e-voting, and e-tax-return mostly as solved (see Chapter 2.3), the rest is still problematic. We will explain these scenarios in the following.

Online shopping is similar to online banking and could adapt the same techniques. From a research perspective, this scenario is, therefore, not very interesting.

At the moment, social networking requires that users entrust their data to the social network provider. That means there are only two states, publish information or keep it secret, since the data is not secure and cannot be taken back. An approach similar to zero-knowledge clouds is desirable, linking this scenario to secure storage.

With service providers and others collecting more and more information about us[4], anonymity is an issue. TOR deals with this, but there are attacks of the so-called "exit nodes" of the network. [11, 16] Encryption between the web-server and the client helps to reduce these attacks, but the abilities of the web-server are beyond the users possibilities.

Smart devices as smartphones, smart TVs, smart watches, etc. are becoming increasingly popular. Apart from entertainment, there are also task-based devices like smart door locks and electronics in medicine. For example, some pace makers or cochlea-implants have a Bluetooth interface. While easy access for clinical staff is positive, everyone else should not be able to access a pace maker. Security is a trade-off: Doctors typing in a password or finding the smart card before accessing one's pace maker is a crucial point.

[4] e.g. `http://fortune.com/2013/03/05/much-like-google-microsoft-also-scans-your-email` (accessed 2014-08-25)

Table 2: Critical Scenarios Summary (Part 1)

Scenario	Main Problems	Hardware Support
1. Online banking	Authentication, Integrity	Additional device with display shows transaction and enables user to verify it.
2. E-voting	Confidentiality, Anonymity	Device hashes vote and hides it from client-computer and server.
3. E-tax-return	Integrity, Authentication, Confidentiality	Device just stores the (authenticated) asymmetric key securely (USB-Stick, Smartcard).
4. E-mail / Communication	Integrity, Confidentiality, Usability	• Encryption already works • Token could store keys • Problems: Exchanging keys, usability
5. Online shopping	Authentication, Confidentiality	Bank transaction could only be valid with correct approval through additional device.
6. Social networking	Authentication, Confidentiality	• Login-credentials stored on additional device • Zero-knowledge cloud for service • “Friendships” have to be acknowledged • Problems: Usability
7. Web surfing / TOR	Integrity, Anonymity	• Device could help with encryption • Problem: Usability • TOR: Attacks mainly by MITM of exit node • Problem: Depends on the web-servers

Table 3: Critical Scenarios Summary (Part 2)

Scenario	Main Problems	Hardware Support
8. Cloud access	Confidentiality	• Stolen data → Encryption • Precondition: Zero-Knowledge Cloud • see 8b. Storage
8b. Storage	Confidentiality, Usability, Integrity	• Lost data → Backup • Lost password is the main problem • Device: Help recovering the password
9. Pervasive computing: Smart devices	a. Entertainment: Data control b. Task-based: Stability, Authentication	a. Undesired, any second device reduces usability b. Trade-off between speed and security, especially for health-applications speed is desired
10. Bitcoin	Integrity (of log), Anonymity	• Stolen/lost Bitcoins are the main problem • Stolen → Encryption • Lost → Backup • Device has to help with these issues • see also 8b. Storage

3.3 Steps

In order to achieve the far-reaching goal of tolerating malware, we defined a few steps. These steps represent "sub-objectives" describing how to approach the overall aim.

1. Define and classify scenarios
2. Explore techniques and devices that are related to this issue

3. Analyse TEE's:

   TEE's suffer from certain disadvantages. A notable one is the problem of how users can verify if the system is running in secure-mode.

   A software mechanism to display this does not work, since malware is able to mimic this behaviour. A brief investigation showed that TrustZone and Samsung Knox apparently did not implement any means to deal with this problem. We suspect that further TEE's miss fixes, too.

   This leaves systems built on these TEE's vulnerable to phishing-like attacks.

   Before building a system based on these TEE's, this has to be considered. If applicable, further research should be carried out to attack and solve this issue.
4. Analyse the scenarios (E-mail, Cloud/Storage, Bitcoin) under certain assumptions:

   Correctly used encryption works fine but one of the main problems is how to get data encrypted and how to retrieve it securely. Suppose someone is sending us an encrypted e-mail. The information carried by the e-mail is safe during transfer, but decrypting and reading it, e.g. at work, is insecure since an administrator has full access to the computer.

