OWASP Top 10 - Do Startups Care?

Halldis Sϕhoel, Martin Gilje Jaatun, Colin Boyd

Citation:
Halldis Sϕhoel, Martin Gilje Jaatun, Colin Boyd, OWASP Top 10 - Do Startups Care? (2018) . In: 2018 International Conference on Cyber Security and Protection of Digital Services (Cyber Security) 2018. DOI:10.1109/CyberSecPODS.2018.8560666

OWASP Top 10 - Do Startups Care?

Halldis Søhoel
BEKK Consulting
Postboks 134 Sentrum
NO-0102 Oslo, Norway

Martin Gilje Jaatun
SINTEF Digital
NO-7465 Trondheim, Norway

Colin Boyd
IIK, NTNU
NO-7491 Trondheim, Norway

***Abstract*—In a cut-throat world where time-to-market can be the difference between success and failure, it can be tempting for startups to think "let's get it to work first, and then we'll worry about security later." However, major security flaws have killed more than one startup. This paper studies a small sample of 5 IT startups that offer services via the web, to determine to what extent they are aware of and can handle the OWASP top 10 threats.**

I. Introduction

"The Internet of half-baked Things is upon us" the security expert Marcus Ranum states on the Silver Bullet podcast [1]. Today we are putting chips in everything. This results in the existence of major unused and poorly secured computing capacity connected to the internet. At the same time Apple and Google have had great success with open development environments opening the door to the software developing world to anyone willing to learn coding. And people are willing; every month hundreds of thousands of new apps and web services with varying quality appear online. We are using these services to solve our needs in every aspect of our everyday lives; sleeping, eating, traveling and shopping. Consumers rarely ask questions about the security of these systems.

According to Steve Bellovin, "any program, no matter how innocuous it seems, can harbour security holes" [2]. A statement that was nicely illustrated on October 2016 when a DOS attack on the internet infrastructure provider, Dyn [3], resulted in downtime for Twitter, Netflix and several other high-profile web pages. It turned out that the attack was performed by a botnet of poorly secured IoT devices such as web cameras, video recorders and surveillance cameras.

As web usage increases and becomes part of our day to day life, compromising these systems become increasingly rewarding. Attackers no longer need physical access to the victims, they can attack more than one at the same time and the possibility of being caught and brought to justice is minimal.

In any other science discipline, the first thing you learn is the consequence of making mistakes. They teach you that poorly designed buildings or bridges may cause great disasters. In computer science, the focus is quite different. In fact, at many universities and university colleges, although security courses are offered, they are not a mandatory part of the ICT programs, leaving many graduates without the most basic knowledge about security [4]. The attitude among some developers seems to be that "as long as it functions the intended way, everything is fine". The problem arises when hackers start using the system in an unintended way.

How secure are all these rapidly appearing services? This paper will contribute to answering that question. Specifically, we will investigate how well security is maintained in five start-up companies. Startup companies are an interesting group to explore because there are no formal requirements to who can start an IT-company. Anyone willing to learn how to code can start a business.

Another reason to think that security might be neglected in startups is that building software is expensive and startups do not have the resources to invest in anything that will not bring them closer to the next round of financing. McGraw and Ranum believe that the core of the problem is that the software industry is more rewarding to people that are a part of the problem, rather than those that are a part of the solution. If companies spend that extra time and resources on creating good, secure software, they will get delayed in the release cycle. Especially for startups this can mean life or death for the business.

On the other hand startup companies have two major advantages; their high motivation drive them to make better products, and because they are small in size they can quickly adapt to secure development life cycles. In fact, smaller companies implement secure development life cycles at a higher rate than larger companies, likely because they have fewer decision levels and are less prone to bureaucracy [5].

Literature suggests that startups manage to make secure applications even when their knowledge level indicate differently [6]. There can be two reasons for this; because they are highly motivated and personally invested in the product they put in the extra effort to do it right. Another reason may be that high profile companies like Microsoft, Google and Facebook are making available complete functionality that take care of critical transactions, and provide secure sample code that help developers make more secure applications.

Different applications have different security needs based on the sensitivity of the information stored and the consequences of the transactions the application is used for. Some applications can accept a higher risk than others. To evaluate this we cannot only look at the technical aspects, but need to consider the threats and the business values of the system. There needs to be compliance between the security needs and the security achieved.

