German University in Cairo
Computer Science Department
Mervat Abuelkheir
Ahmad Helmy
Mohamed Abdelrazik

CSEN1001:Com. & Netw. Security
Spring, 2019

Course Project: Pick your poison!

1 Preface

By reading this document, you agree that all of the links and resources provided are only for research purposes and by no means any of the information represents the author's or the GUC's point of view. You also agree that you will not use your gained knowledge to illegally break into systems and that you will be fully liable for the consequences of your own actions.

2 Summary

Security has often been considered an afterthought rather than a core fundamental. During software engineering, it is usually a non-functional requirement rather than a functional requirement[1]. For this reason, most of the systems and protocols that do not design software architectures with security in mind are inherently insecure[2]. It was claimed that security can have a bad impact on usability and sometimes, performance too. Thus, most currently secure protocols and cryptographic tools are hard to use by the normal non-experienced user. Additionally, security can hinder the process of software development as it gets in the way of debugging and makes testing even harder.

The security community have come to this realization[3] over the past couple of years. It is essential to have usable cryptographic tools even for the non-experienced user. The recent alarming increase in world-scale governmental surveillance programs such as prism[4][5] have also tightened the rope on security researchers and made rethinking the entire World Wide Web from a security point of view an urgent necessity to restore privacy, security and anonymity back to the Internet's users.

Security is rather an entire parallel layer that should be integrated with the entire spectrum of IT project management. Secure agile software development[6][7] is just scratching the surface. Along the same lines, computational theory and formal grammars for computational models should not only be used to write a certain program but, to also treat the input as a "secure" formal grammar[8]. To the aforementioned concerns, the security

[1] http://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/
[2] http://goo.gl/QPaBhI
[3] Keynote of the 30th Chaos Communication Congress http://youtu.be/gyA6NZ9C9pM
[4] Electronic Frountier Foundation: http://youtu.be/SH4_Z0ay5jU
[5] Jacob Appelbaum: Not my departmenthttp://youtu.be/7mnuofn_DXw
[6] https://www.sans.org/reading-room/whitepapers/securecode/software-engineering-security-process-sdlc-1846
[7] http://www.researchgate.net/profile/Premkumar_Devanbu/publication/2393383_Software_Engineering_for_Security_a_Roadmap/file/3deec51de3f2ea22a0.pdf
[8] The Science of insecurity: http://youtu.be/v8F8BqSa-XY

community has responded. Some numerous examples include: GNUnet[9], Tor[10], Mylar[11], Cryptocat[12] and much more.

Your job in this project thusly, is to join the battlefield and become an active member of the hacker community. You will be developing a secure alternative to current "security-broken" applications. Something that is both: easy to use and secure by design at the same time. Didn't you hear the call from Julian Assange?[13]

3 Projects list

Rationale

You should pick any project from the following list and implement it in the most secure way possible. The use of external frameworks and/or libraries is limitedly allowed. You should take the instructor's permission if you would like to use a library/framework in your implementation[14]. Yet, you are free to use whichever programming language you prefer. No matter which project you choose, you should provide the following security features *per project requirement*: *Integrity*, *Availability*, *Message Authentication*, *User Authentication*, *Confidentiality*, *NonRepudiation*, *Anonymity*, *Protocol Confusion/Diffusion*, *Resistance against Active/Passive attacks* and *Security-classified logging* for further intrusion detection audits. Also, make sure you don't fall for the top 10 web vulnerabilities[15]. By the end of your project, you should demonstrate everything you learned throughout the course and your research effort before implementation to make sure everything is secure (impossible) and even if it is not, there is another security layer just in case.

The main objective is not just to implement the ideas but, to look at each project feature and assess its security having the aforementioned information security attributes in mind. Hence, you are free to innovate and choose the proper security algorithms, techniques or protocols that suit your needs as well as implement your own. Try to think of (and research) all possible attacks that could be performed and what happens if a certain defensive layer was breached. Think of the scenarios we discussed per project in the brainstorming session.