   When logging into our e-mail account, the password may be compromised and all plaintext data is leaked, as well as cyphertexts.

   Data should be organised in a layered approach to only expose accessed data to possibly malicious devices. A zero-knowledge cloud or storage is helpful, but has the disadvantage of data loss if the password is not available any more. While a backup offers a good solution to most systems, this is a trade-off to security.

   Malware tolerance should protect the data while offering a fail-safe mechanism to retrieve it again. This mechanism is allowed to be more complicated considering that it happens less frequently.

   The following steps tackles the issue.

   (a) Assuming we cannot protect currently displayed data (because malicious screens could always capture this):

       Research ways to secure data which is not displayed at the moment. For example, when displaying an e-mail, this e-mail is compromised but all other e-mails do not necessarily have to be leaked. For cloud applications, the non-accessed data must stay encrypted.

       Analysing and, if possible, building (= programming) or designing (= creating the theoretical concept) such a system is another goal of this thesis.

Likely, the existing architecture has to be modified or expanded. Imaginable are pure hardware buttons, LEDs, NFC-channels and -tokens, etc.

- Examine the "layered storage approach"

  To protect unused data the system has to be built in a layered or hierarchical fashion.

  Search for ways to realise this system.
- Explore ways to identify attacks even when the device is malicious

  If the data is encrypted in layers, the device could respond to attacks by deleting the encryption key or limiting the requests to the server. Therefore, attacks should be identifiable.

(b) Assuming all necessary techniques (trustworthy own software, additional hardware, trustworthy hardware support like SGX) are available:

Examine existing architectures in order to deliver an output securely to the user when all other parts of the device are untrusted. E.g. search for a way to securely display an e-mail on a phone which has a malicious root-app installed. Trusted parts here are, in any case, the decryption routine and the screen which displays the unencrypted output.

Intel PAVP [36] is the only approach known to us that deals with securely displaying data. Further research on its value for malware tolerance will take place.

If displaying securely turns out to be impossible, give details on the restrictions or describe the missing features which are needed to securely display the output if applicable.

3.4 Timeline

The next steps will be to research TEE's and their possibilities. If the results show gaps in their architecture, research should concentrate on this. Parallel to the first step, circumstances for secure storage have to be examined.

The following graphic (Table 4) gives an overview of the planned activities. The labels “A:” and “B:” stand for different plans and represent alternatives. Grey content is not intended to take place and serves as fall-back.

Table 4: Milestones

2014-03-14	Start
2014-05-19	Report 1
2014-09-20	Report 2
2014-10	Research TEE’s
2014-12	Explore “Storage” under the two assumptions; State goals more precisely
2015-02-06	Report 3: Thesis proposal
2015-02	A: Attack and resolve TEE-issues (Item 3; might get extended) B: Develop a mechanism to only display needed information (Item 4a)
2015-05	Paper 1
2015-06	A: Use the techniques of the previous step to secure data if not displayed (Item 4a) B: Examine possibilities to deliver output securely on a malicious system (Item 4b)
2015-09-20	Report 4
2015-10	Link techniques to Cloud, Bitcoin, and E-mail
2015-12	Paper 2
2016-04-03	Report 5
2016-04	Further improve mechanisms and spread solutions to other scenarios
2016-07	Paper 3
2016-07	Begin writing the dissertation
2016-10-02	Report 6
2017-03-13	Planned submission
2017-04-09	Report 7
2017-10-08	Report 8
2018-03-13	Submission deadline

To expand my knowledge, I already participated in some workshops and conferences (Table 5).

Table 5: Workshops

Date	Workshop/Conference	Location
7th May 2014	Google Hack Day (Certificate Transparency)	Google London
19th–23rd May 2014	Academic Writing Seminar	University of Birmingham
27th–28th May 2014	CryptoForma 2014	University of York
14th–18th July 2014	Enterprise Summer School	University of Birmingham
23rd–24th July 2014	Publishing Academic Journals, Editing your writing	University of Birmingham

4 Assessment and Conclusion

This chapter estimates risks and moves on to possible results. It concludes with benefits for users under the assumption that these results are reached.