The Open Web Application Security Project (OWASP) top

ten constitute a consensus among security experts of the ten most critical web vulnerabilities on the web today. A test plan based on this list will provide technical insight on the security of the applications.

The remainder of this paper is organised as follows: In Section II we present relevant background, with related work in Section III. We present the method employed in Section IV, and the participating companies in Section V. We discuss our findings in Section VII, and conclude in Section VIII.

II. Background

One of the awareness documents provided by the Open Web Application Security Project (OWASP) community is the ten most critical web application vulnerabilities. The latest version that received consensus by the start of this project is the OWASP top 10 2013 [7]. The list was developed through analysing 500,000 vulnerabilities found in thousands of applications. The vulnerabilities are rated based on number of occurrences, exploitability, detectability and impact. The 2013 ranking is listed in Table I.

A1: Injection
A2: Broken Authentication
A3: Cross-Site Scripting (XSS)
A4: Insecure Direct Object References
A5: Security Misconfiguration
A6: Sensitive Data Exposure
A7: Missing Function Level Access Control
A8: Cross-Site Request Forgery (CSRF)
A9: Using Components with Known Vulnerabilities
A10: Unvalidated Redirects and Forwards

TABLE I
OWASP TOP 10 2013

OWASP top 10 2017 was released in the spring 2017, but was later pulled back as the community disagreed on its validity. The final version was finally published on October 20, unfortunately too late to be part of this project. In the 2017 release two of the vulnerabilities from 2013, A8 and A10, were retired, and A7 and A4 were merged into one under "Broken Access Control". There are three new vulnerabilities added; XML External Entities (B4), Insecure Deserialization (B8) and Insufficient Logging & Monitoring (B10). OWASP top 10 2017 is listed in Table II.

There has been quite a lot happening on the application development front the last 4 years. Microservices, RESTful APIs and Single Page Applications have completely changed the architecture of web applications and come with their own set of security challenges. The fundamental technologies have changes and are now dominated by new web frameworks such as Angular and React [7]. Many of the vulnerabilities are rearranged. Luckily all the vulnerabilities from OWASP top 10 2013 are still relevant, only two were taken off the list due to low occurrences. However, they are still among eight vulnerabilities in the 2017 release under "Additional Risks to Consider". B4, B8 and B10 from 2017 were not tested in this project.

B1: Injection
B2: Broken Authentication and Session Management
B3: Sensitive Data Exposure
B4: XML External Entities (XXE)
B5: Broken Access Control
B6: Security Misconfiguration
B7: Cross-Site Scripting (XSS)
B8: Insecure Deserialization
B9: Using Components with Known Vulnerabilities
B10: Insufficient Logging & Monitoring

TABLE II
OWASP TOP 10 2017

III. Related work

A secure development lifecycle (SDL) is a term used for addressing security throughout the software development process and implementing security activities through all phases of development. Another, and perhaps more precise, term used to describe this is Secure Software Development Life cycle (SSDL). Today there exist different methodologies to achieve this; Microsoft SDL, Microsoft SDL Agile, open-SAMM (software assurance maturity model) and Comprehensive Lightweight Application Security Process (CLASP). Because there were so many different opinions and approaches, The Building Security In Maturity Model (BSIMM) was started to describe "existing software security initiatives"[8].

A survey by Errata Security [5] showed that 81 % of the asked companies had heard about the methodologies, however only 30.4 % were using them. When asked about why they did not use them 23.9 % provided that they were too time consuming, 15.2 % that they required too many resources and 4.3 % that they were too expensive.

The survey also showed that smaller companies implemented secure software development life cycles at a higher rate than larger ones. This is natural as smaller companies are more flexible because they do not have as many decision levels and protocols to follow.

A. Software Security in Norwegian organizations

Two studies have recently been conducted to determine the state of practice in Norwegian organizations with regard to security. Nicolaysen et al. [9] studied the software security initiatives of six companies using agile software development methodologies and Jaatun et al. [10] studied 20 public Norwegian organizations developing software.

Consistent with the Errata survey, Nicolaysen et al. [9] found that very few of the companies were utilizing methodologies for creating secure software. The developers had no formal training in developing secure software and very

few were concerned about security. Functionality was often prioritized over security.