3.1 Steganography Chat

Steganography is simply the art of hiding data within data[16]. For example, you could hide some text within the pixels of an image[17] or, within the pixels of a given frame of a video. You could even do the same with audio or any other kind of file. In this project, you are required to implement a *decentralized* P2P chat application which applies steganographic techniques to relay secret messages back and forth between multiple users. Your application should have the following features:

- A lobby chat room, where all users can send messages back and forth in a "send to all" fashion.
- Support for private chat where a user could choose to engage a conversation with another user.

[9] Your broke the internet. We're making ourselves a new onehttp://youtu.be/H7TVRYN6Vhk
[10] Anonymous decentralized access to the webhttps://www.torproject.org
[11] MIT - Building secure web appshttp://css.csail.mit.edu/mylar/
[12] https://crypto.cat/
[13] Sysadmins of the world, unite! http://youtu.be/hzhtGvSflEk
[14] Use the course's Q&A website
[15] https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf
[16] http://www.garykessler.net/library/steganography.html
[17] Worth checking out: https://ccrma.stanford.edu/~eberdahl/Projects/Paranoia/

- Sending files back and forth in the above two scenarios with assumption of a man-in-the-middle.

3.2 Secure Twitter

Social networks nowadays help amplify *disinformation* from multiple users regardless of their true identity. In this project, you are required to rethink twitter[18] from a security point of view. Imagine a social network where the flow of information between users, inter and intra communities is controlled by authenticity and security principles yet, everything still feels like a typical social network to the end user. Think of a surveillance proof social network yet, the information should still be spread through retweeting in a secure fashion allowing the user to expand his "reach" through friends of friends. Your application should support everything that twitter currently does: Tweeting, Following/Unfollowing, Direct Messages, Trending Topics, Staring tweets, Retweets, User Lists.

3.3 Secure WhatsApp

WhatsApp is a great enabler for chatting between mobile devices. However, it is not secure enough. Especially, if it is done through a centralized fashion. If attackers get access to the servers used to relay the messages, everything could be intercepted. Even with encryption, it is only a matter of time till the messages of the clients are deciphered. Another scenario is not actually verifying the mobile device at the other end but, also verifying the user. For example, imagine a stolen device being used by an attacker. In this project, you are required to reinvent WhatsApp[19][20] in a *decentralized* and secure fashion. You should support: Group chat, Private chat, Sending files, Sharing location.

3.4 Secure Dropbox

Secure backup and cloud file storage solutions are argued to lack security and compromise key security requirements. Especially, if the cloud provider or service operator is untrusted or have been compromised. A more secure way would be to disable anyone completely from being able to gain access the hosted data or know where the data actually is at any instance of time. Your role in this project is to renovate Dropbox[21] standing on security grounds. Instead of relying on the cloud, your application should provide a *decentralized P2P* way to securely disseminate your files across other users yet, still be able to retrieve them at will in a fashion similar to how torrent downloads operate. You should support: Sharing of a specific file with multiple users, Uploading/Syncing a file, Downloading a file you are authorized to access.

3.5 Linux Rootkit[22][23][24][25]

It was argued that Linux is one of the most secure operating systems to date with a very low record of detected malware infections over the years. However, implementing rootkits using *Linux Kernel Modules* (LKM)[26][27] allows an attacker to have full root permissions on a Linux machine and intercept system calls in order to perform malicious activities. In this project, you are required to write your own simple Linux rootkit that performs the

[18] Worth checking out: http://twister.net.co/, http://www.trsst.com/
[19] https://threema.ch/en/index.html, https://core.telegram.org/mtproto
[20] Bonus: Can you implement your work on top of whatsapp? https://github.com/tgalal/yowsup
[21] http://syncthing.net/
[22] http://info.fs.tum.de/images/2/21/2011-01-19-kernel-hacking.pdf
[23] http://memset.wordpress.com/2010/12/28/syscall-hijacking-simple-rootkit-kernel-2-6-x/
[24] http://althing.cs.dartmouth.edu/local/LKM_HACKING.html
[25] It is advised to use a virtual machine with any Linux distro that has kernel version 2.6.x or less. You will also need to install the Linux kernel development headers from the repositories.
[26] Sample implementation: http://average-coder.blogspot.de/2011/12/linux-rootkit.html
[27] LKM development book: http://www.tldp.org/LDP/lkmpg/2.6/lkmpg.pdf

following: Gives you root access, hides your module from the list of modules, hides a process from the list of running processes. These extra features are also welcome (bonus): hiding an open socket from the list of open sockets, implementing a keylogger.