4.1 Challenges, Problems, and Comments

One restriction are the platforms themselves. Trust has to be built upon a TCB which can be implemented in software, on internal hardware, on an external device, or even as a combined approach. The interfaces of these platforms limit the system. E.g. it is not possible to display a complete e-mail on a banking token, even though this might be secure.

Another problem arises when distinguishing malicious inputs from user inputs. For example, the system needs to identify if it was the user who wants to decrypt the e-mail or if this was simulated by malware.

Lastly, the reverse has to be ensured, too. A user must be able to differentiate between the real system in secure mode and a phishing attempt of malware.

4.2 Possible Outcomes of the Research

Most likely, the result of this work will not be a bulletproof solution for tolerating malware on any device. This research will only be the first step and improve

the situation. Imaginable are solutions to specific, narrow scenarios under certain assumptions.

It is the aim of this thesis to generalise these assumptions and keep them as open as possible to find practical, usable solutions.

Possible is a layered approach for storage which reveals only data that is needed at the moment. This would affect all three scenarios: Storage of cloud, e-mails, and Bitcoins.

4.3 Benefits of this Research

The advantages of (perfect) malware tolerance are obvious: Even compromised devices stay usable and do not represent a threat, but this is unlikely during the short time of a Ph.D.

Hence, the structure of the objectives is defined by different, adjustable steps. Results of the first analysis will redefine the focus thereof.

E-mail

Communication media are an essential factor in everyone's life with e-mails being just one example. A solution to malware tolerant e-mail is likely portable to chat-clients. Encryption is already possible but is unsuccessful since it is too inconvenient and does not protect from attacks of the own device.

Displayed information is vulnerable. Even with secure software or a trusted path, attacks by the screen or "shoulder surfing" attacks cannot be defended. Secure storage could help by hiding unneeded data from the device and, therefore, also from third parties.

Bitcoin

Bitcoin was designed to operate without banks but, in fact, nearly no one would store all one's money in one's house. For Bitcoin, this means people do not store their Bitcoins on their own computer but use banks or just own a small amount of Bitcoins at home. With a secure, fail-safe storage location, Bitcoin is more attractive.

Storage/Cloud

Both scenarios above need to store data securely, protected from others and protected from loss. Therefore, secure storage, secure input, and secure retrieval (delivering data to the user) are the main issues. Processing the data is not as critical since e-mails are processed by reading or writing (= input/store) them, while Bitcoin already uses a Merkle-Tree to make all transactions public.

In summary, malware tolerant systems, as seen e.g. on the example of secure storage, would solve a lot of security issues. Consequently, this work will research malware tolerance.

References

[1] Pieter Agten, Raoul Strackx, Bart Jacobs, and Frank Piessens. Secure compilation to modern processors. In *Computer Security Foundations Symposium (CSF), 2012 IEEE 25th*, pages 171–185. IEEE, 2012. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6266159. accessed: 2014-07-24.

[2] Ittai Anati, Shay Gueron, Simon P Johnson, and Vincent R Scarlata. Innovative technology for cpu based attestation and sealing. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy, HASP*, volume 13, 2013. URL https://software.intel.com/en-us/articles/innovative-technology-for-cpu-based-attestation-and-sealing. accessed: 2014-04-17.

[3] ARM. *ARM Architecture Reference Manual ARMv8, for ARMv8-A architecture profile*. ARM Limited, 110 Fulbourn Road, Cambridge, England CB1 9NJ, v8 edition, December 2013. URL https://silver.arm.com/download/ARM_and_AMBA_Architecture/AR150-DA-70000-r0p0-00bet3/DDI0487A_b_armv8_arm.pdf. accessed: 2014-06-24.

[4] ARM. Arm trustzone website. online, 2014. URL http://www.arm.com/products/processors/technologies/trustzone/index.php?tab=Hardware+Architecture. accessed: 2014-06-24.