The study conducted by Jaatun et al. [10] showed big variations in the number of security activities being performed. Out of the 112 activities [11] the organizations were asked about, one organization did only nine, while another did 87. On average the organizations did 44 out of the 112 activities. The study also revealed that few of the organizations were using a systematic approach to create secure software and the activities being done depended on initiatives from individuals. Also developing secure software was not a priority, instead the organizations were relying on infrastructure to solve the security needs.

B. Software security in applications made by Silicon Valley start-ups

Bau et al. [6] conducted an experiment where 19 Silicon Valley start-ups and 8 hired freelancers were asked to take a basic software security quiz and submit an application. The security of the applications were then tested through static analysis and penetration testing.

The results from the quiz showed that the scores for the startups were very dispersed. Some of them scored close to 100 % while others knew less that 50 % of the questions. The startups also did slightly better on the quiz than the freelancers, but not statistically significant, hence their knowledge level were concluded to be the same.

The testing showed that the startups made significantly more secure applications than the freelancers. Furthermore, the negative correlation between the quiz score and the number of vulnerabilities found were also bigger among the startups. This means that the startups did a better job at using their security knowledge in practice and implementing what they knew.

The tests also showed that in many cases the startups were successful in developing secure code even in areas where they had failed on the quiz. This shows that even in areas were they lacked knowledge they were able to figure it out and implement it in a correct and secure way. Bau et al. [6] believe that there are two factors contributing to these facts; the startups are more motivated and invested to make good solution, and often developers use frameworks or copy-paste secure code which may "save them". One example of this was one of the freelancers that answered on the quiz that a secure way to store a password was in plain text; still, in the application delivered he had hashed the password with a salt.

IV. Method

To answer the research questions a handful of techniques have been selected.

A. Literature study

The Literature study started with a search on Google scholar for material relevant to the topic. Then the references of the sources found was followed, and a search for additional material by relevant authors was made. The background study is a historical review showing how traditional computer security has evolved into the approaches seen today and how these are used in web application security. The related work section presents work that is relevant for this project.

B. Interview

A semi-structured interview with company representatives was performed. The idea was to have a somewhat free conversation, where the representatives talked about their application and their process. A list of topics necessary to go over was planned beforehand. The companies would talk about each topic and some follow-up questions were asked when necessary. The goal was to determine if and how they work with security, establish the business context, listen to their thoughts around the risks connected to the application and get a picture of how familiar they were with the OWASP top 10.

C. Testing

To find out if any of the applications are exposed to the OWASP top 10, vulnerability testing was performed. There are two possible techniques to test for security holes; Black-box testing and white-box testing.

a) Black-box testing: (or penetration testing) is based on penetrating the running application from a user perspective. This means that the tester has no special privileges or prior knowledge of the application other than that available for everyone. This is the approach most similar to a real attack scenario where an adversary is trying to compromise the application. The tester adapts the mindset of an attacker and explore real security holes instead of hypothetical ones. The disadvantage of this approach is that the discovery of some vulnerability using this method establishes with certainty that the vulnerability exists, however if a vulnerability is not found this is no proof that it is not there. Also, the success of the testing relies upon the skills of the tester and it is assumed that the skills of the attacker does not exceed these.

b) White-box testing: extends penetration testing by reviewing the source code looking for security flaws. This approach is quite time consuming and require high expertise as some logical flaws may require detailed knowledge of the workings of the entire system. The testing can either be done manually or automatically by using automated tools.

Austin and Williams [12] found that in order to fully disclose the state of security in an application, multiple techniques should be used. They found that the different techniques discovered very different types of vulnerabilities. Automated penetration testing was found to be the most effective way of finding vulnerabilities. However, the tools sometimes produce false positives and each vulnerability needs to be verified manually. Manual penetration testing was the best approach at finding architectural flaws and automatic static analysis was better at finding implementation bugs. To get the full picture of the security it is recommended to use all three techniques.

The technique used in this project is a combination of automated penetration testing and systematic manual penetration testing. Testing followed a test plan developed to cover

OWASP top 10, and multiple tests were done for each. The advantage of following a test plan is that by the end of testing the tester is confident that all vulnerabilities are covered. For some of the vulnerabilities there exist useful tools to get a quick assessment. The vulnerabilities found must then be verified manually if they pose a real threat. For other OWASP top then vulnerabilities, for example "Broken Authentication and Session Management" that are of an architectural nature, manual testing was mostly used. The tools used were Burp Suite, Whatweb, Netcat, Dirbuster, Nikto, openSSL and Nmap. All tools were available on Kali Linux.