3.6 Anomaly and Intrusion Detection

Distributed denial of service attacks have become a major contributor to the downtime of many popular web services over the past few years. Through this project, you are required to leverage *machine learning* and *deep packet inspection*[28] in order to passively detect unusual Network traffic. More precisely, you are required to detect at least 3 different types of DoS attacks[29] and log the incidents.

4 Deliverables

- All source code and documentation for your project implementation.
- A README file explaining how to run and configure your project along with the dependencies on libraries, frameworks, etc.)
- A Project report of not less than 6 pages, font size 12, 1.5 line spacing containing:
  - Summary of your project, motivation and how you implemented its features.
  - Your design choices of cryptographic algorithms and protocols for each functionality. Explain why you used a certain methodology in particular and how you made such security techniques fit your project requirements.
  - A detailed comparison between the security methods, protocols and algorithms you researched and why you think your approach is best.
  - Attack scenarios that you thought about (and researched) and how your system is secure against them. Examples and screenshots would be much appreciated.
  - Explanation of each computer security technique you used along with its advantages/disadvantages trade-offs.
  - Explanation of libraries/frameworks you used. In particular, what security features you utilized and how did the library offer you such features?
  - A comprehensive list of all references, links and other aids that helped you during the project implementation, security assessment and research.
- Evaluation will focus on the implementation of the following security features:
  - Authentication
  - Encryption/Hashing
  - Access Control
  - Tokens
- During the evaluation, you should deliver a 10 min presentation about your project and provide a live/recorded demo of the attacks that could be made and how you defended against them.

5 Administrivia

- Any plagiarism detected will be penalized with a **zero**.

[28] http://www.tcpdump.org/pcap.htmlhttp://yuba.stanford.edu/~casado/pcap/section1.html
[29] http://youtu.be/1EAnjZqXK9E

- Submission should be no later than **02.05.2019**.
  - Upload your code to a GitHub repository and send link to your TA.
- Presentation of work done by teams will be scheduled over the days 4-7 May 2019.

5.1 Teams

- Teams of **[4-5]** members are allowed.
- All members are expected to collaborate equally and understand every detail behind the project.

5.2 Grading

- Report: 10%
- Presentation + Demo: 6%
- Secure implementation of features: 84% (each feature implementation gets 21%)
- 5% percent bonus is awarded if substantial work is shown.

6 Resources

Lost? Here are some key terms you will definitely need to use:

- Pretty Good Privacy (PGP)
- Distributed Hash Tables (DHT)
- Public & Private Key Cryptography (Symmetric and Asymmetric)
- One Time Passwords and User Authentication
- Challenge Response Authentication and n-factor authentication
- Hash functions, MAC, …
- Stream ciphers, Block ciphers
- Look at the topics discussed in the tutorials!

6.1 Links

- Self-signing SSL: http://www.akadia.com/services/ssh_test_certificate.html
- DHT implementation: https://github.com/jeanlauliac/kademlia-dht
- DHT: http://jinroh.github.io/kadoh/
- PGP: http://www.gnupg.org/ , http://pgp.mit.edu/
- Tor hidden services: https://www.torproject.org/docs/hidden-services.html.en
- Bitcoin API: http://bitcoinjs.org/
- Bittorrent API: http://btappjs.com/
- OWASP Top 10 Web Vulns: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf

- Crypto hashing for mobile: https://code.google.com/p/farmhash/
  https://code.google.com/p/cityhash/
- Google Crypto tool: https://code.google.com/p/keyczar/
- Javascript cryptography: https://code.google.com/p/crypto-js/
- C++ Cryptography: http://www.cryptopp.com/
- Blowfish, a better DES?: https://www.schneier.com/blowfish.html
- Stronger Password Hashing: BCrypt, SCrypt
- Other hashes: MD5, RC4, SHA-512...: http://kremlinencrypt.com/algorithms.htm