[5] Don Bailey. Uefi hypervisors - winning the race to bare metal. In *Black Hat Conference Proceedings*, Herndon, VA 20170 USA, 2008. Hypervista Technologies. URL http://sebug.net/paper/Meeting-Documents/BlackHat-USA2008/BH_US_08_Bailey_Winning_the_Race_to_Bare_Metal_White_Paper.pdf. accessed: 2014-06-03.

[6] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *Advances in Cryptology—CRYPTO 2001*, pages 1–18. Springer, 2001. URL http://link.springer.com/chapter/10.1007%2F3-540-44647-8_1. accessed: 2014-04-02.

[7] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. Technical report, Cryptology ePrint Archive, February 2014. URL http://eprint.iacr.org/2013/631.pdf. accessed: 2014-04-02.

[8] Claus Boyens and Oliver Günther. Using online services in untrusted environments: a privacy-preserving architecture. In *ECIS*, pages 239–250, 2003. URL sdaw.info/asp/aspecis/20040017.pdf. accessed: 2014-04-07.

[9] David Champagne and Ruby B Lee. Scalable architectural support for trusted software. In *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*, pages 1–12. IEEE, 2010. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5416657. accessed: 2014-07-24.

[10] Fangzhe Chang, Ayal Itzkovitz, and Vijay Karamcheti. User-level resource - constrained sandboxing. In *Proceedings of the 4th USENIX Windows Systems Symposium*, volume 91, 2000. URL https://www.usenix.org/legacy/events/usenix-win2000/full_papers/chang/chang.pdf. accessed: 2014-04-17.

[11] A Christensen et al. Practical onion hacking. Technical report, FortConsult (October 2006), 2006. URL http://www.wired.com/images_blogs/threatlevel/files/Practical_Onion_Hacking.pdf. accessed: 2014-08-11.

[12] Yves Deswarte, Laurent Blain, and J-C Fabre. Intrusion tolerance in distributed computing systems. In *Research in Security and Privacy, 1991. Proceedings., 1991 IEEE Computer Society Symposium on*, pages 110–121. IEEE, 1991. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=130780. accessed: 2014-04-08.

[13] Luic Duflot, D Etiemble, and O Grumelard. Security issues related to pentium system management mode. In *Cansecwest security conference Core06*, 2006. URL http://www.ssi.gouv.fr/archive/fr/sciences/fichiers/lti/cansecwest2006-duflot.pdf. accessed: 2014-09-03.

[14] ELSTER. Information zu den sicherheitsverfahren (information about the security [of elster]). online, 2014. URL https://www.elsteronline.de/eportal/Sicherheit.tax. accessed: 2014-08-25.

[15] Massimo Ficco and Massimiliano Rak. Intrusion tolerance of stealth dos attacks to web services. In *Information Security and Privacy Research*, pages 579–584. Springer, 2012. URL http://link.springer.com/chapter/10.1007/978-3-642-30436-1_52. accessed: 2014-04-08.

[16] Gregory Fleischer. Attacking tor at the application layer. *Presentation at DEFCON*, 17:54, 2009. URL https://www.defcon.org/images/defcon-17/dc-17-presentations/defcon-17-gregory_fleischer-attacking_tor.pdf. accessed: 2014-08-11.

[17] Tal Garfinkel, Ben Pfaff, Jim Chow, Mendel Rosenblum, and Dan Boneh. Terra: A virtual machine-based platform for trusted computing. In *ACM SIGOPS Operating Systems Review*, volume 37, pages 193–206, 2003. URL http://dl.acm.org/citation.cfm?doid=1165389.945464. accessed: 2014-04-17.

[18] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium*, pages 40–49. IEEE, July 2013. URL http://eprint.iacr.org/2013/451.pdf. accessed: 2014-04-02.

[19] Giesecke And Devrient. Starsign crypto usb token. Technical report, Giesecke And Devrient, July 2014. URL http://www.gi-de.com/gd_media/media/en/documents/brochures/mobile_security_2/nb/StarSign_Crypto_USB_Token.pdf. accessed: 2014-08-21.

[20] Giesecke And Devrient. Starsign mobile security card se 1.2. Technical report, Giesecke And Devrient, April 2014. URL http://www.gi-de.com/gd_media/media/documents/brochures/mobile_security_2/nb/StarSign_Mobile_Security_Card_SE_12.pdf. accessed: 2014-08-21.