D. Evaluation/recommendation of relevant tools and resources

Based on the experience gained during testing, a set of tools and resources is recommended according to the following criteria:

- *The tools should be free or not too costly*; start-ups often do not have a lot of resources to pay for expensive tools.
- *They should be quick and easy to install and use*; for non-experts whose main priority is developing a good application, it should be easy and intuitive to get started with security improving tools.
- *There should be a limited amount of tools to cover all of OWASP top 10*; the collected set of tools should cover all of OWASP top 10 but the number should be limited to make it both easier and faster to get acquainted with all.
- *One should not need to be an expert in order to benefit from the tools*; the recipient is a non-expert.
- *The tools should give a limited number of false positives*; If the tools gives too many false positives, verifying all positives get more time consuming than manual testing.
- *The tools should be well documented*; This is important for accelerating the learning process and making the tools useful in less time.

V. Company context

This chapter provides some context about the companies. Section V-A presents the five applications and the start-ups behind them. In Section V-B the insight gained from the interviews are presented.

A. The five participating start-ups

To take part in this study five very diverse start-up companies have been recruited. Some store very sensitive and critical information about their users, while others are less critical in the sense that compromising the site will not have major consequences for the users. Some of the start-ups have a lot of knowledge and experience about web development and security, while others are more inexperienced and "learn-by-doing". One of the companies do not develop the application themselves but have outsourced all coding to a foreign firm. The founders of this company all have backgrounds in business and management.

Company A is a B2B business delivering a time management and project management systems to companies. The site stores sensitive information about their customers such as customer relationships, invoices and personal information about the employees such as national identities, bank accounts and e-mails. The founders all have backgrounds in business and management and have outsourced development to a foreign company.

Company B is a crowd-sourcing website for educational materials. The users are generating content in the form of Q/A, quizzes and discussions. What each user is allowed to do depends on the level of privileges. A user acquire higher privileges by contributing to the site and getting up-votes from other users. It is also possible to visit the site as a guest user. The company consists of 4-5 student studying computer science and related fields.

Company C is an open source project for online booking of laundry facilities. The users can create laundries and invite tenants to book machines. The start-up consist of 4 software development engineers with experience with web development and security. The application is a hobby project beside full time jobs.

Company D provides a communication solution for medical personnel, patients and dependents. It stores highly sensitive and personal information about patients connected to a medical institution. The content can be of the form of pictures, videos, texts or timetable of patient, routines and even when the patients last went to the toilette. One of the things that make this site so critical is that many of the patients are not legally competent and cannot consent to their information being made public. It is therefore important to protect this group form being exploited. The application was developed as an alternative to publishing and sending this information through Facebook.

Company E provides tutoring services for children in primary school. The tutors have to upload a certificate of good conduct form the police and their diploma on the web site. The students enter what grades they have in the subject they want tutoring in what grade they want to achieve. Also the students enter payment information. The founders have degrees in computer science and related studies.

B. Interviews

During the interviews, the companies were asked about what their most important business assets were, and we performed a risk analysis of their application. We wanted to know what was critical to protect on the page, who the threat agents were and what the worst-case scenario is if the application were to be attacked. They were also asked about how they worked with security. In the end we went through the OWASP top 10 to see if they had any knowledge about them, and determine which ones were the most relevant to protect their specific assets.

Company A: During the interview we agreed that their most critical assets are the sensitive information about people and businesses, and leakage of this information would lead to end of business for the start-up. The most critical vulnerabilities are those connected to authentication and access control.

They said that until now the priority has been on functionality and creating a working application, but now that they have real customers they want to focus on making the application secure. From the OWASP top 10 only injection attack was known to them. They have no idea whether their application is secure or not and has so far not been doing any security activities. It is clear that the founders are eager to deliver the best possible product to their customers and that they now realize that security is a part of that.

