DEPARTMENT OF COMPUTER SCIENCE | LUNDS TEKNISKA HÖGSKOLA | PRESENTED 2016-04-28

MASTER THESIS Enhancing Function Matching in Binary Code with Machine Learning Models
STUDENT Valentin Haara
SUPERVISORS Marcus Klang (LTH), Martin Ring, Samir Jasarevic, Philipp Gmaehle (Bosch)
EXAMINER Jonas Skeppstedt (LTH)

Finding out what is under the hood

POPULAR SCIENCE SYNOPSIS **Valentin Haara**

We have all used computer programs. Some that we are aware of; programs on a laptop and the apps on our smart phones. What many of us don't realise is that we also use computer programs when we drive a car or unlock our front door with a key card. What goes on in these programs and do we need to know?

As consumers, no, we don't need to know. But the manufacturers and/or the distributors of the products have a responsibility to know, in order to keep their products and merchandise safe. You may have heard of vulnerabilities in software. Vulnerabilities are sections of code that have proved to be an easy point of attack for hackers.

When computer programs are written, they may in part be assembled from already existing pieces of code from so called libraries - just like using existing screws and nuts instead of redesigning them for every new product. Of course, programmers strive to not include any vulnerabilities in their code, but what about when the vulnerability is discovered after it is already included in many programs. How should the manufacturers know what products include the newly discovered vulnerabilities?

Today it is common to keep a list of the libraries and other software components that are included in a program, a Software Bill of Materials (SBOM). Still, not all programs have an SBOM, especially not older ones. Furthermore, manufacturers may not have access to the SBOM of programs from subcontractors, but they still have a responsibility towards their customers.

In this thesis I have taken steps towards creating a tool that can tell if there are any vulnerabilities in a computer program. There are several existing tools that can create an SBOM and look for vulnerabilities in the source code. This is the raw code that the programmer writes. But the source code is only a computer program after it has been compiled, which means that it is translated, in several steps, from something a human can read to something a machine can read, and packaged in to a neat file.

What's new is that we attempt to find vulnerabilities in the program instead of the source code. For that a backwards translator, a decompiler, is needed. The backwards translator cannot find the original source code, no more than we can translate a book in to a foreign language and back and expect the same wording to come out. Instead, we use backward translation of many programs in a program database to train a machine learning model. That means the model goes through all the translations and gets an idea of what is most significant to each program. Then we give the model the backward translation of our subject program and the model can tell us if it has a lot in common with any of the programs in the database.

This tool can then be used to find out what libraries was used when the program was created and thus find out if it contains any libraries with known vulnerabilities.