[21] G. S. Grewal, M. D. Ryan, L. Chen, and M. R. Clarkson. Du-vote: Remote electronic voting with untrusted computers. Technical report, University of Birmingham, 2014. paper draft.

[22] Matthew Hoekstra, Reshma Lal, Pradeep Pappachan, Vinay Phegade, and Juan Del Cuvillo. Using innovative instructions to create trustworthy software solutions. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy, HASP*, volume 13, 2013. URL https://software.intel.com/sites/default/files/article/413938/hasp-2013-innovative-instructions-for-trusted-solutions.pdf. accessed: 2014-04-17.

[23] Kaspersky. Types of malware. online, 2014. URL http://usa.kaspersky.com/internet-security-center/threats/malware-classifications. accessed: 2014-06-23.

[24] Bernhard Kauer. Oslo: Improving the security of trusted computing. In *Proceedings of the USENIX Security Symposium*, volume 24, page 173, 2007. URL http://static.usenix.org/event/sec07/tech/full_papers/kauer/kauer_html/. accessed: 2014-04-17.

[25] Erica Klarreich. Cryptography breakthrough could make software unhackable. http://www.wired.com/2014/02/cryptography-breakthrough/, March

2014. URL http://www.wired.com/2014/02/cryptography-breakthrough/. accessed: 2014-04-02.

[26] Liwei Kuang and Mohammad Zulkernine. An intrusion-tolerant mechanism for intrusion detection systems. In *Availability, Reliability and Security 2008*, pages 319–326. IEEE, 2008. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4529353. accessed: 2014-09-03.

[27] Ralph Langner. Stuxnet: Dissecting a cyberwarfare weapon. *Security & Privacy, IEEE*, 9:49–51, 2011. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5772960. accessed: 2014-08-06.

[28] Ruby B Lee, Peter CS Kwan, John P McGregor, Jeffrey Dwoskin, and Zhenghong Wang. Architecture for protecting critical secrets in microprocessors. In *Computer Architecture, 2005. ISCA'05. Proceedings. 32nd International Symposium on*, pages 2–13. IEEE, 2005. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1431541. accessed: 2014-07-24.

[29] Yanlin Li, Jonathan McCune, James Newsome, Adrian Perrig, Brandon Baker, and Will Drewry. Minibox: A two-way sandbox for x86 native code. In *2014 USENIX Annual Technical Conference*. USENIX Association, 2014. URL https://128.2.134.25/group/pub/liatc14.pdf. accessed: 2014-08-12.

[30] David Lie, Chandramohan Thekkath, Mark Mitchell, Patrick Lincoln, Dan Boneh, John Mitchell, and Mark Horowitz. Architectural support for copy and tamper resistant software. *ACM SIGPLAN Notices*, 35(11):169–177, 2000. URL http://dl.acm.org/citation.cfm?id=357005. accessed: 2014-07-24.

[31] David J Malan. Cs50 sandbox: secure execution of untrusted code. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 141–146. ACM, 2013. URL https://dl.acm.org/citation.cfm?id=2445242. accessed: 2015-03-11.

[32] Mohammad Mannan and Paul C van Oorschot. Using a personal device to strengthen password authentication from an untrusted computer. In *Financial Cryptography and Data Security*, pages 88–103. Springer, 2007. URL link.springer.com/chapter/10.1007/978-3-540-77366-5_11. accessed: 2014-04-07.

[33] Jonathan M McCune, Bryan J Parno, Adrian Perrig, Michael K Reiter, and Hiroshi Isozaki. Flicker: An execution infrastructure for tcb minimization. In *ACM SIGOPS Operating Systems Review*, volume 42, pages 315–328. ACM, 2008. URL http://dl.acm.org/citation.cfm?id=1352625. accessed: 2014-04-30.

[34] Jonathan M McCune, Yanlin Li, Ning Qu, Zongwei Zhou, Anupam Datta, Virgil Gligor, and Adrian Perrig. Trustvisor: Efficient tcb reduction and attestation. In *Security and Privacy (SP), 2010 IEEE Symposium*, pages 143–158. IEEE, 2010. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5504713. accessed: 2014-08-12.