Company B: There is no sensitive information on the page that is not already open for everyone to see. Therefore the greatest risks on this page are destruction and spamming. CSRF, XSS and privilege escalation are therefore the most severe vulnerabilities. Unsafe redirections are also relevant for this page because the site can be used as an enabler for phishing attacks. Apart from using secure libraries they haven't done any security activities and the coders are in charge of handling security. They had heard about OWASP from taking a security course at university and had knowledge about some of vulnerabilities such as injection and XSS, but they did not have detailed knowledge about OWASP top 10. They suspected that there might be a lot of vulnerabilities on the site and were really eager to have it tested. Also, they offered to help with the testing and clearly wanted to learn more about security.

Company C: The application is developed by 4 professional web application developers as a side project in addition to their full time jobs. They seem to have a lot of knowledge and experience on both web development and security. They store as little information about the users as possible to make the application simple to use. The most sensitive info on the site are the e-mail addresses of users. We agree that the biggest threads to the system are leakage of e-mail addresses and passwords, spamming, changing and deleting of bookings and general destruction. To ensure security they make sure to use secure frameworks and use the guides of renowned developers on how to use them correctly. For example once a week they use snyk to scan through all dependencies in their project to look for libraries with known vulnerabilities. They have been testing for access control and enumeration of e-mails. They were familiar with OWASP top 10 and strives to keep themselves up to date on them.

Company D: The patients are in many cases not able to consent to how their information is managed. The information stored is not really valuable in itself, but because of how sensitive it is to the persons to whom it belongs, it is very critical to protect. In some cases information from medical records may appear on the page. Because of the sensitivity, the business has a lot of focus on security. They have security reviews every six months, make sure to have all frameworks up to date and keep themselves updated on known vulnerabilities. Also they get a lot of advice from the Data Inspectorate to make sure they comply with Norwegian law. When asked about the OWASP top 10, they provided a list stating how they protect themselves against each one. They are currently working on implementing BankID which requires the highest security level in Norway [13].

Company E

This company was ultimately not available for interview.

VI. Results

On all applications there were significant security holes that need to be fixed. Those companies with the lowest knowledge about OWASP were also the ones with the most critical security holes. Those who considered the security from the start had significantly better security. The results are summarized in Table IV, where "P" indicates a passing test, and "F" is a fail.

Company A did not have high hopes about their security and had prioritized functionality. Still they admitted that security breaches could lead to the end of their business and that they had multiple competitors that would be interested in taking over their market shares. Testing showed that their security architecture was faulty with multiple severe holes; targeting specific customers and gaining access to their accounts and even steal admin credentials did not seem at all too difficult.

Company B: Even though application B does not contain any confidential information they were very afraid of someone destroying material or spamming. The testing shows that even though there is not much to gain from this, it was so easy to do that someone might still try. These were students with not too much knowledge about OWASP and not much experience with web development, learning by doing. This is a good example that there are a lot of pitfalls when it comes to developing applications and that security need special attention.

Company C: Although application C did not contain any critical information, it was still one of the most secure. The application was very professionally made and very neat. the company had taken many measures to ensure security. They had been a little careless with the use of tokens, which could in turn lead to someone deleting reservations or booking entire laundries. However, to steal the token the attacker still would need access to the victims computer and an attack would be limited to only one individual most likely. If an attacker were to get hold of a token once he/she could use it forever.

Company D: Application D contains very sensitive information and hence the security needs to be handled accordingly. They were very aware of OWASP and had taken measures to protect themselves from the start. In fact they were very well protected in most ways, although they had some slips. The vulnerabilities here were caused by not considering the misuse case of someone looking and manipulating the actual HTTP request/responses. The exploits for these vulnerabilities were limited in reach by the fact that only authenticated users could use them, and the account provisioning is strong; the attacker needed to be at a specific location at a very specific time to do the exploit. Nevertheless these are vulnerabilities of significant risk and need to be fixed.

Company E: Application E in contrast to application C seemed very amateurishly made. There were multiple functions that did not work according to their purpose and the

TABLE III
SUMMARY OF COMPANIES

	Outsourcing	Information sensitivity	Knowledge about OWASP	Web Development Experience
Company A	Yes	Medium/High	Low	Low
Company B	No	Low	Low	Low
Company C	No	Low	Medium	High
Company D	No	Very High	High	Medium
Company E	No	High	-	-