[35] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R Savagaonkar. Innovative instructions and software model for isolated execution. *HASP*, 133:10, 2013. URL http://css.csail.mit.edu/6.858/2013/readings/intel-sgx.pdf. accessed: 2014-04-17.

[36] Manoranjan Mohanty, Viktor Do, and Christian Gehrmann. Media data protection during execution on mobile platforms – a review. Technical report, SICS Swedish ICT AB, 2014. URL http://soda.swedish-ict.se/5685/1/Media_Data_Protection_during_Execution_on_Mobile_Platforms_%E2%80%93_A_Review.pdf. accessed: 2014-09-01.

[37] Steven J Murdoch. Introduction to trusted execution environments (tee). online, 2014. URL https://www.cl.cam.ac.uk/~sjm217/talks/rhul14tee.pdf. accessed: 2014-04-17.

[38] Job Noorman, Pieter Agten, Wilfried Daniels, Raoul Strackx, Anthony Van Herrewege, Christophe Huygens, Bart Preneel, Ingrid Verbauwhede, and Frank Piessens. Sancus: Low-cost trustworthy extensible networked devices with a zero-software trusted computing base. In *USENIX Security*, pages 479–494, 2013. URL https://www.usenix.org/sites/default/files/conference/protected-files/noorman_sec13_slides.pdf. accessed: 2014-07-24.

[39] Andy Ozment. *Vulnerability discovery & software security*. PhD thesis, University of Cambridge Computer Laboratory Computer Security Group &Magdalene College, 2007. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.167.2921&rep=rep1&type=pdf. accessed: 2014-04-16.

[40] Andy Ozment and Stuart E Schechter. Milk or wine: does software security improve with age. In *15th Usenix Security Symposium*, 2006. URL https://www.usenix.org/legacy/event/sec06/tech/full_papers/ozment/ozment_html/#note2. accessed: 2014-04-16.

[41] Marco Patrignani, Dave Clarke, and Frank Piessens. Secure compilation of object-oriented components to protected module architectures. In *Programming Languages and Systems*, pages 176–191. Springer,

2013. URL http://link.springer.com/chapter/10.1007/978-3-319-03542-0_13. accessed: 2014-07-24.

[42] Vassilis Prevelakis and Diomidis Spinellis. Sandboxing applications. In *USENIX Annual Technical Conference, FREENIX Track*, pages 119–126. Citeseer, 2001. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.22.4424&rep=rep1&type=pdf. accessed: 2014-03-26.

[43] Mark D. Ryan. Enhanced certificate transparency and end-to-end encrypted mail. In *Network and Distributed System Security (NDSS) Symposium*. University of Birmingham, UK CloudTomo Ltd., February 2014. URL http://dx.doi.org/10.14722/ndss.2014.23379. URL was not accessable. Paper received directly from Prof. Ryan.

[44] Mark D. Ryan, Gurchetan Grewal, Michael Clarkson, and Liqun Chen. Du-vote: Remote electronic voting with untrusted computers (invited talk). talk; abstract online, June 2014. URL http://ceur-ws.org/Vol-1158/fms_proceedings.pdf. accessed: 2015-03-16.

[45] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. Technical report, IACR Cryptology ePrint Archive, 2013. URL http://eprint.iacr.org/2013/454.pdf. accessed: 2014-04-02.

[46] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert Van Doorn. Design and implementation of a tcg-based integrity measurement architecture. In *USENIX Security Symposium*, volume 13, pages 16–33, 2004. URL https://www.usenix.org/legacy/events/sec04/tech/full_papers/sailer/sailer.pdf. accessed: 2014-04-17.

[47] Samsung. Samsung knox - white paper : An overview of samsung knox. Technical report, Samsung Electronics Co. Ltd., September 2013. URL https://www.samsungknox.com/en/system/files/whitepaper/files/MP_KNOX_Overview_Whitepaper_V1.0_20131023.pdf. accessed: 2014-06-23.

[48] Samsung. Samsung knox (website). online, October 2013. URL https://www.samsung.com/uk/business/solutions-services/mobile-solutions/security/samsung-knox. accessed: 2014-06-23.