TABLE IV
SUMMARY OF RESULTS

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
Company A	P	F	P	P	F	F	P	F	F	P
Company B	F	F	P	F	F	F	F	F	P	P
Company C	P	F	P	P	P	P	P	F	P	P
Company D	P	F	P	F	P	F	F	P	F	P
Company E	P	F	P	P	P	F	P	P	P	F

password policy was inconsistent. Simple functions like setting the password didn't work at all. In despite of this the application was really secure. Looking at the HTTP packet capture almost none of the requests had the site as host, but in their place were Google, Facebook, Stripe and other third parties. When inspecting the responses they were particularly well configured. So was their Firebase server. When targeting the search to request that were actually handled by the target application we found a redirect vulnerability that could be used to perform a phishing attack. Also even though all critical transactions went through third party code, the less critical parts of the site was not configured to be very secure. We believe that this illustrates that using secure third party code and sample code is a very good help for inexperienced developers to make bullet proof applications.

The results of this project are consistent with the view that, regardless of the security needs of an application, those who consider security from the start make significantly more secure applications. This is illustrated by the findings on company A, B, C and D. A and B had very different security needs, but what they had in common was that they had not yet considered security. The two applications in turn showed a lot of security holes. On the other side there were application C and D, where C had very low security need and D had very high. They both were very up to date on OWASP and had implemented multiple measures to make secure applications. The application in turn showed few security holes and the vulnerabilities found were in general limited in reach.

The results also provide additional weight to the findings of Bau et al. [6] that argued that start-ups make more secure applications because they are more motivated. Application A, which was the only application that was outsourced, was also the application with the most severe security holes. This is despite of having high security needs.

Findings on application E also shows that secure third party code and samples from big actors make a big difference when smaller en slightly inexperienced companies develop applications. Whenever third party code was being used the security was exceptional. This material is a great contribution to the overall security on the web.

VII. DISCUSSION

A. Is security a concern?

It varied how much the start-ups were concerned about security. While company C and D had clear goals on how to secure their application and could name a number of measures that they did to maintain security, Company A and B agreed that security was important but did not seem to realize to what extent before the interview. Company A and B also did not know if their applications were secure or not, and did not do anything to accomplish a secure application. After being in touch with many companies in regards to this thesis there is a clear division in attitudes towards security. Some companies were very eager to participate in the project, always replied to e-mails in timely manner and always provided us with any additional information or documents needed. The four companies already mentioned were all very grateful to have their application tested and asked for advice on how to move forward and improve their security. On the other side, the majority of companies we approached said that they did not have time, did not want to commit, or were in general very hard to get a hold on. We were also in contact with several more companies that said that they wanted to participate but then they did not have time to meet us or answer our questions or we were not able to get in contact with them. It can be mentioned here that the time required from the company

side was answering a ten minute questionnaire and having a 30 minute interview. It is very possible that many of these companies were going through a hectic phase. It obviously takes a lot of time and effort to start a business. However, because security require special attention, if the attitude toward security is "we don't have time" and "It will have to wait until later", it is likely that security is not well preserved. As seen in the literature review, the reason why many companies fail on security is because it is considered too expensive and it is not prioritized. Company E did not answer our e-mails for months and we had to send multiple reminders. In the end when we decided to move forward without them we finally heard from them again. We decided to test them because we were curious to see if this attitude was reflected in some way in the security of the application. Because they did not show up for the interview we scheduled, we cannot say anything about how they work with security. Based on what we have seen during testing we would say that they rely on third party code to be secure, and that they themselves do not know how to develop secure code. This conclusion is based on the observation that security was very high on all requests made to third parties and extremely low on request made to the target application.

B. Do they have knowledge about common resources and tools?

None of the companies had done any prior testing of the security and hence did not use any testing tools for either static analysis or penetration testing. Company C and D both used dependency checkers and framework specific defenses to secure their applications. They also had a lot of knowledge about OWASP top 10 and company D had even done a big risk analysis. Company D were also in contact with multiple institutions to give advise and evaluate their security. Company A and B had some knowledge about OWASP but not a lot, and were not very aware of the specific OWASP top 10 vulnerabilities. All of them used third party code to handle specifically sensitive transactions like log in and credit card information. Company E used Firebase that really helped them come a long way securing their application. This can be considered a tool to achieve secure applications.

C. Are their applications protected against OWASP top 10?

All the applications contained at least one vulnerability although the severity of them differed. Overall we would say that both application C and D passed the OWASP top 10 penetration test as they had no high risk vulnerabilities. That being said the both have vulnerabilities that need to be fixed. Specifically the information leakages and the missing function level access control in application D were not insignificant. However, there will always be vulnerabilities in all applications and what is important to consider is the risk connected to them and whether the specific security needs of that application is maintained. In our opinion security on application C and D is good enough; the security is well enough maintained.