[49] Samsung. Samsung knox - technical details. online, 2014. URL https://www.samsungknox.com/en/solutions/knox/technical. accessed: 2014-06-23.

[50] Guido Schryen. Security of open source and closed source software: An empirical comparison of published vulnerabilities. In *15th Americas Conference on Information Systems*, 2009. URL http://epub.uni-regensburg.de/21296/1/Schryen_-_AMCIS_09_-_Security_of_open_source_and_closed_source_software_-_Web_version.pdf. accessed: 2014-04-16.

[51] Seal One AG. Seal one usb - offering simplicity and elegance. online, 2013. URL http://www.seal-one.com/products-list.en-UK.html. accessed: 2014-08-21.

[52] Raoul Strackx and Frank Piessens. Fides: Selectively hardening software application components against kernel-level or process-level malware. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 2–13. ACM, 2012. URL http://dl.acm.org/citation.cfm?id=2382200. accessed: 2014-07-24.

[53] G Edward Suh, Dwaine Clarke, Blaise Gassend, Marten Van Dijk, and Srinivas Devadas. Aegis: architecture for tamper-evident and tamper-resistant processing. In *Proceedings of the 17th annual international conference on Supercomputing*, pages 160–171. ACM, 2003. URL http://dl.acm.org/citation.cfm?id=782838. accessed: 2014-07-24.

[54] Jakub Szefer and Ruby B Lee. Architectural support for hypervisor-secure virtualization. In *ACM SIGARCH Computer Architecture News* Lee et al. [28], pages 437–450. URL http://dl.acm.org/citation.cfm?id=2151022. accessed: 2014-07-24.

[55] Trustonic. Who we are. online, 2014. URL https://www.trustonic.com/about-us/who-we-are. accessed: 2014-08-25.

[56] Tyfone. Secure transaction platform. online, 2004. URL http://tyfone.com/Platform.php. accessed: 2014-08-21.

[57] Amit Vasudevan, Emmanuel Owusu, Zongwei Zhou, James Newsome, and Jonathan M McCune. *Trustworthy Execution on Mobile Devices: What security properties can my mobile platform give me?*, volume 7344 of *Trust and Trustworthy Computing, CyLab, Carnegie Mellon University, USA*. Springer, November 2012. doi: 10.1007/978-3-642-30921-2_10. URL https://www.cylab.cmu.edu/files/pdfs/tech_reports/CMUCyLab11023.pdf. accessed: 2014-06-23.

[58] Amit Vasudevan, Sagar Chaki, Limin Jia, Jonathan McCune, James Newsome, and Anupam Datta. Design, implementation and verification of an extensible and modular hypervisor framework. In *Security and Privacy (SP), 2013 IEEE Symposium*, pages 430–444. IEEE, 2013. URL http://ieee

xplore.ieee.org/xpls/abs_all.jsp?arnumber=6547125. accessed: 2014-08-12.

[59] Giuliana Santos Veronese, Miguel Correia, Alysson Neves Bessani, Lau Cheuk Lung, and Paulo Verissimo. Efficient byzantine fault-tolerance. *Computers, IEEE Transactions on*, 62(1):16–30, 2013. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6081855. accessed: 2014-04-08.

[60] Yhaohui Wang and Angelos Stavrou. Exploiting smart-phone usb connectivity for fun and profit. In *Proceedings of the 26th Annual Computer Security Applications Conference*, pages 357–366. ACM, 2010. URL delivery.acm.org/10.1145/1930000/1920314/p357-wang.pdf. accessed: 2014-04-07.

[61] Carsten Willems, Thorsten Holz, and Felix Freiling. Toward automated dynamic malware analysis using cwsandbox. *IEEE Security and Privacy*, 5 (2):32–39, 2007. doi: http://dx.doi.org/10.1109/MSP.2007.45. URL http://dl.acm.org/citation.cfm?id=1262675. accessed: 2014-04-17.

[62] Gao Xiaopeng, Wang Sumei, and Chen Xianqin. Vnss: a network security sandbox for virtual computing environment. In *Information Computing and Telecommunications (YC-ICT), 2010 IEEE Youth Conference on*, pages 395–398. IEEE, 2010. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5713128. accessed: 2014-04-17.