Application A and B were not well protected against OWASP top 10 with several high severity vulnerabilities. That being said, we would still feel safe using application B as it did not contain any security holes that would affect the user, also because there is no need to provide any sensitive information. If someone decided to take down application B, they probably would succeed. Although all the critical transactions on application E were handled in a secure way, they still had one high severity vulnerability present on the page.

D. Do Start-ups cover the basic needs when it comes to software security?

Considering whether the basic security needs are covered basically comes down to whether they are protected against OWASP top 10 or not. We would say that Application C and D definitely cover the basic needs that are required. There is compliance between the security needs of the application and the actual security obtained. Application A and B both did not provide even the most basic security needs. They were not well protected against basic attacks like brute forcing and SQL injection which would be the first any person with little security knowledge would try.

Application E was well protected against most vulnerabilities and all critical transactions and sensitive information were handled in a secure way. Still we would not feel safe uploading private documents on this site as we are not confident that they would be handled in a secure way throughout the application. They rely too much on third party code and the performance of the application did not demonstrate any knowledge or effort to secure the information. We feel that they do not ensure that the information is handled in a secure way, but simply assume that it is because they are using third part code. We also believe that a more thorough penetration test not limited to OWASP top 10 or the time constraints on this project might reveal more security flaws.

E. Are there tools that can help start-ups uncover vulnerabilities in a fast and easy way?

There are definitely a lot of tools that start-ups can use to help them improve security that are both free and easy to use. Snyk and Node Security Project are two examples of dependency checkers that help maintain up to date components in an automatic way. Other dependency checkers exist for frameworks not covered by these two. Furthermore, we were really impressed by Firebase both on the security level maintained and how user friendly and easy to use it was. Third party code that will handle financial transactions, log-in and storage of sensitive information is gold for inexperienced developers wanting to develop secure applications.

Kali Linux comes with a wide range of security testing tools. There is a single thing to download and set up, and it is free. Furthermore, it is very common and well used, and there exist a lot of tutorials online. On Kali Linux the specific tools found useful were, Burp Suite, openSSL, whatweb, nikto and nmap. Static analysis tools were not used in this project, but

are helpful for developers to quickly review code. We believe that at least the SQL injection vulnerability would have been prevented if a static analysis tool had been used.

VIII. Conclusion

We have performed penetration testing on applications made by five startup companies. The test plan was limited to OWASP top 10 and a number of tests from the OWASP Testing Guide were selected for each vulnerability mentioned. Although they all contained significant vulnerabilities, considering these together with the business context revealed that 2 out of 5 had maintained security in a good enough manner; Application C and D had no high risk vulnerabilities. Also in 3 out of 5 the user interests were well enough maintained; even though application B had some serious security flaws, the vulnerabilities would not lead to any loss for any of the users. For company E there was only one high risk vulnerability found where the application could be used as a stepping stone for launching a phishing attack. Although not posing a threat to the business goals of the company, this security flaw could have great consequences for innocent users.

None of the companies had done any prior security tests and they did not use any formal methodologies for implementing security. However, company D had done risk analysis and developed a document discussing how to prevent OWASP top 10.

All of the applications were using third party code to handle critical and sensitive transactions. This helped ensure that the security of the most sensitive parts of the applications was well maintained. Also the companies were using other types of tools that increased security. Examples of this were dependency checkers, Firebase and framework specific defences. Although third party code and security tools will help startups avoid common mistakes, they still have a responsibility to consider the whole system and ensure that security is maintained at all stages of the application. They need to ensure full coverage.

In OWASP top 10 2017 it is stated: "Don't stop at 10. There are hundreds of issues that could affect the overall security of a web application". Looking at OWASP top 10 only scratches the surface of the most basic security needs; to really be confident in the result, more comprehensive testing has to be performed. Also the disadvantage with penetration testing is that even if a vulnerability is not disclosed, this is no guarantee that it is not there. Other testing techniques and activities should be applied to increase security.

None of the startups were using a systematic approach to ensure security and the measures being done seemed somewhat arbitrary. This is not hard to imagine, as startups need to rely on their own knowledge because they do not have dedicated security staff.

There is always a trade off between usability and security, and between cost and risk. One of the founders of company D said: "If you are afraid to drown, don't swim". An application will never be completely free of security flaws. It is also clear that when a company is new it cannot be expected to have the experience and routines of an established company. Also, a developer just learning how to code cannot be expected to have the knowledge of a security expert. What seems to be most crucial is that the startups need to be aware that security is something that needs to be handled and what the consequences are if it is not. After only 30 minutes of talking to them about potential vulnerabilities and risks, they seemed to have whole new urgency of the matter. There are a number of helpful resources and tools free online to work with security, but it will take time, effort and experience to succeed.

Acknowledgements

This paper is based on the first author's MSc thesis at NTNU. The research in this paper has been supported in part by the Norwegian Research Council through the project SoS-Agile, grant number 247678.

References

[1] M. Ranum, "Silver bullet talks with gary mcgraw," *IEEE Security Privacy*, vol. 14, no. 5, pp. 7–10, Sept 2016.

[2] G. McGraw, "Software security: Building security in," in *17th International Symposium on Software Reliability Engineering (ISSRE 2006), 7-10 November 2006, Raleigh, North Carolina, USA*. IEEE Computer Society, 2006, p. 6. [Online]. Available: https://doi.org/10.1109/ISSRE.2006.43

[3] K. York, "Dyn statement on 10/21/2016 DDoS attack," 2016, last accessed: 2017-11-27. [Online]. Available: https://dyn.com/blog/dyn-statement-on-10212016-ddos-attack/

[4] O. Lysne, K. Beitland, J. Hagen, Å. Holmgren, E. Lunde, K. Gjøsteen, F. Manne, E. Jarbekk, and S. Nystrøm, "Digital srbarhet - sikkert samfunn," Norges offentlige utredninger, Tech. Rep., 2015.

[5] D. Geer, "Are companies actually using secure development life cycles?" *IEEE Computer*, vol. 43, no. 6, pp. 12–16, 2010. [Online]. Available: https://doi.org/10.1109/MC.2010.159

[6] J. Bau, F. Wang, E. Bursztein, P. Mutchler, and J. C. Mitchell, "Vulnerability factors in new web applications: Audit tools, developer selection & languages," Stanford University, Tech. Rep., 2012. [Online]. Available: https://seclab.stanford.edu/websec/scannerPaper.pdf

[7] owasp.org, "Top 10 2013," 2013, available online at: https://www.owasp.org/index.php/Top_10_2013-Top_10, Last accessed: 2016-10-13.

[8] BSIMM, "About the BSIMM," 2017, available online at: https://www.bsimm.com/about.html, last accessed: 2017-12-25.

[9] T. Nicolaysen, R. Sasson, M. B. Line, and M. G. Jaatun, "Agile software development: The straight and narrow path to secure software?" *IJSSE*, vol. 1, no. 3, pp. 71–85, 2010. [Online]. Available: https://doi.org/10.4018/jsse.2010070105

[10] M. G. Jaatun, D. S. Cruzes, K. Bernsmed, I. A. Tøndel, and L. Røstad, "Software security maturity in public organisations," in *Information Security - 18th International Conference, ISC 2015, Trondheim, Norway, September 9-11, 2015, Proceedings*, ser. Lecture Notes in Computer Science, J. Lopez and C. J. Mitchell, Eds., vol. 9290. Springer, 2015, pp. 120–138. [Online]. Available: https://doi.org/10.1007/978-3-319-23318-5_7

[11] G. McGraw, S. Migues, and J. West, "Building Security In Maturity Model (BSIMM V)," 2013, http://bsimm.com.

[12] A. Austin and L. Williams, "One technique is not enough: A comparison of vulnerability discovery techniques," in *Proceedings of the 5th International Symposium on Empirical Software Engineering and Measurement, ESEM 2011, Banff, AB, Canada, September 22-23, 2011*. IEEE Computer Society, 2011, pp. 97–106. [Online]. Available: https://doi.org/10.1109/ESEM.2011.18

[13] Difi, "Ulike sikkerhetsnivå," 2017, available online at: "http://eid.difi.no/nb/sikkerhet-og-informasjonskapsler/ulike-sikkerhetsniva", Last accessed: 2017-10-24.